
LSF Batch Administrator's Guide

Sixth Edition, August 1998

Platform Computing Corporation

LSF Batch Administrator's Guide

Copyright © 1994-1998 Platform Computing Corporation
All rights reserved.

This document is copyrighted. This document may not, in whole or part, be copied, duplicated, reproduced, translated, electronically stored, or reduced to machine readable form without prior written consent from Platform Computing Corporation.

Although the material contained herein has been carefully reviewed, Platform Computing Corporation does not warrant it to be free of errors or omissions. Platform Computing Corporation reserves the right to make corrections, updates, revisions or changes to the information contained herein.

UNLESS PROVIDED OTHERWISE IN WRITING BY PLATFORM COMPUTING CORPORATION, THE PROGRAM DESCRIBED HEREIN IS PROVIDED AS IS WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT WILL PLATFORM BE LIABLE TO ANYONE FOR SPECIAL, COLLATERAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, INCLUDING ANY LOST PROFITS OR LOST SAVINGS, ARISING OUT OF THE USE OF OR INABILITY TO USE THIS PROGRAM.

LSF Base, LSF Batch, LSF JobScheduler, LSF MultiCluster, LSF Analyzer, LSF Make, LSF Parallel, Platform Computing, and the Platform Computing and LSF logos are trademarks of Platform Computing Corporation.

Other products or services mentioned in this document are identified by the trademarks or service marks of their respective companies or organizations.

Printed in Canada

Revision Information for LSF Batch Administrator's Guide

Edition	Description
First	This document describes LSF 2.0. Based on Chapters 2 through 7 of the <i>LSF User's and Administrator's Guide</i> , Second Edition.
Second	Revised to incorporate the LSF 2.1 Release Notes.
Third	Revised to reflect the changes in LSF 2.2.
Fourth	Revised and redesigned to describe LSF Suite 3.0.
Fifth	Revised and redesigned to describe LSF Suite 3.1, and is based on the <i>LSF Administrator's Guide</i> , fourth Edition.
Sixth	Revised to reflect the changes in LSF Suite 3.2.

Contents

Preface	xiii
Audience	xiii
LSF Suite 3.2	xiii
LSF Enterprise Edition	xiv
LSF Standard Edition	xiv
Related Documents	xv
Online Documentation	xv
Technical Assistance	xv
1 - LSF Batch Concepts	1
LSF Base	1
LSF Batch	2
LSF MultiCluster	2
Definitions	3
Jobs, Tasks, and Commands	3
Hosts, Machines, and Computers	3
Clusters	3
Local and Remote Hosts	4
Submission, Master, and Execution Hosts	4
Fault Tolerance	5
Shared Directories and Files	6
Shared User Directories	7
Executables and the <code>PATH</code> Environment Variable	7
Time Windows	8
Resource and Resource Requirements	8
Shared Resources	9
Remote Execution Control	10
User Authentication Methods	10
How LSF Chooses Authentication Methods	14
Host Authentication Methods	15
User Account Mapping	16
Job Starters	16

Contents

Command-Level Job Starters	17
Queue-Level Job Starters	17
Load Sharing with LSF Base	18
How LSF Batch Schedules Jobs	19
Job States	20
Eligible Hosts	22
Dispatch Windows	23
Run Windows	23
Resource Requirements	24
Host Lists	24
Host Load Levels	24
Order of Job Dispatching	25
Job Slot Limits	26
User Job Slot Limits	27
Host Job Slot Limits	28
Queue Job Slot Limits	29
Resource Limits and Resource Usage	30
Scheduling Policies	31
Suspending Jobs	33
Resuming Suspended Jobs	35
User Suspended Jobs	35
Interactive Batch Job Support	36
Pre- and Post-execution Commands	36
Checkpointing and Migration	37
Job Migration	38
Job Control Actions	38
Resource Reservation	39
Processor Reservation	39
Remote File Access	40
Job Requeue	41
External Submission and Execution Executables	42
External Load Indices and ELIM	43
External Group Membership Definition	43
2 - Managing LSF Base	45
Managing Error Logs	45
LSF Daemon Error Log	45
FLEXlm Log	46
Controlling LIM and RES Daemons	47
Checking Host Status	47

Restarting LIM and RES.	48
Remote Startup of LIM and RES.	48
Shutting down LIM and RES	49
Locking and Unlocking Hosts	49
Managing LSF Configuration.	50
Overview of LSF Configuration Files	50
Configuration File Formats	52
Example Configuration Files.	54
Changing LIM Configuration.	55
Reconfiguring an LSF Cluster	62
External Resource Collection	63
Restrictions	64
Writing an External LIM	64
Overriding Built-In Load Indices	66
LIM Policies	66
Tuning CPU Factors	68
Tuning LIM Load Thresholds	69
Cluster Monitoring with LSF	71
LSF License Management	72
How FLEXlm Works	72
Updating an LSF License.	75
Changing the FLEXlm Server TCP Port	75
Modifying LSF Products and Licensing	76
3 - Managing LSF Batch.	79
Managing LSF Batch Logs	79
LSF Batch Accounting Log	80
LSF Batch Event Log.	80
Duplicate Event Logging	81
Configuring Duplicate Event Logging	81
How Duplicate Event Logging Works	81
Controlling LSF Batch Servers	82
LSF Batch System Status	83
Remote Start-up of sbatchd	84
Restarting sbatchd	85
Shutting Down LSF Batch Daemons	85
Opening and Closing of Batch Server Hosts	86
Controlling LSF Batch Queues.	86
bqueues — Queue Status.	86
Opening and Closing Queues.	87

Contents

Activating and Inactivating Queues	87
Managing LSF Batch Configuration	88
Adding a Batch Server Host	89
Removing a Batch Server Host	89
Adding a Batch Queue	90
Removing a Batch Queue	90
Validating Job Submissions	91
Controlling LSF Batch Jobs	96
Moving Jobs — <code>bswitch</code> , <code>btop</code> , and <code>bbot</code>	96
Signalling Jobs — <code>bstop</code> , <code>bresume</code> , and <code>bkill</code>	97
Forcing Job Execution — <code>brun -f</code>	98
Managing an LSF Cluster Using <code>xlsadmin</code>	99
<code>xlsadmin</code> Management Mode	100
<code>xlsadmin</code> Configuration Mode	102
4 - Tuning LSF Batch	107
Tuning LSF Batch	107
Controlling Interference via Load Conditions	108
Understanding Suspended Jobs	111
Controlling Fairshare	113
Hierarchical Fairshare	117
Understanding How Fairshare Works	119
Job Dispatching According to Fairshare	120
Limits and Windows	121
Dispatch and Run Windows	121
Controlling Job Slot Limits	122
Resource Limits	122
Reservation Based Scheduling	122
Resource Reservation	122
Processor Reservation and Backfilling	123
Controlling Job Execution	126
Understanding Job Execution Environment	126
Environment Variable Handling	128
NICE Value	129
Pre-execution and Post-execution commands	129
Queue-Level Job Starters	129
Using Licensed Software with LSF Batch	131
Host Locked Licenses	131
Host Locked Counted Licenses	131
Floating Licenses	132

Example LSF Batch Configuration Files	136
Example Queues	136
Example <code>lsb.hosts</code> file	140
5 - Managing LSF MultiCluster	143
What is LSF MultiCluster?	143
Enabling MultiCluster Functionalities	144
The <code>lsf.shared</code> File	145
The <code>lsf.cluster.cluster</code> File	146
Root Access	148
LSF Batch Configuration	148
Remote-Only MultiCluster Queues	149
Inter-cluster Load and Host Information Sharing	150
Running Interactive Jobs on Remote Clusters	152
Distributing Batch Jobs Across Clusters	153
Account Mapping Between Clusters	155
User Level Account Mapping	156
System Level Account Mapping	157
6 - LSF Base Configuration Reference	161
The <code>lsf.conf</code> File	161
The <code>lsf.shared</code> File	173
Clusters	173
Host Types	174
Host Models	174
Resources	175
The <code>lsf.cluster.cluster</code> File	178
Parameters	178
LSF Administrators	181
Hosts	182
Resource Map	185
The <code>lsf.task</code> and <code>lsf.task.cluster</code> Files	187
The <code>hosts</code> File	188
The <code>lsf.sudoers</code> File	189
7 - LSF Batch Configuration Reference	193
The <code>lsb.params</code> File	193
Parameters	193
Handling Cray NQS Incompatibilities	197
The <code>lsb.users</code> File	198

Contents

UNIX/NT User Groups	198
LSF Batch User Groups	198
Share Tree Defined in User Groups	199
External User Groups	200
User and Group Job Slot Limits	200
The <code>lsb.hosts</code> File	202
Host Section	202
Host Groups	205
External Host Groups	206
Host Partitions	206
The <code>lsb.queues</code> File	208
General Parameters	208
Processor Reservation for Parallel Jobs	211
Backfill Scheduling	211
Deadline Constraint Scheduling	212
Flexible Expressions for Queue Scheduling	213
Load Thresholds	216
Resource Limits	217
Eligible Hosts and Users	220
Scheduling Policy	221
Migration	224
Queue-Level Pre-/Post-Execution Commands	224
Job Starter	227
Configurable Job Control Actions	228
Automatic Job Requeue	231
Exclusive Job Requeue	232
Default Host Specification for CPU Speed Scaling	232
NQS Forward Queues	233
Queue Level Checkpoint and Rerun	234
The <code>lsb.nqsmaps</code> File	235
Hosts	235
Users	237
A - Troubleshooting and Error Messages	239
Error Log Messages	239
Finding the Error Logs	239
Shared File Access	240
Shared Files Across UNIX and NT	241
Common LSF Base Problems	241
LIM Dies Quietly	241

LIM Unavailable	241
RES Does Not Start	242
User Permission Denied	242
Non-uniform File Name Space	243
Common LSF Batch Problems	244
Batch Daemons Die Quietly	244
sbatchd Starts But mbatchd Does Not	244
Host Not Used By LSF Batch	244
Error Messages	245
General Errors	245
Configuration Errors	248
LIM Messages	249
RES Messages	251
LSF Batch Messages	252
B - LSF Directories	255
C - Sample System Support	257
IRIX 6 Processor Sets	257
Time-Based Processor Allocation	258
User-Based Processor Allocation	259
Other Situations	259
Support for Solaris Processor Sets	260
Time-Based Processor Allocation	260
User-Based Processor Allocation	262
Other Situations	263
IBM SP-2 Support	263
Support for HP Exemplar Technical Servers	265
Adding Load Indices Definitions	266
Adding Queue Definitions	267
Configuring NQS Interoperation	269
Registering LSF with NQS	270
lsb.nqsmaps	271
Configuring Queues for NQS jobs	272
Handling Cray NQS Incompatibilities	273
Support for Atria ClearCase	275
Using LSF Without Shared File Systems	277
D - LSF on Windows NT	279
Requirements	279

Contents

Recommended	279
Differences Between LSF for UNIX and NT	279
File Permissions	281
Mail	282
The cmd.exe Program	282
Heterogeneous NT/UNIX Environments	283
User Accounts	283
Configuration Files	284
Environment Variables	284
Cross-Platform Daemon Startup	285
Signal Conversion	285
Starting Services and Daemons	286
Using LSF	287
Miscellaneous	287
E - The LSF SNMP Agent	289
About the Agent	289
Requirements	289
Distribution	290
Starting the Agent	290
Structure of the LSF MIB	291
The lsfHosts MIB Group	291
The lsfResources MIB Group	291
The lsfBatch MIB Group	292
Optional Configuration of the Agent	292
Index	295

Preface

Audience

In this book, you will find all the information you need to configure and maintain your LSF Base, LSF Batch, or LSF MultiCluster installation. This guide assumes you have knowledge of common system administration tasks such as exporting and mounting Network File System (NFS) partitions.

The focus of this guide is the administration of LSF Base, LSF Batch, and LSF MultiCluster, and as such is intended for LSF cluster administrators who manage LSF Base, LSF Batch, and LSF MultiCluster. Users who wish to understand the details of LSF operation should also read this guide.

It is assumed that you have already read the *LSF Installation Guide* and installed one or more products from the LSF Suite at your site.

LSF Suite 3.2

LSF is a suite of workload management products including the following:

LSF Batch is a batch job processing system for distributed and heterogeneous environments, which ensures optimal resource sharing.

LSF JobScheduler is a distributed production job scheduler that integrates heterogeneous servers into a virtual mainframe or virtual supercomputer

Preface

LSF MultiCluster supports resource sharing among multiple clusters of computers using LSF products, while maintaining resource ownership and cluster autonomy.

LSF Analyzer is a graphical tool for comprehensive workload data analysis. It processes cluster-wide job logs from LSF Batch and LSF JobScheduler to produce statistical reports on the usage of system resources by users on different hosts through various queues.

LSF Parallel is a software product that manages parallel job execution in a production networked environment.

LSF Make is a distributed and parallel Make based on GNU Make that simultaneously dispatches tasks to multiple hosts.

LSF Base is the software upon which all the other LSF products are based. It includes the network servers (LIM and RES), the LSF API, and load sharing tools.

There are two editions of the LSF Suite:

LSF Enterprise Edition

Platform's LSF Enterprise Edition provides a reliable, scalable means for organizations to schedule, analyze, and monitor their distributed workloads across heterogeneous UNIX and Windows NT computing environments. LSF Enterprise Edition includes all the features in LSF Standard Edition (LSF Base and LSF Batch), plus the benefits of LSF Analyzer and LSF MultiCluster.

LSF Standard Edition

The foundation for all LSF products, Platform's Standard Edition consists of two products, LSF Base and LSF Batch. LSF Standard Edition offers users robust load sharing and sophisticated batch scheduling across distributed UNIX and Windows NT computing environments.

Related Documents

The following guides are available from Platform Computing Corporation:

LSF Installation Guide
LSF Batch Administrator's Guide
LSF Batch Administrator's Quick Reference
LSF Batch User's Guide
LSF Batch User's Quick Reference
LSF JobScheduler Administrator's Guide
LSF JobScheduler User's Guide
LSF Analyzer User's Guide
LSF Parallel User's Guide
LSF Programmer's Guide

Online Documentation

- Man pages (accessed with the `man` command) for all commands
- Online help available through the Help menu for the `xlsbatch`, `xbmod`, `xbsub`, `xbalarms`, `xbcal` and `xlsadmin` applications.

Technical Assistance

If you need any technical assistance with LSF, please contact your reseller or Platform Computing's Technical Support Department at the following address:

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

Tel: +1 905 948 8448
Toll-free: 1-87PLATFORM (1-877-528-3676)
Fax: +1 905 948 9975
Electronic mail: support@platform.com

Preface

Please include the full name of your company.

You may find the answers you need from Platform Computing Corporation's home page on the World Wide Web. Point your browser to *www.platform.com*.

If you have any comments about this document, please send them to the attention of LSF Documentation at the address above, or send email to *doc@platform.com*.

1. LSF Batch Concepts

LSF is a suite of workload management products that schedule, monitor, and analyze the workload for a network of computers. LSF Batch allows you, as a system administrator, to control and manage all of your computing resources effectively and efficiently.

LSF consists of a set of daemons that provide workload management services across the whole cluster, an API that allows access to such services at the procedure level, and a suite of tools or utilities that end users can use to access such services at the command or GUI level.

This chapter introduces important LSF concepts related to the design and operation of LSF Batch.

LSF Base

LSF Base provides basic load-sharing services across a heterogeneous network of computers. It is the base software upon which all other LSF products are built. It provides services such as resource information, host selection, placement advice, transparent remote execution, and remote file operation.

LSF Base includes Load Information Manager (LIM), Remote Execution Server (RES), the LSF Base API, and LSF Base Tools that allow the use of the LSF Base system to run simple load-sharing applications and `lscsh`, a load sharing enabled C shell.

An LSF Base cluster contains a network of computers running LIM, RES, and associated tools. The cluster is defined by LSF cluster configuration files, which are read by LIM. LIM then provides the cluster configuration information, together with all other dynamic information to the rest of the LSF Base system, as well as to other LSF products.

1 LSF Batch Concepts

LSF Base system API allows users to write their own load-sharing applications on top of the LSF Base system.

LSF Batch

LSF Batch is a distributed batch queuing system built on top of the LSF Base. The services provided by LSF Batch are extensions to the LSF Base system services. LSF Batch makes a computer network a network computer. It has all the features of a mainframe batch job processing system, as well as load balancing and policy-driven resource allocation control. LSF Batch implements sophisticated job scheduling and resource control for batch workload.

LSF Batch relies on services provided by the LSF Base system. It makes use of the resource and load information from the LIM to perform load balancing. LSF Batch also uses the cluster configuration information from LIM and follows the master election service provided by LIM. LSF Batch uses RES for interactive batch job execution and uses the remote file operation service provided by RES for file transfer. LSF Batch includes a Master Batch Daemon (`mbatchd`) running on the master host and a slave Batch Daemon (`sbatchd`) running on each batch server host.

LSF Batch has its own configuration files, it also uses the cluster configuration from the LSF Base system.

LSF MultiCluster

LSF MultiCluster extends the capabilities of LSF Base and LSF Batch by sharing the resources of an organization across multiple cooperating clusters of computers. Load-sharing happens not only within the clusters, but also among them. Resource ownership and autonomy is enforced, non-shared user accounts and file systems are supported, and communication limitations among the clusters are also considered in job scheduling.

Definitions

In the rest of this document, LSF refers to LSF Base, Batch, and Multicluster, unless otherwise explicitly mentioned.

Jobs, Tasks, and Commands

This document uses the terms *job*, *task*, and *command* to refer to one or more processes invoked together to perform some action. The terms are interchangeable, though *task* is more often used to refer to interactive commands and *job* is more often used for commands run using the batch system.

Each command can be a single process, or it can be a group of cooperating processes. LSF creates a new process group for each command it runs, and the job control mechanisms act on all processes in the process group.

Hosts, Machines, and Computers

This document uses the terms *host*, *machine*, and *computer* to refer to a single computer, which may have more than one processor. An informal definition is as follows: if it runs a single copy of the operating system and has a unique Internet (IP) address, it is one computer. More formally, LSF treats each process queue as a separate machine. A multiprocessor computer with a single process queue is considered a single machine, while a box full of processors that each have their own process queue is treated as a group of separate machines.

Clusters

A *cluster* is a group of hosts that provide shared computing resources. Hosts can be grouped into clusters in a number of ways. A cluster could contain:

- All the hosts in a single administrative group
- All the hosts on one file server or sub-network
- Hosts that perform similar functions.

1 LSF Batch Concepts

If you have hosts of more than one type, it is often convenient to group them together in the same cluster. LSF allows you to use these hosts transparently, so applications that run on only one host type are available to the entire cluster.

Local and Remote Hosts

When LSF runs a remote command, two hosts are involved. The host where the remote execution is initiated is the *local host*. The host where the command is executed is the *remote host*. For example, in this sequence:

```
hostA% lsrun -v uname
<<Execute uname on remote host hostD>>
HP-UX
```

Here, the local host is *hostA*, and the remote host is *hostD*. Note that it is possible for the local and remote hosts to be the same.

Submission, Master, and Execution Hosts

When LSF Batch runs a job, three hosts are involved. The host from which the job is submitted is the *submission host*. The job information is sent to the *master host*, which is the host where the master LIM and `mbatchd` are running. The job is run on the *execution host*. It is possible for more than one of these to be the same host.

The master host is displayed by the `lsid` command:

```
% lsid

LSF 3.2, Aug 1, 1998
Copyright 1992-1998 Platform Computing Corporation
My cluster name is test_cluster
My master name is hostA
```

The following example shows the submission and execution hosts for a batch job:

```
hostD% bsub sleep 60
Job <1502> is submitted to default queue <normal>.
hostD% bjobs 1502
JOBID USER  STAT  QUEUE  FROM_HOST  EXEC_HOST  JOB_NAME  SUBMIT_TIME
1502  user2  RUN   normal  hostD      hostB      sleep 60  Nov 22 14:03
```

The master host is *hostA*, as shown by the `lsid` command. The submission host is *hostD*, and the execution host is *hostB*.

Fault Tolerance

LSF has a number of features that support fault tolerance. LSF can tolerate the failure of any host or group of hosts in the cluster.

The LSF master host is chosen dynamically. If the current master host becomes unavailable, another host takes over automatically. The master host selection is based on the order in which hosts are listed in the `lsf.cluster.cluster` file. If the first host in the file is available, that host acts as the master. If the first host is unavailable, the second host takes over, and so on. LSF might be unavailable for a few minutes while hosts are waiting to be contacted by the new master.

If the cluster is partitioned by a network failure, a master LIM takes over on each side of the partition. Interactive load-sharing remains available, as long as each host still has access to the LSF executables.

Fault tolerance in LSF Batch depends on the event log file, `lsb.events`, which is kept on the primary file server. Every event in the system is logged in this file, including all job submissions and job and host status changes. If the master host becomes unavailable, a new master is chosen by the LIMs. The slave batch daemon `sbatchd` on the new master starts a new master batch daemon `mbatchd`. The new `mbatchd` reads the `lsb.events` file to recover the state of the system.

For sites not wanting to rely solely on a central file server for recovery information, LSF can be configured to maintain a duplicate event log by keeping a replica of the `lsb.events` file. The replica is stored on the file server, and used if the primary copy is unavailable. When using LSF's duplicate event log function, the primary event log

1 LSF Batch Concepts

is stored on the first master host, and re-synchronized with the replicated copy when the host recovers.

If the network is partitioned, only one of the partitions can access the `lsb.events` log, so batch services are only available on one side of the partition. A lock file is used to guarantee that only one `mbatchd` is running in the cluster.

Running jobs are managed by the `sbatchd` on each batch server host. When the new `mbatchd` starts up it polls the `sbatchd` daemons on each host and finds the current status of its jobs. If the `sbatchd` fails but the host is still running, jobs running on the host are not lost. When the `sbatchd` is restarted it regains control of all jobs running on the host.

If an LSF server host fails, jobs running on that host are lost. No other jobs are affected. LSF Batch jobs can be submitted so that they are automatically rerun from the beginning or restarted from a checkpoint on another host if they are lost because of a host failure.

If all of the hosts in a cluster go down, all running jobs are lost. When a host comes back up and takes over as master, it reads the `lsb.events` file to get the state of all batch jobs. Jobs that were running when the systems went down are assumed to have exited, and email is sent to the submitting user. Pending jobs remain in their queues, and are scheduled as hosts become available.

Shared Directories and Files

LSF is designed for networks where all hosts have shared file systems, and files have the same names on all hosts.

UNIX

On UNIX systems, LSF supports the Network File System (NFS), the Andrew File System (AFS), and DCE's Distributed File System (DFS). NFS file systems can be mounted permanently or on demand using `automount`.

NT

On Windows NT, directories containing LSF files can be shared amongst hosts from an NT server machine.

LSF includes support for copying user data to the execution host before running a batch job, and for copying results back after the job executes. In networks where the file systems are not shared, this can be used to give remote jobs access to local data.

For more information about running LSF on networks where no shared file space is available, see *'Using LSF Without Shared File Systems'* on page 277.

Shared User Directories

To provide transparent remote execution, LSF commands determine the user's current working directory and use that directory on the remote host. For example, if the command `cc file.c` is executed remotely, `cc` only finds the correct `file.c` if the remote command runs in the same directory.

The LSF Batch automatically creates an `.lsbatch` subdirectory in the user's home directory on the execution host. This directory is used to store temporary input and output files for jobs.

Executables and the `PATH` Environment Variable

Search paths for executables (the `PATH` environment variable) are passed to the remote execution host unchanged. In mixed clusters, LSF works best when the user binary directories (for example, `/usr/bin`, `/usr/local/bin`) have the same path names on different host types. This makes the `PATH` variable valid on all hosts.

If your LSF user binaries are NFS mounted, place all binaries in a shared file system under `/usr/local/lsf/mnt` (or some similar name), and then make a symbolic link from `/usr/local/bin` to `/usr/local/lsf/mnt/bin/type` for the correct host type on each machine. These are the default install directories.

LSF configuration files are normally stored in a shared directory. This makes administration easier. There is little performance penalty for this, because the configuration files are not frequently read.

For more information on LSF installation directories see *'LSF Directories'* on page 115 of the *LSF Installation Guide*.

Time Windows

Time windows are an important concept in LSF. Time windows are a useful means to control resource access such that you can disable access to some resources during certain times. A time window is the basic building block for configuring dispatch windows and run windows.

A time window is specified by two time values separated by '-'. Each time value is specified by up to three fields are shown below:

[day :] hour [: min]

If only one field exists, it is assumed to be *hour*; if two fields exist, they are assumed to be *hour:min*. Days are numbered from 0 (Sunday) to 6 (Saturday). Hours are numbered from 0 to 23, and minutes from 0 to 59.

In a time window *time1-time2*, if neither *time1* nor *time2* specifies a day, the time window applies to every day of the week. If *time1* is greater than *time2*, the time window applies from *time1* of each day until *time2* of the following day.

If either *time1* or *time2* specifies a day, the other must specify a day. If *time1* is on a later day of the week than *time2*, or is a later time on the same day, then the time window applies from *time1* of each week until *time2* of the following week.

A dispatch or run window is specified as a series of time windows. When a dispatch or run window specification includes more than one time window, the window is open if any of the time windows are open. The following example specifies that the host is available only during weekends (Friday evening at 19:00 until Monday morning at 08:30) and during nights (20:00 to 08:30 every day).

```
5:19:00-1:8:30 20:00-8:30
```

Resource and Resource Requirements

LSF provides a powerful means for you to describe your heterogeneous cluster in terms of resources. One of the most important decisions LSF makes when scheduling a job is to map a job's resource requirements to resources available on individual hosts.

There are several types of resources. *Load indices* measure dynamic resource availability such as a host's CPU load or available swap space. *Static resources* represent unchanging information such as the number of CPUs a host has, the host type, and the maximum available swap space.

Resources can also be described in terms of where they are located. For example, a *shared resource* is a resource that is associated with the entire cluster or a subset of hosts within the cluster.

Resource names can be any string of characters, excluding the characters reserved as operators. The `lsinfo` command lists the resources available in your cluster.

For a complete description of resources and how they are used, see *Section 4, 'Resources'*, beginning on page 35 of the *LSF Batch User's Guide*.

To best place a job with optimized performance, resource requirements can be specified for each application. A resource requirement is an expression that contains resource names and operators. Resource requirements can be configured for individual applications, or specified for each job. The detailed format for resource requirements can be found in *'Resource Requirement Strings'* on page 46 of the *LSF Batch User's Guide*.

Shared Resources

A shared resource is a resource that is associated with the entire cluster or a subset of hosts within the cluster. In contrast to host-based resources such as memory or swap space, using a shared resource from one machine affects the availability of that resource as seen by other machines. Common examples of shared resources include floating licenses for software packages, shared file systems, and network bandwidth. LSF provides a mechanism to configure which machines share a particular resource and to monitor the availability of those resources. LSF Batch jobs can be scheduled based on the availability of shared resources.

Remote Execution Control

There are two aspects to controlling access to remote execution. The first requirement is to authenticate the user. When a user executes a remote command, the command must be run with that user's permission. The LSF daemons need to know which user is requesting the remote execution. The second requirement is to check access controls on the remote host. The user must be authorized to execute commands remotely on the host.

User Authentication Methods

LSF supports user authentication using external authentication (the default). On UNIX, LSF also supports user authentication using privileged ports and using the RFC 931 or RFC 1413 identification protocols.

UNIX Authentication Using Privileged Ports

If a load-sharing program is owned by *root* and has the *setuid* bit set, the LSF API functions use a privileged port to communicate with LSF servers, and the servers accept the user ID supplied by the caller. This is the same user authentication mechanism as used by *rlogin* and *rsh*.

When a *setuid* application calls the `L_SLIB` initialization routine, a number of privileged ports are allocated for remote connections to LSF servers. The effective user ID then reverts to the real user ID. Therefore, the number of remote connections is limited. Note that an LSF utility reuses the connection to the RES for all remote task executions on that host, so the number of privileged ports is only a limitation on the number of remote hosts that can be used by a single application, not on the number of remote tasks. Programs using `L_SLIB` can specify the number of privileged ports to be created at initialization time.

UNIX Authentication Using Identification Daemons

The RFC 1413 and RFC 931 protocols use an identification daemon running on each client host. Using an identification daemon incurs more overhead, but removes the need for LSF applications to allocate privileged ports. All

LSF commands except `lsadmin` can be run without `setuid` permission if an identification daemon is used.

You should use identification daemons if your site cannot install programs owned by `root` with the `setuid` bit set, or if you have software developers creating new load-sharing applications in C using LSLIB.

An implementation of RFC 931 or RFC 1413 such as `pidentd` or `authd` can be obtained from the public domain (if you have access to Internet FTP, a good source for ident daemons is host `ftp.lysator.liu.se`, directory `pub/ident/servers.`). RFC 1413 is a more recent standard than RFC 931. LSF is compatible with both.

External Authentication

When an LSF client program is invoked (for example, `lsrun`), the client program automatically executes `eauth -c hostname` to get the external authentication data. `hostname` is the name of the host running the LSF daemon (for example, `RES`) on which the operation will take place. The external user authentication data can be passed to LSF via `eauth`'s standard output.

When the LSF daemon receives the request, it executes `eauth -s` under the primary LSF administrator user ID. The parameter `LSF_EAUTH_USER` must be configured in the `/etc/lsf.sudoers` file if your site needs to run authentication under another user ID (see *'The lsf.sudoers File' on page 189* for details). `eauth -s` is executed to process the user authentication data. The data is passed to `eauth -s` via its standard input. The standard input stream has the following format:

```
uid gid username client_addr client_port user_auth_data_len user_auth_data
```

where:

- *uid* is the user ID in ASCII of the client user.
- *gid* is the group ID in ASCII of the client user.
- *username* is the user name of the client user.
- *client_addr* is the host address of the client host in ASCII dot notation.

1 LSF Batch Concepts

- `client_port` is the port number from where the client request is made.
- `user_auth_data_len` is the length of the external authentication data in ASCII passed from the client.
- `user_auth_data` is the external user authentication data passed from the client.

The LSF daemon expects `eauth -s` to write 1 to its standard output if authentication succeeds, or 0 if authentication fails.

The same `eauth -s` process can service multiple authentication requests; if the process terminates, the LSF daemon will re-invoke `eauth -s` on the next authentication request.

By default, `eauth` uses an internal key to encrypt authentication data. To use an external key to improve the security, configure the parameter `LSF_EAUTH_KEY` in the `lsf.sudoers` file (see [page 165](#)).

You can configure your own user authentication scheme using the `eauth` mechanism of LSF. To use external authentication, an executable called `eauth` must be installed in `LSF_SERVERDIR`. This is set up automatically during the installation.

You may choose to write your own `eauth` executable and use it instead of the LSF default. Example uses of external authentication include support for Kerberos 4 and DCE client authentication using the GSSAPI. These examples can be found in the `examples/krb` and `examples/dce` directories in the standard LSF distribution. Installation instructions are found in the `README` file in these directories.

Security of LSF Authentication

All authentication methods supported by LSF depend on the security of the root account on all hosts in the cluster. If a user can get access to the root account, they can subvert any of the authentication methods. There are no known security holes that allow a non-root user to execute programs with another user's permission.

Some people have particular concerns about security schemes involving RFC 1413 identification daemons. When a request is coming from an unknown host, there is no way to know whether the identification daemon on that host is correctly identifying the originating user.

LSF only accepts job execution requests that originate from hosts within the LSF cluster, so the identification daemon can be trusted.

UNIX

The identification protocol uses a port in the UNIX privileged port range, so it is not possible for an ordinary user to start a hacked identification daemon on an LSF host.

NT

On Windows NT, external authentication is installed automatically. You may disable external authentication by disabling the `LSF_AUTH` parameter in the `lsf.conf` file.

On UNIX, this means that authentication is done using privileged ports and binaries that need to be authenticated (for example, `bsub`) to `setuid root`.

On Windows NT, this does not provide any security because Windows NT does not have the concept of `setuid` binaries and does not restrict which binaries can use privileged ports. A security risk exists in that a user can discover the format of LSF protocol messages and write a program that tries to communicate with an LSF server. External authentication should be used where this security risk is a concern.

The system environment variable `LSF_ENVDIR` is used by LSF to obtain the location of `lsf.conf` which points to important configuration files. Any user who can modify system environment variables can modify `LSF_ENVDIR` to point to their own configuration and start up programs under the `lsfadmin` account.

Once the LSF Service is started, it will only accept requests from the cluster administrator accounts specified during the installation. To allow other users to interact with the LSF Service, you must set up the `lsf.sudoers` file under the directory specified by the `SYSTEMROOT` environment variable. See *'The lsf.sudoers File'* on page 189 for the format of the `lsf.sudoers` file.

Note

Only the `LSF_STARTUP_USERS` and `LSF_STARTUP_PATH` are used on NT. You should ensure that only authorized users modify the files under the `SYSTEMROOT` directory.

1 LSF Batch Concepts

All external binaries invoked by the LSF daemons (such as `esub`, `eexec`, `elim`, `eauth`, and queue level pre- and post-execution commands) are run under the `lsfadmin` account.

How LSF Chooses Authentication Methods

LSF uses the `LSF_AUTH` parameter in the `lsf.conf` file to determine which type of authentication to use.

If an LSF application is not *setuid* to *root*, library functions use a non-privileged port. If the `LSF_AUTH` flag is not set in the `/etc/lsf.conf` file, the connection is rejected. If `LSF_AUTH` is defined to be `ident` the RES on the remote host, or the `mbatchd` in the case of a `bsub` command, contacts the identification daemon on the local host to verify the user ID. The identification daemon looks directly into the kernel to make sure the network port number being used is attached to a program being run by the specified user.

LSF allows both the *setuid* and authentication daemon methods to be in effect simultaneously. If the effective user ID of a load-sharing application is *root*, then a privileged port number is used in contacting the RES. RES always accepts requests from a privileged port on a known host even if `LSF_AUTH` is defined to be `ident`. If the effective user ID of the application is not *root*, and the `LSF_AUTH` parameter is defined to be `ident`, then a normal port number is used and RES tries to contact the identification daemon to verify the user's identity.

External user authentication is used if `LSF_AUTH` is defined to be `eauth`. In this case, LSF will run the external executable `eauth` in the `LSF_SERVERDIR` directory to perform the authentication.

The error message "User permission denied" is displayed by `lshrun`, `bsub`, and other LSF commands if LSF cannot verify the user's identity. This might be because the LSF applications are not installed *setuid*, the NFS directory is mounted with the `nosuid` option, the identification daemon is not available on the local or submitting host, or the external authentication failed.

If you change the authentication type while the LSF daemons are running, you will need to run the command `lsfdaemons start` on each of the LSF server hosts so that the daemons will use the new authentication method.

Host Authentication Methods

When a batch job or a remote execution request is received, LSF first determines the user's identity. Once the user's identity is known, LSF decides whether it can trust the host from which the request comes from.

Trust LSF Host

LSF normally allows remote execution by all users except *root*, from all hosts in the LSF cluster; LSF trusts all hosts that are configured into your cluster. The reason behind this is that by configuring an LSF cluster you are turning a network of machines into a single computer. Users must have valid accounts on all hosts. This allows any user to run a job with their own permission on any host in the cluster. Remote execution requests and batch job submissions are rejected if they come from a host not in the LSF cluster.

A site can configure an external executable to perform additional user or host authorization. By defining `LSF_AUTH` to be `eauth`, the LSF daemon will invoke `eauth -s` when it receives a request that needs authentication and authorization. As an example, this `eauth` can check if the client user is on a list of authorized users or if a host has the necessary privilege to be trusted.

UNIX Using `/etc/hosts.equiv`

If the `LSF_USE_HOSTEQUIV` parameter is set in the `lsf.conf` file, LSF uses the same remote execution access control mechanism as the `rsh` command. When a job is run on a remote host, the user name and originating host are checked using the `ruserok(3)` function on the remote host.

This function checks in the `/etc/hosts.equiv` file and the user's `$HOME/.rhosts` file to decide if the user has permission to execute jobs.

The name of the local host should be included in this list. `RES` calls `ruserok()` for connections from the local host. `mbatchd` calls `ruserok()` on the master host, so every LSF Batch user must have a valid account and remote execution permission on the master host.

The disadvantage of using the `/etc/hosts.equiv` and `$HOME/.rhosts` files is that these files also grant permission to use the `rlogin` and `rsh`

1 LSF Batch Concepts

commands without giving a password. Such access is restricted by security policies at some sites.

See the *hosts.equiv(5)* and *ruserok(3)* manual pages for details on the format of the files and the access checks performed.

The error message “User permission denied” is displayed by `lssrun`, `bsub`, and other LSF commands if you configure LSF to use `ruserok()` and the client host is not found in either the `/etc/hosts.equiv` or the `$HOME/.rhosts` file on the master or remote host.

User Account Mapping

By default, LSF assumes uniform user accounts throughout the cluster. This means that job will be executed on any host with exactly the same user ID and user login name.

LSF Batch has a mechanism to allow user account mapping across dissimilar name spaces. Account mapping can be done at the individual user level and system level. Individual users of the LSF cluster can set up their own account mapping by setting up an `.lsfhosts` file in their home directories. See ‘*User-Level Account Mapping Between Clusters*’ on page 192 of the *LSF Batch User’s Guide* for details of user level account mapping. An LSF administrator can set up system-level account mapping in the `lsb.users` file. See ‘*System Level Account Mapping*’ on page 157 for details.

The LSF administrator can disable user account mapping.

Job Starters

A job starter is a specified command (or set of commands) that executes immediately prior to a submitted batch job or an interactive job. This can be useful if you are submitting or running jobs that require specific setup steps to be performed before execution, or jobs that must be executed in a specific environment. Any situation in which you would ordinarily write a wrapper around the job you want executed is a candidate for a job starter.

There are two types of job starters in LSF: command-level and queue-level. A command-level job starter is user-defined, and precedes interactive jobs (submitted using `lsrun`, for example). A queue-level job starter is defined by the LSF administrator, and precedes batch jobs submitted to a specific queue (for example, using `bsub` or `xbsub`).

You can accomplish similar things with either job starter, but their functional details are slightly different.

Command-Level Job Starters

Individual users can select an existing command to be a job starter, or they can create a script containing a desired set of commands to serve as a job starter. Setting the `LSF_JOB_STARTER` environment variable to the selected command or script causes that command or script to be executed immediately before an interactive job.

UNIX For example, when a command-level job starter is defined as `"/bin/ksh -c"`, interactive jobs will be run under a Korn shell environment.

NT For example, when a command-level job starter is defined as `"C:\cmd.exe /C"`, interactive jobs will run under a DOS shell environment.

When a job is run with a command-level job starter defined, LSF's Remote Execution Server runs the job starter rather than running the job itself, which is passed to the job starter as a command-line argument.

Command-level job starters have no effect on batch jobs, including interactive batch jobs (see *'Interactive Batch Job Support'* on page 36 for information on interactive batch jobs).

See *'Command-Level Job Starters'* on page 144 of the *LSF Batch User's Guide* for detailed information about setting up and using a command-level job starter to run interactive jobs.

Queue-Level Job Starters

The LSF administrator can select an existing command to be a job starter, or create a script containing a desired set of commands to serve as a job starter. Setting the

1 LSF Batch Concepts

`JOB_STARTER` parameter in the queue definition (contained in the `lsb.queues` file) to the selected command or script causes that command or script to be executed immediately before all batch jobs submitted to that queue are executed.

UNIX For example, by defining a queue-level job starter as “`xterm -e`”, all jobs in the queue will run in an X terminal window.

NT For example, by defining a queue-level job starter as “`C:\cmd.exe /C`”, all jobs in the queue will run under the DOS shell environment.

Queue-level job starters have no effect on interactive jobs, unless the interactive job is submitted to a queue as an interactive batch job (see ‘*Interactive Batch Job Support*’ on page 36 for information on interactive batch jobs).

See ‘*Queue-Level Job Starters*’ on page 129 for detailed information about defining a job starter for an LSF queue.

Load Sharing with LSF Base

LSF Base system provides a very basic level of services that allow you to perform load-sharing and distributed processing. This is implemented via the LSF Base system services. Many utilities of the LSF Base system use the basic services for placement decision, host selection, and remote execution.

LIM provides convenient services that help job placement, host selection, and load information that are essential to the scheduling of jobs. `lsrun` and `lsgrun`, for example, use the LIM’s placement advice to run jobs on the least loaded yet most powerful hosts. When LIM gives placement advice, it takes into consideration many factors, such as current load information, job’s resource requirements, and configured policies in the LIM cluster configuration file.

RES provides transparent and efficient remote execution and remote file operation services so that jobs can be easily shipped to anywhere in the network once a placement decision has been made. Files can be accessed easily from anywhere in the network using remote file operation services.

The LSF Base provides sufficient services to many simple load-sharing applications and utilities, as exemplified by `LSF Base tools` and `lstcsh`. If sophisticated job scheduling and resource allocation policies are necessary, more complex scheduling must be built on top of the LSF Base, such as LSF Batch. Since the placement service from LIM is just advice, LSF Batch makes its own placement decision based on advice from LIM as well as further policies that the site configures.

How LSF Batch Schedules Jobs

LSF Batch provides a rich collection of mechanisms for controlling the sharing of resources by jobs. Most sites do not use all of them; a few would provide enough control. However, it is important that you be aware of all of them to understand how LSF Batch works and to choose suitable controls for your site. More discussions of job scheduling policies are given in *‘Tuning LSF Batch’ on page 107*.

When a job is placed on an LSF Batch queue, many factors control when and where the job starts to run:

- active time window of the queue or hosts
- resource requirements of the job
- availability of eligible hosts
- various job slot limits
- job dependency conditions
- fairshare constraints
- load conditions.

When LSF Batch is trying to place a job, it obtains current load information for all hosts from LIM. The load levels on each host are compared to the scheduling thresholds configured for that host in the `Host` section of the `lsb.hosts` file, as well as the per-queue scheduling thresholds configured in the `lsb.queues` file. If any load index exceeds either its per-queue or its per-host scheduling threshold, no new job is started

1 LSF Batch Concepts

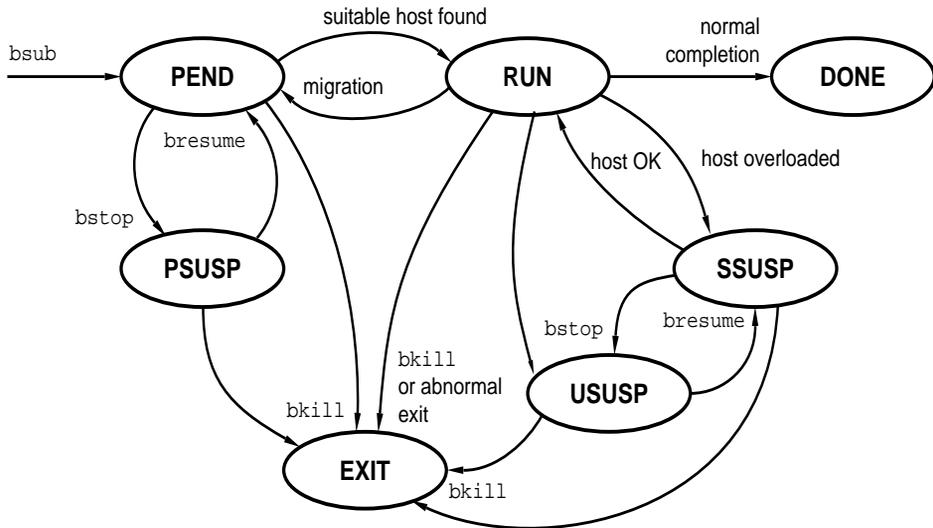
on that host. When a job is running, LSF Batch periodically checks the load level on the execution host. If any load index is beyond either its per-host or its per-queue suspending conditions, the lowest priority batch job on that host is suspended.

LSF Batch supports both batch jobs and interactive jobs. So by configuring appropriate resource allocation policies, all workload in your cluster can be managed by LSF Batch.

Job States

An LSF Batch job goes through a series of state transitions until it eventually completes its task, fails, or is terminated. The possible states of a job during its life cycle are shown in the diagram below.

Figure 1. Batch Job States



Many jobs enter only three states:

PEND:
waiting in the queue

RUN:
dispatched to a host and running

DONE:

terminated normally

A job remains pending until all conditions for its execution are met. Some of the conditions are:

- start time specified by the user when the job is submitted
- load conditions on qualified hosts
- dispatch windows during which the queue can dispatch and qualified hosts can accept jobs
- run windows during which jobs from the queue can run
- limits on the number of job slots configured for a queue, a host, or a user
- relative priority to other users and jobs
- availability of the specified resources
- job dependency and pre-execution conditions.

The `bjobs -lp` command displays the names of hosts that cannot accept a job at the moment together with the reasons the job cannot be accepted.

A job might terminate abnormally for various reasons. Job termination can happen from any state. An abnormally terminated job goes into `EXIT` state. The situations where a job terminates abnormally include:

- The job is cancelled by the user while pending, or after being started.
- The job is not able to be dispatched before it reaches its termination deadline, and thus is aborted by LSF Batch.
- The job fails to start successfully. For example, the wrong executable is specified by the user when the job is submitted.
- The job exits with a non-zero exit status.

1 LSF Batch Concepts

Jobs can also be suspended at any time. A job can be suspended by its owner, by the LSF administrator, by the root user (superuser), or by the LSF Batch system. There are three different states for suspended jobs:

PSUSP

suspended by its owner or the LSF administrator while in `PEND` state

USUSP

suspended by its owner or the LSF administrator after being dispatched

SSUSP

suspended by the LSF Batch system after being dispatched

After a job has been dispatched and started on a host, it can be suspended by LSF Batch. If the load on the execution host or hosts becomes too high, batch jobs could be interfering among themselves or could be interfering with interactive jobs. In either case, some jobs should be suspended to maximize host performance or to guarantee interactive response time.

LSF Batch suspends jobs according to the priority of the job's queue. When a host is busy, LSF Batch suspends lower priority jobs first unless the scheduling policy associated with the job dictates otherwise. Jobs are also suspended by the system if the job queue has a run window and the current time goes outside the run window.

The `bjobs -s` command displays the reason why a job was suspended.

A system-suspended job can later be resumed by LSF Batch if the load condition on the execution host(s) falls low enough or when the closed run window of the queue opens again.

Eligible Hosts

Each time LSF Batch attempts to dispatch a job, it checks to see which hosts are eligible to run the job. A number of conditions determine whether a host is eligible:

- Host dispatch windows
- Resource requirements of the job
- Resource requirements of the queue

- Host list of the queue
- Host load levels
- Job slot limits of the host.

A host is only eligible to run a job if all the conditions are met. If a batch job is queued and there is an eligible host for that job, the batch job is started on that host. If more than one host is eligible, the job is started on the best host based on both the job and the queue resource requirements.

Dispatch Windows

Each queue can be configured with a list of time periods, called *dispatch windows*, during which jobs in the queue can be dispatched. Jobs submitted to a queue are dispatched only when a queue dispatch window is open. Jobs can be submitted to a queue at any time; if the queue dispatch windows are closed, the jobs remain pending in the queue until a dispatch window opens. If no queue dispatch window is configured, the default is always open. Queue dispatch windows are displayed by the `bqueues -l` command.

Each host can also have dispatch windows. A host is not eligible to accept jobs when its dispatch windows are closed. Each batch job is dispatched from a specific queue, so a host is eligible to run a batch job if it is eligible for jobs from the queue, its dispatch windows are open, and it has the LSF resources required by the job. If no host dispatch window is configured, the default is always open. Host dispatch windows are displayed by the `bhosts -l` command.

Dispatch windows only control dispatching. Once a job has been dispatched to a host, it is unaffected by the status of dispatch windows.

Run Windows

Each queue can be configured with a list of time periods, called *run windows*, during which jobs from the queue can run. Jobs submitted to a queue only run when a queue run window is open. Jobs can be submitted to a queue at any time; if the queue run windows are closed, the jobs remain pending in the queue until a queue run window opens. When all of a queue's run windows close, any jobs dispatched from the queue are suspended until the queue's next run window opens. If no queue run window is

1 LSF Batch Concepts

configured, the default is always open. Queue run windows are displayed by the `bqueues -l` command.

Run windows also affect dispatching. No jobs are dispatched from a queue while its run windows are closed.

Note

Hosts only have dispatch windows, not run windows.

Resource Requirements

Each job can specify resource requirements. The resource requirements restrict which hosts the job can run on. For example, if your cluster contains three hosts with the `spice` resource and you give the argument “`-R spice`” to the `bsub` command, your job can only run on one of those three hosts. The `lshosts` command displays the resources available on each host. Each job can also specify an explicit list of eligible hosts, using the `-m` option to `bsub`. The `bjobs -l` command displays this list for each job.

Each queue can define resource requirements that will be applied to all the jobs in the queue. The queue-level resource requirements can also serve as job scheduling conditions shared by all jobs in the queue.

Host Lists

Each queue can be configured with a list of eligible hosts. For example, a queue for running programs on shared memory multiprocessors can be configured so that only the multiprocessor hosts are eligible. The eligible hosts for a queue are displayed by the `bqueues -l` command.

Host Load Levels

A host is available if the values of the load indices (such as `r1m`, `pg`, `mem`) of the host are within the configured *scheduling thresholds*. There are two sets of scheduling thresholds: host and queue. If any load index on the host exceeds the corresponding host threshold or queue threshold, the host is not eligible to run any job. The `bhosts -l` command displays the host thresholds. The `bqueues -l` command displays the queue thresholds.

Resource requirements at the queue level can also be used to specify scheduling conditions (for example, `r1m<0.4 && pg<3`).

Order of Job Dispatching

Each LSF Batch queue has a priority number. LSF Batch tries to start jobs from the highest priority queue first. Within each queue, by default jobs are dispatched in First-Come, First-Served (FCFS) order. If a fairshare scheduling policy has been specified for the queue or if host partitions have been configured, jobs are dispatched in accordance with these policies. (See *'Fairshare in Queues'* on page 31 and *'Fairshare in Host Partitions'* on page 31.)

The `bjobs` command shows the order in which jobs in a queue will actually be dispatched for the FCFS policy. This order can be changed by the `btop` and `bbot` commands (see *'Moving Jobs — bswitch, btop, and bbot'* on page 96).

Jobs can be dispatched out of turn if pre-execution conditions are not met, specific hosts or resources are busy or unavailable, or a user has reached the user job slot limit. (See *'Host Load Levels'* on page 24, *'User Job Slot Limits'* on page 27, and *'Queue-Level Pre-/Post-Execution Commands'* on page 224.)

Jobs are dispatched at 60 second intervals (the interval is configured by the `MBD_SLEEP_TIME` parameter in the `lsb.params` file). In each dispatching turn, LSF Batch tries to start as many jobs as possible.

To prevent overloading any host, LSF Batch waits for a configured number of dispatching intervals before sending another job to the same host. The waiting time is configured by the `JOB_ACCEPT_INTERVAL` parameter in the `lsb.params` file; the default is one dispatch interval. If `JOB_ACCEPT_INTERVAL` is set to zero, more than one job can be started on a host in the same dispatch turn.

The algorithm for starting jobs is:

- for each queue, from highest to lowest priority
- for each job in the queue, from first to last
- if any host is eligible to run this job, start the job on the best eligible host, and mark that host ineligible to run any other job until `JOB_ACCEPT_INTERVAL` dispatch turns have passed

1 LSF Batch Concepts

A higher priority or earlier batch job is only bypassed if no hosts are available that meet the requirements of the job. If a host is available but is not eligible to run a particular job, LSF Batch looks for a later job to start on that host. LSF Batch starts the first job found for which that host is eligible.

Job Slot Limits

Job slot is the basic unit of processor allocation in LSF Batch. A sequential job uses one job slot whereas a parallel job that has N components (tasks) uses N job slots, which can span multiple hosts. A job slot can be used by a maximum of one job. A *job slot limit* restricts the number of job slots that can be used at any one time. Each LSF Batch host, queue, and user can have a job slot limit. The table below gives the combinations for which job slot limits can be configured, along with the parameter used to configure the corresponding limit.

Table 1. Job Slot Limits

	User (in <code>lsb.users</code>)	Host (in <code>lsb.hosts</code>)	Queue (in <code>lsb.queues</code>)
Total	MAX_JOBS	MXJ	QJOB_LIMIT
Per user		JL/U	UJOB_LIMIT
Per processor	JL/P		PJOB_LIMIT
Per host			HJOB_LIMIT

Job slot limits are used by queues when deciding whether a particular job belonging to a particular user should be started on a specific host. Depending on whether or not preemptive scheduling policy has been configured for individual queues, each queue can have a different method of counting jobs toward job slot limits. The following points describe how jobs use job slots from a queue's point of view:

- If preemptive scheduling policy is not defined for the queue, slots taken by jobs that are started from any queues but have not yet finished are counted toward the respective job slot limits defined in the *User* and *Host* columns of *Table 1*. This includes the slots used by both running and suspended jobs (jobs in the `RUN`, `USUSP` and `SSUSP` states).

-
- If preemptive scheduling policy is defined, only the slots that are taken by jobs that are running and cannot be preempted by the current queue are counted toward the corresponding job slot limits defined in the `User` and `Host` columns of *Table 1*. This also includes running jobs from the current queue.

This means that slots taken by suspended jobs are not counted toward the job slot limits in `User` and `Host` columns.

- No matter what the queue policy is, slots taken by jobs that have been started from the current queue but have not yet finished are counted toward the job slot limits defined in the `Queue` column of *Table 1*.
- No matter what the queue policy is, slots that are reserved by some jobs on some hosts are counted toward the respective job slot limits defined in the `User`, `Host`, and `Queue` columns of *Table 1*. This means some pending jobs could occupy job slots.

The resulting counters are then used by this queue against various job slot limits during the scheduling of new jobs. Queues that can preempt others are more aggressive in scheduling jobs to hosts because a host appearing as full by a non-urgent queue would appear as not full from an urgent queue's point of view. See 'Preemptive Scheduling' on page 32 for the concept of preemptive scheduling.

Note

Although high priority preemptive queues neglect running jobs from low priority preemptable queues in checking job slot limits, LSF Batch will make sure that the total number of running jobs from a queue, a user, or on a host will not exceed the configured job slot limits in `lsb.queues`, `lsb.users`, and `lsb.hosts`. This is done by preempting (usually suspending) running jobs that can be preempted should the execution of a preemptive job cause the violation of the configured job slot limits.

User Job Slot Limits

Jobs are normally queued on a first-come, first-served (FCFS) basis. It is possible for some users to abuse the system by submitting a large number of jobs; jobs from other users must wait in the queue until these jobs complete. One way to prevent this is to use user job slot limits.

1 LSF Batch Concepts

User job slot limits control the number of job slots that can be used at once by a specific user or group of users. The definition of a job slot usage is dependent on the queue's policy, as described in *'Job Slot Limits' on page 26*.

A user can submit an unlimited number of jobs to LSF Batch system, but the system will only schedule this user's jobs up to his/her job slot limits. The system will not schedule further jobs for the user until some of the scheduled jobs free up the used job slots. User job slot limits come in different forms.

Each user or group of users can be assigned a system-wide job slot limit using the `MAX_JOBS` parameter in the `lsb.users` file.

Each user and user group can also be assigned a per-processor job slot limit using the `JL/P` parameter in the `lsb.users` file. For hosts that can run more than one LSF Batch job per processor, this prevents a user or group from using all the available job slots on the host.

User job slot limits are configured in the `User` section of the `lsb.users` file. See *'The lsb.users File' on page 198*.

Host Job Slot Limits

It is frequently useful to limit the maximum number of jobs that can be run on a host to prevent a host from being over-loaded with too many jobs and to maximize the throughput of a machine. Each host can be restricted to run a limited number of jobs at one time using the `MXJ` parameter in the `Host` section of the `lsb.hosts` file.

Each host can also restrict the number of jobs from each user allowed to run on the host, using the `JL/U` parameter in the `lsb.hosts` file. This limit is similar to the `JL/P` parameter in the `lsb.users` file. The `JL/U` parameter is configured for a particular host, and applies to all users on that host. The `JL/P` parameter is configured for a particular user, and applies to all hosts.

When a queue finds a host reaching one of its job slot limits, it will not start more jobs to this host until one or more job slots on the host are freed. The definition of job slot usage is described in *'Job Slot Limits' on page 26*.

For preemptive queues, if lower priority jobs are running on a host that has reached one of its job slot limits, LSF Batch will suspend one of these jobs to enable dispatch or resumption of a higher priority job.

Host job slot limits are configured in the `Host` section of the `lsb.hosts` file, which is described in *'The lsb.hosts File' on page 202*.

Queue Job Slot Limits

The `QJOB_LIMIT` parameter in the `lsb.queues` file controls the number of job slots a queue can use at any time. This parameter can be used to prevent a single queue from using all the processing resources in the cluster. For example, a high priority queue could have a `QJOB_LIMIT` set so that a few hosts remain available to run lower priority jobs.

Each queue can have a limit on the number of job slots a single user is allowed to use in that queue at one time. This limit prevents a single user from filling a queue with jobs and delaying other users' jobs. For example, each user could be limited to use one job slot at a time in a high priority queue to discourage overuse of the high priority queue.

The per-user job slot limit of a queue is configured with the `UJOB_LIMIT` parameter in the `lsb.queues` file.

Each queue can also have a limit on the number of jobs dispatched from the queue to a single processor, configured using the `PJOB_LIMIT` parameter in the `lsb.queues` file. This limit restricts the number of jobs a particular queue sends to any one host, while still allowing jobs from other queues to be dispatched to that host.

The `PJOB_LIMIT` parameter applies to each processor on a host. This allows the same limit to apply for both uniprocessor and multiprocessor hosts, without leaving multiprocessors underused.

A queue can limit the number of job slots available to jobs that are sent to the same host regardless of the number of processors the host has. This is set using the `HJOB_LIMIT` parameter in the `lsb.queues` file. If all of the job slots of a host have been taken or reserved by the jobs in this queue, no more jobs in this queue can be started on that host until some of the slots are released.

A queue's job slot limit per host does not prevent jobs from other queues from being dispatched to that host. For example, a low priority queue could be restricted to starting one job per processor. Higher priority queues would still be allowed to start other jobs on that host. By setting a low suspending threshold on the low priority queue, the low priority job can be forced to suspend when the high priority job starts.

1 LSF Batch Concepts

Queue job slot limits are configured in the `Queue` sections of the `lsb.queues` file, which is described in *'The lsb.queues File'* on page 208.

Resource Limits and Resource Usage

Jobs submitted through the LSF Batch system will have the resources they use monitored while they are running. This information is used to enforce job-level resource limits as well as to improve the fairshare scheduling to consider the current CPU time used by a job.

Resource limits supported by LSF Batch are described in *'Resource Limits'* on page 217.

Job-level resource usage is collected through a special process called PIM (Process Information Manager). PIM is managed internally by LSF. The information collected by PIM includes:

- Total CPU time consumed by all processes in the job
- Total resident memory usage in kilobytes of all currently running processes in a job
- Total virtual memory usage in kilobytes of all currently running processes in a job
- Currently active process group ID in a job
- Currently active processes in a job.

The `-l` option of the `bjobs` command displays the current resource usage of the job. The usage information is sampled by PIM every 30 seconds and collected by the `sbatchd` at a maximum frequency of every `SBD_SLEEP_TIME` (configured in the `lsb.params` file) and sent to the `mbatchd`. 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.

Scheduling Policies

Fairshare in Queues

Fairshare scheduling is an alternative to the default first-come, first-served scheduling. Fairshare scheduling divides the processing power of the LSF cluster among users and groups to provide fair access to resources for all the jobs in a queue. LSF allows fairshare policies to be defined at the queue level so that different queues can have different sharing policies. The fairshare policy of a queue applies to all hosts used by the queue.

Fairshare scheduling at the level of queues and host partitions (see below) are mutually exclusive.

For more information about how fairshare scheduling works and how to configure a fairshare queue, see *'Controlling Fairshare'* on page 113 and *'Queue Level Fairshare'* on page 221.

Fairshare in Host Partitions

Host partition provides fairshare policy at the host level. Unlike queue-level fairshare as described above, a host partition provides fairshare of resources on a group of hosts, and it applies to all queues that use hosts in the host partition.

Fairshare scheduling at the level of queues and host partitions are mutually exclusive.

For more information about how fairshare works and how it can be used to create specific scheduling policies, see *'Controlling Fairshare'* on page 113 and *'Host Partitions'* on page 206.

Hierarchical Fairshare

Hierarchical fairshare allows resources to be allocated to users in a hierarchical manner (for both queues and host partitions). Groups of users can collectively be allocated a share, and that share can be further subdivided and given to subgroups, resulting in a share tree. For a discussion of the terminology associated with hierarchical fairsharing, see *'Hierarchical Fairshare'* on page 60 in the *LSF Batch User's Guide*.

Preemptive Scheduling

Preemptive scheduling allows the LSF administrator to configure job queues such that a high priority job can preempt a low priority running job by suspending the low priority job. This is useful to ensure that long-running low priority jobs do not hold resources while high priority jobs are waiting for a job slot or job slots.

For more information about how preemptive scheduling works and how to configure a preemptive or preemptable queue, see '*Preemptive Scheduling*' on page 222.

Exclusive Scheduling

Exclusive scheduling makes it possible to run exclusive jobs on a host. A job only runs exclusively if it is submitted to an exclusive queue, and the job is submitted with the `bsub -x` option. An exclusive job runs by itself on a host — it is dispatched only to a host with no other batch jobs running, and LSF does not send any other jobs to the host until the exclusive job completes.

For more information about how exclusive scheduling works and how to configure an exclusive queue, see '*Exclusive Queue*' on page 223.

Processor Reservation and Backfilling

Processor Reservation and Backfilling ensure that large parallel jobs are able to run without underutilizing resources.

There might be delays in the execution of parallel jobs when they are competing with sequential jobs for resources. This is because as job slots become available, they are used in smaller numbers by sequential jobs. This results in the larger number of job slots required by a parallel job never becoming available at any given instant. *Processor reservation* allows job slots to be reserved for a parallel job until enough are available to start the job. When a job slot is reserved for a job, it is unavailable to other jobs.

However, there are situations where the system can determine that the job reserving the processors cannot start before a certain time. *Backfilling* is the execution of a job that is short enough to fit into the time slot during which the processors are reserved, allowing more efficient use of available resources. Short jobs are said to backfill processors reserved for large jobs. Backfilling requires that users specify how long each job will run so that LSF Batch can estimate when it will start and complete.

Suspending Jobs

Jobs running under LSF Batch can be suspended based on the load conditions on the execution host(s). Each host and each queue can be configured with a set of suspending conditions. If the load conditions on an execution host exceed either the corresponding host or queue suspending conditions, one or more jobs running on that host will be suspended to reduce the load until it falls below the suspending conditions.

LSF Batch provides different alternatives for configuring suspending conditions. Suspending conditions are configured at the host-level as *suspending thresholds*, whereas suspending conditions are configured at the queue-level as either *suspending thresholds*, or by using the `STOP_COND` parameter in the `lsb.queues` file, or both. See *'Host Section' on page 202*, *'Flexible Expressions for Queue Scheduling' on page 213*, and *'Load Thresholds' on page 216* for details about configuration options for suspending conditions at host and queue levels.

The suspending conditions are displayed by the `bhosts -l` and `bqueues -l` commands. The thresholds that apply to a particular job are the more restrictive of the host and queue thresholds, and are displayed by the `bjobs -l` command.

LSF Batch checks the host load levels periodically. The period is defined by the `SBD_SLEEP_TIME` parameter in the `lsb.params` file. There is a time delay between when LSF Batch suspends a job and when the changes to host load are seen by the LIM. To allow time for load changes to take effect, LSF Batch suspends at most one job per `SBD_SLEEP_TIME` on each host.

Each turn, LSF Batch gets the load levels for that host. Then for each job running on the host, LSF Batch compares the load levels against the host suspending conditions and the queue suspending conditions for the queue that job was submitted to. If any suspending condition at either the corresponding host or queue level is satisfied as a result of increased load, the job is suspended.

Jobs from the lowest priority queue are checked first. If two jobs are running on a host and the host is too busy, the lower priority job is suspended and the higher priority job is allowed to continue. If the load levels are still too high on the next turn, the higher priority job is also suspended.

Note that a job is only suspended if the load levels are too high for that particular job's suspending conditions. It is possible, though not desirable, to configure LSF Batch so that a low priority queue has very loose suspending conditions. In this case a job from

1 LSF Batch Concepts

a higher priority queue might be suspended first, because the load levels are not yet too high for the low priority queue.

In addition to excessive load, jobs from a queue are also suspended if all the run windows of the queue close. The jobs are resumed when the next run window of the queue opens. For example, a night queue might be configured to run jobs between 7 p.m. and 8 a.m. If a job is still running in the morning, it is suspended, and is resumed around 7 p.m. of that day.

In contrast, when the dispatch windows of a queue or host close, jobs from that queue or running on that host continue running. The dispatch windows control job dispatching only.

Migration

Each host and queue can be configured so that suspended checkpointable or rerunnable jobs are automatically migrated to another host. See *'Checkpointing and Migration' on page 37*.

Special Cases

Three special cases affect job suspension. Two cases are intended to prevent batch jobs from suspending themselves because of their own load, and one case is intended to allow an urgent job to run to completion despite unfavourable load conditions. If a batch job is suspended because of its own load, the load drops as soon as the job is suspended. When the load goes back within the thresholds, the job is resumed until it causes itself to be suspended again.

First, when only one batch job is running on a host, the batch job is not suspended for any reason except that the host is not idle (the `it` interactive idle time load index is less than one minute). This means that once a job is started on a host, at least one job continues to run unless there is an interactive user on the host. Once the job is suspended, it is not resumed until all the scheduling conditions are met, so it should not interfere with the interactive user.

Second, this case applies only for the `pg` (paging rate) load index. A large batch job often causes a high paging rate. Interactive response is strongly affected by paging, so it is desirable to suspend batch jobs that cause paging when the host has interactive users. The `PG_SUSP_IT` parameter in the `lsb.params` file controls this behaviour. If

the host has been idle for more than `PG_SUSP_IT` minutes, the `pg` load index is not checked against the suspending threshold.

Finally, conditions such as thresholds for hosts and queues and windows for queues can cause a running job to be suspended. However, certain urgent jobs can be run until completion without being suspended by these conditions. By using the `-f` option of the command `brun(1)`, an LSF administrator can force a job to run and the job will not be suspended by LSF Batch due to load conditions. See *'Forcing Job Execution — `brun -f`' on page 98* for details.

Resuming Suspended Jobs

Jobs are suspended to prevent overloading hosts, to prevent batch jobs from interfering with interactive use, or to allow a more urgent job to run. When the host is no longer overloaded, suspended jobs should continue running.

LSF Batch uses queue level and host level scheduling thresholds as described in *'Host Load Levels' on page 24* to decide whether a suspended job should be resumed. At the queue level, LSF Batch also uses the `RESUME_COND` parameter in the `lsb.queues` file. Unlike suspending conditions, all the resuming conditions must be satisfied for a job to resume.

If there are any suspended jobs on a host, LSF Batch checks the load levels in each turn. If the load levels are within the scheduling thresholds of both queue level and host levels, and the resume condition `RESUME_COND` configured at the queue level is satisfied, the job is resumed.

Jobs from higher priority queues are checked first. Only one job is resumed in each turn to prevent overloading the host again.

The scheduling thresholds that control when a job is resumed are displayed by the `bjobs -l` command.

User Suspended Jobs

A job can also be suspended by its owner or the LSF administrator with the `bstop` command. These jobs are considered user-suspended (displayed by `bjobs` as `USUSP`).

1 LSF Batch Concepts

When the user restarts the job with the `brresume` command, the job is not started immediately to prevent overloading. Instead, the job is changed from `USUSP` to `SSUSP` (suspended by the system). The `SSUSP` job is resumed when the host load levels are within the scheduling thresholds for that job, exactly as for jobs suspended due to high load.

If a user suspends a high priority job from a non-preemptive queue, the load might become low enough for LSF Batch to start a lower priority job in its place. The load created by the low priority job can prevent the high priority job from resuming. This can be avoided by configuring preemptive queues (see *'Preemptive Scheduling'* on page 32).

Interactive Batch Job Support

A batch job can be submitted in interactive mode such that all input and output are through the terminal from which the `bsub` command is issued. The principal advantage of running an interactive job through the LSF Batch system is that it takes advantage of the batch scheduling policy and host selection features for resource intensive jobs. Additionally, all statistics related to the job are recorded in the `lsb.acct` file to allow a common accounting system for both interactive and non-interactive jobs.

You can configure a queue to be interactive only, batch only, or both interactive and batch (see *'General Parameters'* on page 208 for details on configuring an interactive queue). An interactive batch job is submitted by specifying the `-I` option to the `bsub` command. An interactive batch job is scheduled using the same policy as all other jobs in a queue. This means an interactive job can wait for a long time before it gets dispatched. If fast response time is required, interactive jobs should be submitted to high priority queues with loose scheduling constraints.

Pre- and Post-execution Commands

Each batch job can be submitted with optional pre- and post-execution commands.

If a pre-execution command is specified, the job is held in the queue until the specified pre-execution command returns a successful exit status (zero). While the job is pending, other jobs can proceed ahead of the waiting job.

If a post-execution command is specified, then the command is run after the job is finished.

Pre- and post-execution commands are arbitrary command lines.

Pre-execution commands can be used to support job starting decisions which cannot be configured directly in LSF Batch.

Post-execution commands are typically used to clean up some state left by the pre-execution and the job execution.

LSF Batch supports both job level and queue level pre-execution. Post-execution is only supported at the queue level.

See *'Queue-Level Pre-/Post-Execution Commands'* on page 224 for more information about queue level pre- and post-execution commands, and the chapter *'Submitting Batch Jobs'* on page 89 in the *LSF Batch User's Guide* for more information about the job-level pre-execution commands.

Checkpointing and Migration

Batch jobs can be checkpointed and migrated to other hosts of the same type. LSF supports three forms of checkpointing:

- Kernel-level checkpointing—The operating system kernel supports checkpointing without application changes.
- User-level checkpointing—The application is linked with a special library to support checkpoint and restart, but no source changes are required to the application.
- Application-level checkpointing—The application has source changes that allow it to interact with the supplied checkpointing interface commands.

1 LSF Batch Concepts

Kernel level checkpointing is currently supported on ConvexOS, Cray Unicos, IRIX 6.4 and later, and HP Exemplar systems. LSF Batch provides a uniform checkpointing protocol to support checkpointing at all levels for all platforms by providing the commands `echkpnt` and `erestart` (located in the `LSF_SERVERDIR` directory, which is defined in the `lsf.conf` file—otherwise, the location is defined by the `LSF_ECHKPNTDIR` environment variable).

Details of checkpointing are described in the chapter ‘*Checkpointing and Migration*’ on page 165 in the *LSF Batch User’s Guide*.

Job Migration

Check-pointable jobs and re-runnable jobs can be migrated to another host for execution if the current host is too busy or the host is going to be shut down. A rerunnable job is a job that is submitted with the `bsub -r` option and can be correctly rerun from the beginning. Jobs can be moved from one host to another, as long as both hosts are binary compatible and run the same version of the operating system.

The job’s owner or the LSF administrator can use the `bmig` command to migrate jobs. If the job is checkpointable, the `bmig` command first checkpoints it. Then LSF kills the running or suspended job, and restarts or reruns the job on another host if one is available. If LSF is unable to rerun or restart the job due to a system or network reason, the job reverts to `PEND` status and is queued with a higher priority than any submitted job, so it is rerun or restarted before other queued jobs are dispatched.

Job Control Actions

LSF Batch needs to control jobs dispatched to a host to enforce scheduling policies, or in response to user requests. The principal actions that the system performs on a job include suspending, resuming, and terminating it. The actions are carried out by sending the signal `SIGSTOP` for suspending a job, `SIGCONT` for resuming a job, and `SIGKILL` for terminating a job. On NT, equivalent functions have been implemented to perform the same tasks.

Occasionally, you might want to override the default actions. For example, instead of suspending a job, you might want to kill or checkpoint it. The default job control

actions can be overridden by defining the `JOB_CONTROLS` parameter in your queue configuration. Each queue can have its separate job control actions. See *'Job Starter'* on page 227 for more details.

Resource Reservation

When a job is dispatched, the system assumes that the resources that the job consumes will be reflected in the load information. However, many jobs do not consume the resources they require when they first start. Instead, they will typically use the resources over a period of time. For example, a job requiring 100 megabytes of swap is dispatched to a host having 150 megabytes of available swap. The job starts off initially allocating 5 megabytes and gradually increases the amount consumed to 100 megabytes over a period of 30 minutes. During this period, another job requiring more than 50 megabytes of swap should not be started on the same host to avoid over-committing the resource.

Resources can be reserved to prevent over commitment by LSF Batch. Resource reservation requirements can be specified as part of the resource requirements when submitting a job, or can be configured into the queue level resource requirements. See *'Queue Level Resource Reservation'* on page 214 for details about configuring resource reservation at the queue level. For descriptions about specifying resource reservation with job submission, see *'Resource Reservation'* on page 91 of the *LSF Batch User's Guide*.

Processor Reservation

When parallel jobs have to compete with sequential jobs for resources, a common situation is that parallel jobs will find it very difficult to get enough processors to run. This is because a parallel job needs to collect more than one job slot before it can be dispatched. There might not be enough job slots at any one instant to satisfy a large parallel job, but there might be enough to allow a sequential job to be started. This might cause parallel jobs to wait forever, if there are enough sequential jobs.

Processor reservation of the LSF Batch solves this problem by reserving processors for parallel jobs. When a parallel job cannot be dispatched because there are not enough

1 LSF Batch Concepts

job slots to satisfy its minimum processor requirements, the currently available slots will be reserved for the job. These reserved job slots are accumulated until there are enough available to start the job. When a slot is reserved for a job it is unavailable to any other job. To avoid deadlock situations, the period of reservation needs to be configured so that the parallel job will give up the reserved job slots if it still cannot run after the reservation period. See '*Processor Reservation for Parallel Jobs*' on page 211 for details about the reservation period configuration.

In addition, there are situations where the system can determine that the job reserving the processors cannot start before a certain time. In this situation it makes sense to run a job that is short enough to fit into the time slot during which the processors are reserved; this is referred to as *backfilling*. Short jobs are said to backfill processors reserved for large jobs. Backfilling requires that users specify how long each job will run so that LSF Batch can estimate when a job will start and complete. Backfilling, together with processor reservation, allows large parallel jobs to run while not underutilizing resources.

Remote File Access

When LSF Batch runs a job, it attempts to run the job in the directory where the `bsub` command was invoked. If the execution directory is under the user's home directory, `sbatchd` looks for the path relative to the user's home directory. This handles some common configurations, such as cross-mounting users' home directories with the `/net automount` option.

If the directory is not available on the execution host, the job is run in `/tmp`. Any files created by the batch job, including the standard output and error files created by the `-o` and `-e` options to the `bsub` command, are left on the execution host.

LSF Batch provides support for moving user data from the submission host to the execution host before executing a batch job, and from the execution host back to the submitting host after the job completes. The file operations are specified with the `-f` option to `bsub`.

The LSF Batch remote file access mechanism uses `lsrscp(1)` to process the file transfer. `lsrscp` first tries to connect to the RES daemon on the submission host to handle the file transfer.

UNIX

If `lsrscp` cannot contact the 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)` and `rsh(1)` manual pages for more information on using `rcp`.

A site can replace `lsrscp` with its own file transfer mechanism as long as it supports the same syntax as `lsrscp(1)`. This might be done to take advantage of a faster interconnection network, or to overcome limitations with the existing `lsrscp`. `sbatchd` looks for the `lsrscp` executable in the `LSF_BINDIR` directory as specified in the `lsf.conf` file.

For a complete description of the LSF remote file access facilities, see the `bsub(1)` manual page and ‘*Other bsub Options*’ on page 112 of the *LSF Batch User’s Guide*.

Job Requeue

A networked computing environment is vulnerable to any failure or temporary conditions in network services or processor resources. For example, you might get NFS stale handle errors, disk full errors, process table full errors, or network connectivity problems. In addition, your application can also be subject to external conditions such as a software license problem, or an occasional failure due to a bug in your application.

Such errors are temporary and probably will happen at one time but not another, or on one host but not another. You might be upset to learn all your jobs exited due to temporary errors and you did not know about it until 12 hours later.

LSF Batch provides a way to automatically recover from temporary errors. You can configure certain exit values such that in case a job exits with one of the values, the job will be automatically requeued as if it had not yet been dispatched. This job will then be retried later. It is also possible for you to configure your queue such that a requeued job will not be scheduled to hosts on which the job had previously failed to run. See ‘*Automatic Job Requeue*’ on page 231 and ‘*Exclusive Job Requeue*’ on page 232 for details.

External Submission and Execution Executables

Administrators can write external submission and execution time executables to perform additional site-specific actions on jobs. These executables are called `esub` and `eexec` and they must reside in `LSF_SERVERDIR` (defined in the `lsf.conf` file). When a job is submitted, `esub` is executed if it is found in `LSF_SERVERDIR`. On the execution host, `eexec` is run at job start-up and completion time, and when checkpointing is initiated. The environment variable `LS_EXEC_T` is set to `START`, `END`, and `CHKPNT`, respectively, to indicate when `eexec` is invoked. If `esub` needs to pass some data to `eexec`, `esub` can write the data to its standard output; `eexec` can read the data from its standard input. Thus, LSF is effectively implementing the pipe in `esub | eexec`.

`eexec` is executed as the user after the job's environment variables have been set. If you need to run `eexec` as a different user, such as `root`, you must properly define `LSF_EEXEC_USER` in the file `/etc/lsf.sudoers` (see *'The lsf.sudoers File' on page 189* for details). The parent job process waits for `eexec` to complete before proceeding; thus, `eexec` is expected to complete. The environment variable `LS_JOBPID` stores the process ID of the process that invoked `eexec`. If `eexec` is intended to monitor the execution of the job, `eexec` must fork a child and then have the parent `eexec` process exit. The `eexec` child should periodically test that the job process is still alive using the `LS_JOBPID` variable.

Under LSF Batch, `esub` can also be used to validate the submission parameters and reject the job. The submission parameters are saved in a file before `esub` is invoked (see *'Validating Job Submissions' on page 91* for details). `esub` can read the file and exit with a special exit code to cause the job submission or modification to be aborted. A typical use of this feature of external submission is to validate users for project membership. If a submission parameter for the project is not valid, or the user is not permitted to charge his job to that project, the job can be rejected.

Interactive remote execution also runs these external executables if they are found in `LSF_SERVERDIR`. For example, `lsrun` invokes `esub`, and the `RES` runs `eexec` before starting the task. `esub` is invoked at the time of the `ls_connect(3)` call, and the `RES` invokes `eexec` each time a remote task is executed. Unlike LSF Batch, the `RES` runs `eexec` only at task startup time.

The `esub/eexec` facility is used for processing DCE credentials and AFS tokens (see *'Installation on AFS'* and *'Installation on DCE/DFS'* in the *LSF Installation Guide*).

External Load Indices and ELIM

LSF Base contains a LIM that collects 11 built-in load indices that reflect the load situations of CPU, memory, disk space, I/O, and interactive activities on individual hosts.

While built-in load indices might be sufficient for most user sites, there are always user sites with special workload or resource dependencies that require additional load indices. LSF's open system architecture allows users to write an External Load Information Manager (ELIM) that gathers the additional load and shared resource information a site needs. This ELIM can then be plugged into LIM so that they appear as a single LIM to the users. External load indices are used in exactly the same way as built-in load indices in various scheduling or host selection policies.

An ELIM can be as simple as a small script, or as complicated as a sophisticated C program. A well defined protocol allows the ELIM to talk to LIM. See *'Changing LIM Configuration'* on page 55 for details about writing and configuring an ELIM.

External Group Membership Definition

User group or host group definitions can be maintained outside of LSF and imported into the LSF Batch configuration at initialization time. An executable `egroup` in the `LSF_SERVERDIR` directory is invoked to obtain the list of members for a given group. The group members, separated by spaces, should be written to the standard output stream of `egroup`. In the LSF configuration file, the special character `!` should be specified for the group member to indicate that `egroup` should be invoked. See *'External User Groups'* on page 200 and *'Host Groups'* on page 205 for details about writing an `egroup` program.

1 LSF Batch Concepts

2. Managing LSF Base

This chapter describes the operation, maintenance, and tuning of LSF Base cluster. Since LSF Base is essential to all LSF components, the correct operation of LSF Base is essential to other LSF products.

Managing Error Logs

Error logs contain important information about daemon operations. When you see any abnormal behavior related to any of the LSF daemons, you should check the relevant error logs to find out the cause of the problem.

LSF log files grow over time. These files should occasionally be cleared, either by hand or using automatic scripts.

LSF Daemon Error Log

All LSF log files are reopened each time a message is logged, so if you rename or remove a log file of an LSF daemon, the daemons will automatically create a new log file.

The LSF daemons log messages when they detect problems or unusual situations.

The daemons can be configured to put these messages into files.

UNIX On UNIX, the message can be sent to the system error logs using the `syslog` facility.

If `LSF_LOGDIR` is defined in the `lsf.conf` file, LSF daemons try to store their messages in files in that directory. Note that `LSF_LOGDIR` must be writable by `root`.

2 Managing LSF Base

The error log file names for the LSF Base system daemons, LIM and RES, are `lim.log.hostname`, `res.log.hostname`.

The error log file names for LSF Batch daemons are `sbatchd.log.hostname`, `mbatchd.log.hostname`, and `pim.log.hostname`.

UNIX

If `LSF_LOGDIR` is defined, but the daemons cannot write to files there, the error log files are created in `/tmp`.

On Unix, if `LSF_LOGDIR` is not defined, then errors are logged to `syslog` using the `LOG_DAEMON` facility. `syslog` messages are highly configurable, and the default configuration varies widely from system to system. Start by looking for the file `/etc/syslog.conf`, and read the manual pages for `syslog` and/or `syslogd`.

NT

If `LSF_LOGDIR` is defined, but the daemons cannot write to files there, the error log files are created in `C:\temp`.

LSF daemons log error messages in different levels so that you can choose to log all messages, or only log messages that are deemed critical. Message logging is controlled by the parameter `LSF_LOG_MASK` in the `lsf.conf` file. Possible values for this parameter can be any log priority symbol that is defined in `<syslog.h>`. The default value for `LSF_LOG_MASK` is `LOG_WARNING`.

If the error log is managed by `syslog`, it is probably already being automatically cleared.

If LSF daemons cannot find the `lsf.conf` file when they start, they will not find the definition of `LSF_LOGDIR`. In this case, error messages go to `syslog`. If you cannot find any error messages in the log files, they are likely in the `syslog`.

See *'Troubleshooting and Error Messages'* on page 239 for a discussion of common problems and error log messages.

FLEXlm Log

The FLEXlm license server daemons log messages about the state of the license servers, and when licenses are checked in or out. This log helps to resolve problems with the license servers and to track license use.

The FLEXlm log is configured by the `lsflicsetup` command as described in ‘Installing a New Permanent License’ in the *LSF Installation Guide*. This log file grows over time. You can remove or rename the existing FLEXlm log file at any time. The script `lsf_license` used to run the FLEXlm daemons creates a new log file when necessary.

Note

If you already have FLEXlm server running for other products and LSF licenses are added to the existing license file, then the log messages for FLEXlm should go to the same place as you previously set up for other products.

Controlling LIM and RES Daemons

The LSF cluster administrator can monitor the status of the hosts in a cluster, start and stop the LSF daemons, and reconfigure the cluster. Many operations are performed using the `lsadmin` command, which performs administrative operations on LSF Base daemons, LIM, and RES.

Checking Host Status

The `lshosts` and `lsload` commands report the current status and load levels of hosts in an LSF cluster. The `lsmmon` and `xlsmmon` commands provide a running display of the same information. The LSF administrator can find unavailable or overloaded hosts with these tools.

```
% lsload
HOST_NAME status r15s r1m r15m ut pg ls it tmp swp mem
hostD      ok      1.3 1.2 0.9 92% 0.0 2 20 5M 148M 88M
hostB      -ok     0.1 0.3 0.7 0% 0.0 1 67 45M 25M 34M
hostA      busy    8.0 *7.0 4.9 84% 4.6 6 17 1M 81M 27M
```

When the status of a host is preceded by a ‘-’, it means RES is not running on that host. In the above example, RES on `hostB` is down.

2 Managing LSF Base

Restarting LIM and RES

LIM and RES can be restarted to upgrade software or clear persistent errors. Jobs running on the host are not affected by restarting the daemons. The LIM and RES daemons are restarted using the `lsadmin` command:

```
% lsadmin
lsadmin>limrestart hostD
Checking configuration files ...
No errors found.
```

```
Restart LIM on <hostD> ..... done
lsadmin>resrestart hostD
Restart RES on <hostD> ..... done
lsadmin>quit
```

Note

You must login as LSF cluster administrator to run `lsadmin` command.

The `lsadmin` command can be applied to all available hosts by using the host name `all`; for example, `lsadmin limrestart all`. If a daemon is not responding to network connections `lsadmin` displays an error message with the host name. In this case you must kill and restart the daemon manually.

Remote Startup of LIM and RES

LSF administrators can start up any, or all, LSF daemons, on any, or all, LSF hosts, from any host in the LSF cluster. For this to work, file `lsf.sudoers` has to be set up properly to allow you to start up daemons as `root`. You should be able to run `rsh` across LSF hosts without having to enter a password. See ‘*The `lsf.sudoers` File*’ on page 189 for configuration details of `lsf.sudoers`.

The `limstartup` and `resstartup` options in `lsadmin` allow for the startup of the LIM and RES daemons respectively. Specifying a host name allows for starting up a daemon on a particular host. For example:

```
% lsadmin limstartup hostA
Starting up LIM on <hostA> ..... done
```

```
% lsadmin resstartup hostA
Starting up RES on <hostA> ..... done
```

The `lsadmin` command can be used to start up all available hosts by using the host name `all`; for example, `lsadmin limstartup all`. All LSF daemons, including LIM, RES, and `sbatchd`, can be started on all LSF hosts using the command `lsfstartup`.

Shutting down LIM and RES

All LSF daemons can be shut down at any time. If the LIM daemon on the current master host is shut down, another host automatically takes over as master. If the RES daemon is shut down while remote interactive tasks are running on the host, the running tasks continue but no new tasks are accepted. To shutdown LIM and RES, use `lsadmin` command:

```
% lsadmin
lsadmin>resshutdown hostD
Shut down RES on <hostD> ..... done
lsadmin>limshutdown hostD
Shut down LIM on <hostD> ..... done
lsadmin>quit
```

You can run `lsadmin reconfig` while the LSF system is in use; users might be unable to submit new jobs for a short time, but all current remote executions are unaffected.

Locking and Unlocking Hosts

A LIM can be locked to temporarily prevent any further jobs from being sent to the host. The lock can be set to last either for a specified period of time, or until the host is explicitly unlocked. Only the local host can be locked and unlocked.

2 Managing LSF Base

```
% lsadmin limlock
Host is locked
% lsload
HOST_NAME  status r15s  r1m  r15m  ut  pg  ls  it  tmp  swp  mem
hostD      ok      1.3   1.2  0.9   92% 0.0 2   20  5M   148M 28M
hostA      busy    8.0   *7.0 4.9   84% 0.6 0   17  *1M  31M   7M
hostC      lockU   0.8   1.0  1.1   73% 1.2 3   0   4M   44M  12M
% lsadmin limunlock
Host is unlocked
```

Only *root* and the LSF administrator can lock and unlock hosts.

Managing LSF Configuration

Overview of LSF Configuration Files

LSF configuration consists of several levels:

- `lsf.conf`—The primary LSF environment configuration file
- `lsf.shared` and `lsf.cluster.cluster`—Configuration files for the Load Information Manager
- `lsf.task` and `lsf.task.cluster`—The files containing task to default resource requirement string mappings
- `LSB_CONFDIR/cluster`—The directory containing configuration files for LSF Batch

The `lsf.conf` File

This is the generic LSF environment configuration file. This file defines general installation parameters so that all LSF executables can find the necessary information. This file is typically installed in the `LSF_CONFDIR` directory (the same directory as the LIM configuration files), and a symbolic link is made from a convenient directory as defined by the environment variable `LSF_ENVDIR`, or the default directory `/etc`. This file is created by the `lsfsetup` during installation. Note that many of the parameters

in this file are machine specific. The contents of this file are described in detail in *'The lsf.conf File'* on page 161.

LIM Configuration Files

LIM is the kernel of your cluster that provides the single system image to all applications. LIM reads the LIM configuration files and determines your cluster and the cluster master host.

LIM files include `lsf.shared` and `lsf.cluster.cluster`, where *cluster* is the name of your LSF cluster. These files define the host members, general host attributes, and resource definitions for your cluster. The individual functions of each of the files are described below.

`lsf.shared` defines the available resource names, host types, host models, cluster names, and external load indices that can be used by all clusters. This file is shared by all clusters.

`lsf.cluster.cluster` file is a per cluster configuration file. It contains two types of configuration information: cluster definition information and LIM policy information. Cluster definition information impacts all LSF applications, while LIM policy information impacts applications that rely on LIM's policy for job placement.

The cluster definition information defines cluster administrators, all the 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 defines the load sharing and job placement policies provided by LIM. More details about LIM policies are described in *'Tuning LIM Load Thresholds'* on page 69.

LIM configuration files are stored in directory `LSF_CONFDIR` as defined in `lsf.conf` file. Details of LIM configuration files are described in *'The lsf.shared File'* on page 173.

The `lsf.task` File

`lsf.task` is a system-wide task to 'default resource requirement string' mapping file. This file defines mappings between task names and their default resource requirements. LSF maintains a task list for each user in the system. The `lsf.task` file is useful for the cluster administrator to set task-to-resource requirement mapping at

2 Managing LSF Base

the system level. Individual users can customize their own list by using the `lsrtasks` command (See `lsrtasks(1)` man page for details on this command).

When you run a job with an LSF command such as `bsub` or `lrun`, the command consults your task list to find out the default resource requirement string of the job if they are not already specified explicitly. If a match is not found in your task list, the system will assume a default, which typically means run the job on a host that has the same host type as the local host.

There is also a per cluster file `lsf.task.cluster` that applies to the cluster only and overrides the system-wide definition. Individual users can have their own files to override the system-wide and cluster-wide files by using the `lsrtasks` command.

`lsf.task` and `lsf.task.cluster` files are installed in directory `LSF_CONFDIR` as defined in `lsf.conf` file.

LSF Batch Configuration Files

These files define LSF Batch specific configuration such as queues, batch server hosts, and batch user controls. These files are only read by `mbatchd`. The LSF Batch configuration relies on LIM configuration. LSF Batch daemons get the cluster configuration information from the LIM via the LSF API.

LSF Batch configuration files are stored in directory `LSB_CONFDIR/cluster`, where `LSB_CONFDIR` is defined in `lsf.conf`, and `cluster` is the name of your cluster. Details of LSF Batch configuration files are described in ‘*Managing LSF Batch*’ on page 79.

Configuration File Formats

All configuration files except `lsf.conf` use a section-based format. Each file contains a number of sections. Each section starts with a line beginning with the reserved word `Begin` followed by a section name, and ends with a line beginning with the reserved word `End` followed by the same section name. `Begin`, `End`, section names, and keywords are all case insensitive.

Sections can either be vertical or horizontal. A horizontal section contains a number of lines, each having the format: `keyword = value`, where `value` is one or more strings. For example:

```
Begin exampleSection
key1 = string1
key2 = string2 string3
key3 = string4
End exampleSection
```

```
Begin exampleSection
key1 = STRING1
key2 = STRING2 STRING3
End exampleSection
```

In many cases you can define more than one object of the same type by giving more than one horizontal section with the same section name.

A vertical section has a line of keywords as the first line. The lines following the first line are values assigned to the corresponding keywords. Values that contain more than one string must be bracketed with '(' and ')'. The above examples can also be expressed in one vertical section:

```
Begin exampleSection
key1      key2          key3
string1 (string2 string3) string4
STRING1 (STRING2 STRING3) -
End exampleSection
```

Each line in a vertical section is equivalent to a horizontal section with the same section name.

Some keys in certain sections are optional. For a horizontal section, an optional key does not appear in the section if its value is not defined. For a vertical section, an optional keyword must appear in the keyword line if any line in the section defines a value for that keyword. To specify the default value use '-' or '()' in the corresponding column, as shown for key3 in the example above.

Each line can have multiple columns, separated by either spaces or TAB characters. Lines can be extended by a '\' (back slash) at the end of a line. A '#' (pound sign) indicates the beginning of a comment; characters up to the end of the line are not interpreted. Blank lines are ignored.

2 Managing LSF Base

Example Configuration Files

Below are some examples of LIM configuration files. The detailed explanations of the variables are described in '*LSF Base Configuration Reference*' on page 161.

Example `lsf.shared` file

```
Begin Cluster
ClusterName                # This line is keyword(s)
test_cluster
End Cluster

Begin HostType
TYPENAME                   # This line is keyword(s)
hppa
SUNSOL
rs6000
alpha
NTX86
End HostType

Begin HostModel
MODELNAME                  CPUFACTOR          # This line is keyword(s)
HP735                      4.0
DEC3000                    5.0
ORIGIN2K                   8.0
PENTI1120                 3.0
End HostModel

Begin Resource
RESOURCENAME TYPE      INTERVAL INCREASING DESCRIPTION #This line is keyword(s)
hpux          Boolean ()          ()          (HP-UX operating system)
decunix       Boolean ()          ()          (Digital Unix)
solaris       Boolean ()          ()          (Sun Solaris operating system)
NT            Boolean ()          ()          (Windows NT operating system)
fserver       Boolean ()          ()          (File Server)
cserver       Boolean ()          ()          (Compute Server)
scratch       Numeric 30          N          (Shared scratch space on server)
verilog       Numeric 30          N          (Floating licenses for Verilog)
console       String 30          N          (User Logged in on console)
End Resource
```

Example `lsf.cluster.test_cluster` file:

```
Begin ClusterManager
Manager = lsf user7
End ClusterManager

Begin Host
HostName      Model      Type      server      swp      Resources
hostA         HP735     hppa      1           2       (fserver hpux)
hostD         ORIGIN2K  sgi       1           2       (cserver)
hostB         PENT200   NTX86     1           2       (NT)
End Host
```

In the above file, section `ClusterManager` takes horizontal format, while `Host` section takes vertical format.

Other LSF Batch configuration files are described in ‘*Example LSF Batch Configuration Files*’ on page 136.

Changing LIM Configuration

This section provides procedures for some common changes to the LIM configuration. There are three different ways for you to change LIM configuration:

- Use the `lsfsetup` program as described in various sections of the *LSF Installation Guide*
- Edit individual files using a text editor
- Use the `xlsadmin` tool (a graphical application).

The following discussions focus on changing configuration files using a text editor so that you can understand the concepts behind the configuration changes. See ‘*Managing an LSF Cluster Using xlsadmin*’ on page 99 for the use of `xlsadmin` in changing configuration files.

Note

If you run LSF Batch, you must restart `mbatchd` using the `badmin reconfig` command each time you change the LIM configuration, even if the LSF Batch configuration files do not change. This is necessary because the LSF Batch configuration depends on the LIM configuration.

2 Managing LSF Base

Adding a Host to a Cluster

- Step 1** If you are adding a host of a new host type, make sure you perform the steps described in *Installing an Additional Host Type* in the *LSF Installation Guide* first.
- Step 2** If you are adding a host of a type for which you have already installed LSF binaries, make sure that the LSF binaries, configuration files, and working directories are NFS-mounted on the new host. For each new host you add, follow the host setup procedure as described in *Adding an Additional Host to an Existing Cluster* in the *LSF Installation Guide*.
- Step 3** If you are adding a new host type to the cluster, modify the `HostType` section of the `lsf.shared` file to add the new host type. A host type can be any alphanumeric string up to 29 characters long.
- Step 4** If you are adding a new host model, modify the `HostModel` section of your `lsf.shared` file to add in the new model together with its CPU speed factor relative to other models.
- Step 5** For each host you add into the cluster, you should add a line to the `Host` section of the `lsf.cluster.cluster` file with host name, host type, and all other attributes defined, as shown in *Example Configuration Files* on page 54.
- The master LIM and `mbatchd` daemons run on the first available host in the `Host` section of your `lsf.cluster.cluster` file, so you should list reliable batch server hosts first. For more information see *Fault Tolerance* on page 5.
- If you are adding a client host, set the `SERVER` field for the host to 0 (zero).
- Step 6** Reconfigure your LSF cluster so that LIM knows that you have added a new host to the cluster. Follow instructions in *Reconfiguring an LSF Cluster* on page 62. If you are adding more than one host, perform this step after you have performed steps 1 to 6 for all added hosts.
- Step 7** If you are adding hosts as LSF Batch server hosts, add these hosts to the LSF Batch configuration by following steps described in *Restarting sbatchd* on page 85.

Step 8 Start the LSF daemons on the newly added host(s) by running `LSF_SERVERDIR/lsf_daemons start` and use `ps` to make sure that `res`, `lim` and `sbatchd` have started.

CAUTION!

The `lsf daemons start` command must be run as *root*. If you are creating a private cluster, do not attempt to use `lsf_daemons` to start your daemons, as this command will kill all running daemons on the system before starting new ones. Start them manually.

Removing Hosts From a Cluster

Step 1 If you are running LSF Batch, make sure you remove unwanted hosts from the LSF Batch first following steps described in ‘*Restarting sbatchd*’ on page 85.

Step 2 Edit your `lsf.cluster.cluster` file and remove the unwanted hosts from the `Host` section.

Step 3 Log in to any host in the cluster as the LSF administrator. Run:
`lsadmin resshutdown host1 host2 ...`
where `host1`, `host2`, ... are hosts you want to remove from your cluster.

Step 4 Follow instructions in ‘*Reconfiguring an LSF Cluster*’ on page 62 to reconfigure your LSF cluster. The LIMs on the removed hosts will quit upon reconfiguration.

UNIX Removing Hosts From a Cluster

Step 1 Remove the LSF section from the host’s system startup files. This undoes what you have done previously to start LSF daemons at boot time. See ‘*Starting LSF Servers at Boot Time*’ in the *LSF Installation Guide* for details.

Step 2 If any users use `lscsh` as their login shell, change their login shell to `tcsh` or `csh`. Remove `lscsh` from the `/etc/shells` file.

2 Managing LSF Base

Customizing Host Resources

Your cluster is most likely heterogeneous. Even if your computers are all the same, it might still be heterogeneous. For example, some machines are configured as file servers, while others are compute servers; some have more memory, others have less; some have four CPUs, others have only one; some have host-locked software licenses installed, others do not.

LSF provides powerful resource selection mechanisms so that correct hosts with required resources are chosen to run your jobs. For maximum flexibility, you should characterize your resources clearly enough so that users have satisfactory choices. For example, if some of your machines are connected to both Ethernet and FDDI, while others are only connected to Ethernet, then you probably want to define a resource called `fddi` and associate the `fddi` resource to machines connected to FDDI. This way, users can specify resource `fddi` if they want their jobs to run on machines connected to FDDI.

To customize host resources for your cluster, perform the following procedure:

- Step 1** Log in to any host in the cluster as the LSF administrator.
- Step 2** Define new resource names by modifying the “Resource” section of the `lsf.shared` file. Add a brief description to each of the added resource names. Resource descriptions will be displayed to a user by `lsinfo` command.
- Step 3** If you want to associate added resource names to an application, edit `lsf.task` file properly to reflect the resource in the resource requirements of the application. Alternatively, you can leave this to individual users who can use `lsrtasks` command to customize their own files.
- Step 4** Edit the `lsf.cluster.cluster` file to modify the `RESOURCES` column of the “Host” section so that all hosts that have the added resources will now have the added resource names in that column.
- Step 5** Follow instructions in ‘*Reconfiguring an LSF Cluster*’ on page 62 to reconfigure your LSF cluster.

Configuring Resources in LSF Base

Resources are defined in the `Resource` section of the `lsf.shared` file. The definition of a resource involves specifying a name and description, as well as, optionally, the type of its value, its update interval, and whether a higher or lower value indicates greater availability.

The mandatory resource information fields are:

- A `RESOURCENAME` indicating the name of the resource
- A `DESCRIPTION` that should indicate what the resource represents.

The optional resource information fields are:

- A `TYPE` indicating its value (boolean, numeric, or string)
- An `INTERVAL` indicating how often the value is updated (for resources whose value changes dynamically)
- An `INCREASING` flag indicating whether a higher value represents a greater availability of the resource (for numeric resources which can be used for scheduling jobs).

When the optional attributes are not specified, the resource is treated as static and boolean-valued.

The following is a sample of a `Resource` section from an `lsf.shared` file:

```
Begin Resource
RESOURCENAME TYPE INTERVAL INCREASING DESCRIPTION
mips          Boolean  ()    ()    (MIPS architecture)
dec           Boolean  ()    ()    (DECStation system)
sparc        Boolean  ()    ()    (SUN SPARC)
hppa         Boolean  ()    ()    (HPPA architecture)
bsd          Boolean  ()    ()    (BSD unix)
sysv         Boolean  ()    ()    (System V UNIX)
hpux         Boolean  ()    ()    (HP-UX UNIX)
aix          Boolean  ()    ()    (AIX UNIX)
nt           Boolean  ()    ()    (Windows NT)
scratch      Numeric  30    N     (Shared scratch space on server)
```

2 Managing LSF Base

```
synopsys      Numeric  30   N           (Floating licenses for Synopsys)
verilog       Numeric  30   N           (Floating licenses for Verilog)
console       String   30   N           (User Logged in on console)
End Resource
```

There is no distinction between shared and non-shared resources in the resource definition in the `lsf.shared` file.

Note

The `NewIndex` section in the `lsf.shared` file is obsolete. To achieve the same effect, the `Resource` section of the `lsf.shared` file can be used to define a dynamic numeric resource, and the `default` keyword can be used in the `LOCATION` field of the `ResourceMap` section of the `lsf.cluster.cluster` file.

Associating Resources with Hosts

Resources are associated with the host(s) on which they are available in the `ResourceMap` section of the `lsf.cluster.cluster` file (where `cluster` is the name of the cluster). The following fields must be completed for each resource:

- A `RESOURCENAME` indicating the name of the resource, as defined in the `lsf.shared` file
- A `LOCATION` indicating whether the resource is shared or non-shared, across which hosts, and with which initial value(s).

The following is an example of a `ResourceMap` section from an `lsf.cluster.cluster` file:

```
Begin ResourceMap
RESOURCENAME  LOCATION
verilog       5@[all]
synopsys      (2@[apple] 2@[others])
console       (1@[apple] 1@[orange])
End ResourceMap
```

The possible states of a resource that may be specified in the `LOCATION` column are:

- Each host in the cluster has the resource

- The resource is shared by all hosts in the cluster
- There are multiple instances of a resource within the cluster, and each instance is shared by a unique subset of hosts.

For static resources, the `LOCATION` column should contain the value of the resource.

The syntax of the information in the `LOCATION` field takes one of two forms. For static resources, where the value must be specified, use:

- `(value1@[host1 host2 ...] value2@[host3 host4] ...)`

For dynamic resources, where the value is updated by an `ELIM`, use:

- `([host1 host2 ...] [host3 host4 ...] ...)`

Each set of hosts listed within the square brackets specifies an instance of the resource. All hosts within the instance share the resource whose quantity is indicated by its value. In the above example, `host1, host2,...` form one instance of the resource, `host3, host4,...` form another instance, and so on.

Note

The same host cannot be in more than one instance of a resource.

Three predefined words have special meaning in this specification:

- `all` refers to all the server hosts in the cluster; for example, `value@[all]` means the resource is shared by all server hosts in the cluster made up of `host1 host2 ... hostn`
- `others` refers to the rest of the server hosts listed in the cluster; for example, `(2@[apple] 2[others])` means there are 2 units of "syno" on apple, and 2 shared by all other hosts
- `default` refers to each host; for example, `value@[default]` is equivalent to `(value@[host1] value@[host2] ... value@[hostn])` where `host1, ... hostn` are all server hosts in the cluster.

These syntax examples assume that static resources (requiring values) are being specified. For dynamic resources, use the same syntax but omit the value.

2 Managing LSF Base

The following items should be taken into consideration when configuring resources under LSF Base.

In the `lsf.cluster.cluster` file, the `Host` section must precede the `ResourceMap` section since the `ResourceMap` section uses the host names defined in the `Host` section.

- The `RESOURCES` column in the `Host` section of the `lsf.cluster.cluster` file should be used to associate static boolean resources with particular hosts. Using the `ResourceMap` section for static boolean resources section will result in an empty `RESOURCES` column in the `lshosts(1)` display.
- All resources specified in the `ResourceMap` section are treated as shared resources, which are displayed using the `lsload -s` or `lshosts -s` commands. The exception is for dynamic numeric resources specified using the `default` predefined word. These will be treated together with load indices such as `mem` and `swap` and are viewed using the `lsload -l` command.

If the `ResourceMap` section is not defined, then any dynamic resources specified in `lsf.shared` are considered to be host-based (the resource is available on each host in the cluster).

Reconfiguring an LSF Cluster

After changing LIM configuration files, you must tell LIM to read the new configuration. Use the `lsadmin` command to tell LIM to pick up the new configuration.

Operations can be specified on the command line or entered at a prompt. Run the `lsadmin` command with no arguments, and enter `help` to see the available operations.

The `lsadmin reconfig` command checks the LIM configuration files for errors. If no errors are found, the command confirms that you want to restart the LIMs on all hosts, and reconfigures all the LIM daemons:

```
% lsadmin reconfig
Checking configuration files ...
No errors found.

Do you really want to restart LIMs on all hosts? [y/n] y
Restart LIM on <hostD> ..... done
Restart LIM on <hostA> ..... done
Restart LIM on <hostC> ..... done
```

In the above example, no errors are found. If any non-fatal errors are found, the command asks you to confirm the reconfiguration. If fatal errors are found, the reconfiguration is aborted.

If you want to see details on any errors, run the command `lsadmin ckconfig -v`. This reports all errors to your terminal.

If you change the configuration file of LIM, you should also reconfigure LSF Batch by running `badmin reconfig` because LSF Batch depends on LIM configuration. If you change the configuration of LSF Batch, then you only need to run `badmin reconfig`.

External Resource Collection

The values of static external resources are specified through the `lsf.cluster.cluster` configuration file. All dynamic resources, regardless of whether they are shared or host-based, are collected through an ELIM. An ELIM is started in the following situations:

- On every host if any dynamic resource is configured as host-based. For example, if the `LOCATION` field in the `ResourceMap` section of `lsf.cluster.cluster` is `([default])`, then every host will start an ELIM.
- On the master host for any cluster-wide resources. For example, if the `LOCATION` field in the `ResourceMap` section of `lsf.cluster.cluster` is `([all])`, then an ELIM is started on the master host.
- On the first host specified for each instance, if multiple instances of the resource exist within the cluster. For example, if the `LOCATION` field in the `ResourceMap`

2 Managing LSF Base

section of `lsf.cluster.cluster` is (`[hostA hostB hostC] [hostD hostE hostF]`), then an ELIM will be started on `hostA` and `hostD` to report the value of that resource for that set of hosts.

If the host reporting the value for an instance goes down, then an ELIM is started on the next available host in the instance. In above example, if `hostA` became unavailable, an ELIM is started on `hostB`. If the `hostA` becomes available again then the ELIM on `hostB` is shut down and the one on `hostA` is started.

Note

There is only one ELIM on each host, regardless of the number of resources on which it reports. If only cluster-wide resources are to be collected, then an ELIM will only be started on the master host. When LIM starts, the following environment variables are set for ELIM:

- `LSF_MASTER`: This variable is defined if the ELIM is being invoked on the master host. It is undefined otherwise. This can be used to test whether the ELIM should report on cluster-wide resources that only need to be collected on the master host.
- `LSF_RESOURCES`: This variable contains a list of resource names (separated by spaces) on which the ELIM is expected to report. A resource name is only put in the list if the host on which the ELIM is running shares an instance of that resource.

Restrictions

The following restrictions apply to the use of shared resources in LSF products.

- A shared resource cannot be used as a load threshold in the `Hosts` section of the `lsf.cluster.cluster` file.
- A shared resource cannot be used in the `loadSched/loadStop` thresholds, or in the `STOP_COND` or `RESUME_COND` parameters in the queue definition in the `lsb.queues` file.

Writing an External LIM

The ELIM can be any executable program, either an interpreted script or compiled code. Example code for an ELIM is included in the `examples` directory in the LSF

distribution. The `elim.c` file is an ELIM written in C. You can customize this example to collect the load indices you want.

The ELIM communicates with the LIM by periodically writing a load update string to its standard output. The load update string contains the number of indices followed by a list of name-value pairs in the following format:

```
N name1 value1 name2 value2 ... nameN valueN
```

For example,

```
3 tmp2 47.5 nio 344.0 licenses 5
```

This string reports three indices: `tmp2`, `nio`, and `licenses`, with values 47.5, 344.0, and 5 respectively. Index values must be numbers between `-INFINIT_LOAD` and `INFINIT_LOAD` as defined in the `lsf.h` header file.

If the ELIM is implemented as a C program, as part of initialization it should use `setbuf(3)` to establish unbuffered output to `stdout`.

The ELIM should ensure that the entire load update string is written successfully to `stdout`. This can be done by checking the return value of `printf(3s)` if the ELIM is implemented as a C program or as the return code of `/bin/echo(1)` from a shell script. The ELIM should exit if it fails to write the load information.

Each LIM sends updated load information to the master every 15 seconds. Depending on how quickly your external load indices change, the ELIM should write the load update string at most once every 15 seconds. If the external load indices rarely change, the ELIM can write the new values only when a change is detected. The LIM continues to use the old values until new values are received.

The executable for the ELIM must be in `LSF_SERVERDIR` and must have the name `elim`. If LIM expects some resources to be collected by an ELIM according to configuration, it invokes the ELIM automatically on startup. The ELIM runs with the same user id and file access permission as the LIM.

2 Managing LSF Base

The LIM restarts the ELIM if it exits; to prevent problems in case of a fatal error in the ELIM, it is restarted at most once every 90 seconds. When the LIM terminates, it sends a SIGTERM signal to the ELIM. The ELIM must exit upon receiving this signal.

Overriding Built-In Load Indices

The ELIM can also return values for the built-in load indices. In this case the value produced by the ELIM overrides the value produced by the LIM. The ELIM must ensure that the semantics of any index it supplies are the same as that of the corresponding index returned by the `lsinfo(1)` command.

For example, some sites prefer to use `/usr/tmp` for temporary files. To override the `tmp` load index, write a program that periodically measures the space in the `/usr/tmp` file system and writes the value to standard output. Name this program `elim` and store it in the `LSF_SERVERDIR` directory.

Note

The name of an external load index must not be one of the resource name aliases `cpu`, `idle`, `logins`, or `swap`. To override one of these indices, use its formal name: `rlm`, `it`, `ls`, or `swp`.

You must configure the external load index even if you are overriding a built-in load index.

LIM Policies

LIM provides very critical services to the all LSF components. In addition to the timely collection of resource information, LIM also provides host selection and job placement policies. If you are using the LSF MultiCluster product, LIM policies also determine how different clusters should exchange load and resource information.

LIM policies are advisory information for applications. Applications can either use the placement decision from the LIM, or make further decisions based on information from the LIM.

Most of the LSF interactive tools, such as `lsrun` and `lscsh`, use LIM policies to place jobs on the network. LSF Batch uses load and resource information from LIM and makes its own placement decisions based on other factors in addition to load information.

As was described in ‘*Overview of LSF Configuration Files*’ on page 50, LIM configuration file defines load-sharing policies. The LIM configuration parameters that affect LIM policies include:

- Load threshold parameters. These define the conditions beyond which a host is considered busy by LIM. No jobs will be dispatched to a busy host by LIM’s policy. Each of these parameters is a load index value, so that if the host load goes beyond that value, the host becomes busy.

If a particular load index is not specified, then LIM assumes that there is no threshold for that load index. Define looser values for load thresholds if you want to aggressively run jobs on a host. See ‘*Threshold Fields*’ on page 184 for details about load thresholds.

- Dispatch window parameter. This defines one or more time windows during which a host is considered available for sharing a load from other hosts. If the current time is outside all the defined time windows, the host is considered locked and LIM will not advise any applications to run jobs on the host.

If you do not want LIM to place jobs to some hosts during certain hours, you can define run windows for these hosts in the `lsf.cluster.cluster`. Dispatch windows in `lsf.cluster.cluster` cause hosts to become locked outside the time windows so that LIM will not advise jobs to go to those hosts. Details of this parameter are described in ‘*Hosts*’ on page 182.

Note

LIM thresholds and run windows affect the job placement advice of the LIM. Job placement advice is not enforced by LIM. LSF Batch, for example, does not follow the policies of the LIM.

- Intercluster policies. These are parameters specified in the RemoteClusters section of the `lsf.cluster.cluster` file. These parameters apply to LSF MultiCluster product only. The parameters define the relationship between the local cluster and remote clusters and the direction of job placement flows across clusters. See ‘*Managing LSF MultiCluster*’ on page 143 for details.

2 Managing LSF Base

There are two main goals in adjusting the LIM configuration parameters: improving response time, and reducing interference with interactive use. To improve response time, LSF should be tuned to correctly select the best available host for each job. To reduce interference, LSF should be tuned to avoid overloading any host.

Tuning CPU Factors

CPU factors are used to differentiate the relative speed of different machines. LSF runs jobs on the best possible machines so that the response time is minimized. To achieve this, it is important that you define correct CPU factors for each machine model in your cluster by changing the `HostModel` section of your `lsf.shared` file.

CPU factors should be set based on a benchmark that reflects your work load. (If there is no such benchmark, CPU factors can be set based on raw CPU power.) The CPU factor of the slowest hosts should be set to one, and faster hosts should be proportional to the slowest. For example, consider a cluster with two hosts, `hostA` and `hostB`, where `hostA` takes 30 seconds to run your favourite benchmark and `hostB` takes 15 seconds to run the same test. `hostA` should have a CPU factor of 1, and `hostB` (since it is twice as fast) should have a CPU factor of 2.

LSF uses a normalized CPU performance rating to decide which host has the most available CPU power. The normalized ratings can be seen by running the `lsload -N` command. The hosts in your cluster are displayed in order from best to worst. Normalized CPU run queue length values are based on an estimate of the time it would take each host to run one additional unit of work, given that an unloaded host with CPU factor 1 runs one unit of work in one unit of time.

Incorrect CPU factors can reduce performance in two ways. If the CPU factor for a host is too low, that host may not be selected for job placement when a slower host is available. This means that jobs would not always run on the fastest available host. If the CPU factor is too high, jobs are run on the fast host even when they would finish sooner on a slower but lightly loaded host. This causes the faster host to be overused while the slower hosts are underused.

Both of these conditions are somewhat self-correcting. If the CPU factor for a host is too high, jobs are sent to that host until the CPU load threshold is reached. The LIM then marks that host as busy, and no further jobs will be sent there. If the CPU factor is too

low, jobs may be sent to slower hosts. This increases the load on the slower hosts, making LSF more likely to schedule future jobs on the faster host.

Tuning LIM Load Thresholds

The `Host` section of the `lsf.cluster.cluster` file can contain busy thresholds for load indices. You do not need to specify a threshold for every index; indices that are not listed do not affect the scheduling decision. These thresholds are a major factor in influencing LSF performance. This section does not describe all LSF load indices; see *'Resource Requirements'* on page 24 and *'Threshold Fields'* on page 184 for more complete discussions.

The parameters that most often affect performance are:

r15s	15-second average
r1m	1-minute average
r15m	15-minute average
pg	paging rate in pages per second
swp	Available swap space

For tuning these parameters, you should compare the output of `lsload` to the thresholds reported by `lshosts -l`.

The `lsload` and `lsmon` commands display an asterisk '*' next to each load index that exceeds its threshold. For example, consider the following output from `lshosts -l` and `lsload`:

2 Managing LSF Base

```
% lshosts -l
HOST_NAME: hostD
...
LOAD_THRESHOLDS:
  r15s  r1m  r15m  ut    pg    io    ls    it    tmp    swp    mem
    -    3.5    -    -    15    -    -    -    -    2M    1M

HOST_NAME: hostA
...
LOAD_THRESHOLDS:
  r15s  r1m  r15m  ut    pg    io    ls    it    tmp    swp    mem
    -    3.5    -    -    15    -    -    -    -    2M    1M

% lsload
HOST_NAME status r15s  r1m  r15m  ut    pg  ls  it  tmp  swp  mem
hostD      ok  0.0  0.0  0.0  0%   0.0  6  0  30M  32M  10M
hostA      busy 1.9  2.1  1.9 47% *69.6 21  0  38M  96M  60M
```

In this example, *hostD* is *ok*. However, *hostA* is *busy*; the *pg* (paging rate) index is 69.6, above the threshold of 15.

Other monitoring tools such as `xlsmon` also help to show the effects of changes.

If the LIM often reports a host to be *busy* when the CPU run queue length is low, the most likely cause is the paging rate threshold. Different operating systems assign subtly different meanings to the paging rate statistic, so the threshold needs to be set at different levels for different host types. In particular, HP-UX systems need to be configured with significantly higher *pg* values; try starting at a value of 50 rather than the default of 15.

If the LIM often shows systems *busy* when the CPU utilization and run queue lengths are relatively low and the system is responding quickly, try raising the *pg* threshold. There is a point of diminishing returns; as the paging rate rises, eventually the system spends too much time waiting for pages and the CPU utilization decreases. Paging rate is the factor that most directly affects perceived interactive response. If a system is paging heavily, it feels very slow.

The CPU run queue threshold can be reduced if you find that interactive jobs slow down your response too much while the LIM still reports your host as *ok*. Likewise, it can be increased if hosts become *busy* at too low a load.

On multi-processor systems, the CPU run queue threshold is compared to the effective run queue length as displayed by the `lsload -E` command. The run queue threshold should be configured as the load limit for a single processor. Sites with a variety of uniprocessor and multi-processor machines can use a standard value for `r15s`, `r1m` and `r15m` in the configuration files, and the multi-processor machines will automatically run more jobs. Note that the normalized run queue length printed by `lsload -N` is scaled by the number of processors. See *Section 4, 'Resources'*, beginning on page 35 of the *LSF Batch User's Guide* and `lsfintro(1)` for the concept of effective and normalized run queue lengths.

Cluster Monitoring with LSF

Because LSF takes a wide variety of measurements on the hosts in your network, it can be a powerful tool for monitoring and capacity planning. The `lsmon` command gives updated information that can quickly identify problems such as inaccessible hosts or unusual load levels. The `lsmon -L` option logs the load information to a file for later processing. See the `lsmon(1)` and `lim.acct(5)` manual pages for more information.

For example, if the paging rate (`pg`) on a host is always high, adding memory to the system will give a significant increase in both interactive performance and total throughput. If the `pg` index is low but the CPU utilization (`ut`) is usually more than 90 percent, the CPU is the limiting resource. Getting a faster host, or adding another host to the network, would provide the best performance improvement. The external load indices can be used to track other limited resources such as user disk space, network traffic, or software licenses.

The `xlsmon` program is a Motif graphic interface to the LSF load information. The `xlsmon` display uses colour to highlight busy and unavailable hosts, and can show both the current levels and scrolling histories of selected load indices.

See *Section 3, 'Cluster Information'*, beginning on page 25 of the *LSF Batch User's Guide* for more information about `xlsmon`.

LSF License Management

LSF software is licensed using the FLEXlm license manager from Globetrotter Software, Inc. The LSF license key controls the hosts allowed to run LSF. The procedures for obtaining, installing, and upgrading license keys are described in 'Getting License Key Information' and 'Setting Up the License Key' in the *LSF Installation Guide*. This section provides background information on FLEXlm.

FLEXlm controls the total number of hosts configured in all your LSF clusters. You can organize your hosts into clusters however you choose. Each server host requires at least one license; multi-processor hosts require more than one, as a function of the number of processors. Each client host requires 1/5 of a license.

LSF uses two kinds of FLEXlm license: time-limited DEMO licenses and permanent licenses.

The DEMO license allows you to try LSF out on an unlimited number of hosts on any supported host type. The trial period has a fixed expiry date, and the LSF software will not function after that date. DEMO licenses do not require any additional daemons.

Permanent licenses are the most common. A permanent license limits only the total number of hosts that can run the LSF software, and normally has no time limit. You can choose which hosts in your network will run LSF, and how they are arranged into clusters. Permanent licenses are counted by a license daemon running on one host on your network.

For permanent licenses, you need to choose a license server host and send hardware host identification numbers for the license server host to your software vendor. The vendor uses this information to create a permanent license that is keyed to the license server host. Some host types have a built-in hardware host ID; on others, the hardware address of the primary LAN interface is used.

How FLEXlm Works

FLEXlm is used by many software packages because it provides a simple and flexible method for controlling access to licensed software. A single FLEXlm license server can handle licenses for many software packages, even if those packages come from different vendors. This reduces the systems administration load, since you do not need to install a new license manager every time you get a new package.

The License Server Daemon

FLEXlm uses a daemon called `lmgrd` to manage permanent licenses. This daemon runs on one host on your network, and handles license requests from all applications. Each license key is associated with a particular software vendor. `lmgrd` automatically starts a *vendor daemon*; the LSF version is called `lsf_ld` and is provided by Platform Computing Corporation. The vendor daemon keeps track of all licenses supported by that vendor. DEMO licenses do not require you to run license daemons.

The license server daemons should be run on a reliable host, since licensed software will not run if it cannot contact the server. The FLEXlm daemons create very little load, so they are usually run on the file server. If you are concerned about availability, you can run `lmgrd` on a set of three or five hosts. As long as a majority of the license server hosts are available, applications can obtain licenses.

The License File

Software licenses are stored in a text file. The default location for this file is `/usr/local/flexlm/licenses/license.dat`, but this can be overridden. For example, when LSF is installed following the default installation procedure, the license file is installed in the same directory where all LSF configuration files are installed; for example, `/usr/local/lsf/mt/conf`. The license file must be readable on every host that runs licensed software. It is most convenient to place the license file in a shared NFS directory.

The `license.dat` file normally contains:

- A `SERVER` line for each FLEXlm server host. The `SERVER` line contains the host name, hardware host ID, and network port number for the server.
- A `DAEMON` line for each software vendor, which gives the file path name of the vendor daemon.
- A `FEATURE` line for each software license. This line contains the number of copies that can be run, along with other necessary information.

The `FEATURE` line contains an encrypted code to prevent tampering. For permanent licenses, the licenses granted by the `FEATURE` line can be accessed only through license servers listed on the `SERVER` lines.

2 Managing LSF Base

For DEMO licenses, no FLEXlm daemons are needed, so the license file contains only the `FEATURE` line.

Here is an example of a DEMO license file.

```
FEATURE lsf_base lsf_ld 3.100 20-Dec-1997 0 5CE371439854221102F7 "Platform" DEMO
FEATURE lsf_batch lsf_ld 3.100 20-Dec-1997 0 3CC371C33076712F433B "Platform" DEMO
FEATURE lsf_multicluster lsf_ld 3.100 20-Dec-1997 0 5C63119330771250944C "Platform" DEMO
```

This license file allows a site to run LSF Base, Batch, and MultiCluster until December 20, 1997. Note that a DEMO license does not have a `SERVER` line and a `DAEMON` line because no license server is needed for DEMO licenses.

The following is an example of a permanent license:

```
SERVER hostD 690a377d 1700
DAEMON lsf_ld /usr/local/lsf/etc/lsf_ld
FEATURE lsf_base lsf_ld 3.100 1-jan-0000 1000 5C239486C4D72739BAF8 "Platform"
FEATURE lsf_batch lsf_ld 3.100 1-jan-0000 1000 6CB344F6E2A5B7A31526 "Platform"
FEATURE lsf_multicluster lsf_ld 3.100 1-jan-0000 1000 5C535446DAE5DEE6B736 "Platform"
```

LSF uses the notion of license units in calculating the amount of licenses required for a product on a host. The number of license units required to run LSF depends on the number of CPUs the host has as well as the type of the machine. For example, a single CPU HP-UX machine would require ten license units, whereas a client-only machine would need two license units.

The above license is configured to run on `hostD`, using TCP port 1700. This license allows 1000 license units for version 3.1 of LSF Base, LSF Batch, and LSF MultiCluster.

License Management Utilities

FLEXlm provides several utility programs for managing software licenses. These utilities and their manual pages are included in the LSF software distribution.

Because these utilities can be used to shut down the FLEXlm license server, and thus prevent licensed software from running, they are installed in the `LSF_SERVERDIR` directory. The file permissions are set so that only `root` and members of group 0 can use them.

The utilities included are:

lmcksum

Calculate check sums of the license key information

lmdown

Shut down the FLEXlm server

lmhostid

Display the hardware host ID

lmremove

Remove a feature from the list of checked out features

lmreread

Tell the license daemons to re-read the license file

lmstat

Display the status of the license servers and checked out licenses

lmver

Display the FLEXlm version information for a program or library

For complete details on these commands, see the on-line manual pages.

Updating an LSF License

FLEXlm only accepts one license key for each feature listed in a license key file. If there is more than one `FEATURE` line for the same feature, only the first `FEATURE` line is used. To add hosts to your LSF cluster, you must replace the old `FEATURE` line with a new one listing the new total number of licenses.

The procedure for updating a license key file to include new license keys is described in *'Adding a Permanent License'* in the *LSF Installation Guide*.

Changing the FLEXlm Server TCP Port

The fourth field on the `SERVER` line specifies the TCP port number that the FLEXlm server uses. Choose an unused port number. LSF usually uses port numbers in the range 3879 to 3882, so the numbers from 3883 forward are good choices. If the `lmgrd`

2 Managing LSF Base

daemon complains that the license server port is in use, you can choose another port number and restart `lmgrd`.

For example, if your license file contains the line:

```
SERVER hostname host-id 1700
```

and you want your FLEXlm server to use TCP port 3883, change the SERVER line to:

```
SERVER hostname host-id 3883
```

Modifying LSF Products and Licensing

LSF Suite 3.1 includes the following products: LSF Base, LSF Batch, LSF JobScheduler, LSF MultiCluster, and LSF Analyzer.

The configuration changes to enable a particular product in a cluster are handled during installation by `lsfsetup`. If at some later time you want to modify the products in your cluster, edit the `PRODUCTS` line in the `Parameters` section of the `lsf.cluster.cluster` file. You can specify one or more of the strings `LSF_Base`, `LSF_Batch`, `LSF_JobScheduler`, `LSF_Analyzer`, and `LSF_MultiCluster` to enable the operation of LSF Base, LSF Batch, LSF JobScheduler, LSF Analyzer, and LSF MultiCluster, respectively. If any of `LSF_Batch`, `LSF_JobScheduler`, or `LSF_MultiCluster` are specified, then `LSF_Base` is automatically enabled as well.

If the `lsf.cluster.cluster` file is shared, adding a product name to the `PRODUCTS` line enables that product for all hosts in the cluster. For example, to enable the operation of LSF Base, LSF Batch, and LSF MultiCluster:

```
Begin Parameters
PRODUCTS=LSF_Base LSF_Batch LSF_MultiCluster
End Parameters
```

To enable the operation of LSF Base only:

```
Begin Parameters
PRODUCTS=LSF_Base
End Parameters
```

To enable the operation of LSF JobScheduler:

```

Begin Parameters
PRODUCTS=LSF_JobScheduler LSF_Base
End Parameters

```

Selected Hosts

It is possible to indicate that only certain hosts run LSF Batch or LSF JobScheduler within a cluster. This is done by specifying `LSF_Batch` or `LSF_JobScheduler` in the `RESOURCES` field on the `HOSTS` section of the `lsf.cluster.cluster` file. For example, the following enables hosts `hostA`, `hostB`, and `hostC` to run LSF JobScheduler and hosts `hostD`, `hostE`, and `hostF` to run LSF Batch.

```

Begin Parameters
PRODUCTS=LSF_Batch LSF_Base
End Parameters

```

```

Begin      Host
HOSTNAME   model      type  server  RESOURCES
hostA      SUN41  SPARC5LC  1      (sparc bsd LSF_JobScheduler)
hostB      HPPA9   HP735   1      (linux LSF_JobScheduler)
hostC      SGI     SGIINDIG 1      (irix cs LSF_JobScheduler)
hostD      SUNSOL  SunSparc 1      (solaris)
hostE      HP_UX   A900    1      (hpux cs bigmem)
hostF      ALPHA  DEC5000 1      (alpha)
End Hosts

```

The license file used to serve the cluster must have the corresponding features. A host will show as unlicensed if the license for the component it was configured to run is unavailable. For example, if a cluster is configured to run `LSF_Batch` on all hosts, and the license file does not contain the `LSF_JobScheduler` feature, then the hosts will be unlicensed, even if there are licenses for LSF Base.

3. Managing LSF Batch

This chapter describes the operating concepts and maintenance tasks of the batch queuing system, LSF Batch. This chapter requires you to understand concepts from *'Managing LSF Base'* on page 45. The topics covered in this chapter are:

- managing LSF Batch logs
- duplicate event logging
- controlling LSF Batch servers
- controlling LSF Batch queues
- managing LSF Batch configuration
- validating job submissions
- controlling LSF Batch jobs
- forcing job execution
- managing an LSF Cluster using `xlsadmin`

Managing LSF Batch Logs

Managing error log files for LSF Batch daemons was described in *'Managing Error Logs'* on page 45. This section discusses the other important log files LSF Batch daemons produce. The LSF Batch log files are found in the directory `LSB_SHAREDIR/cluster/logdir`.

3 Managing LSF Batch

LSF Batch Accounting Log

Each time a batch job completes or exits, an entry is appended to the `lsb.acct` file. This file can be used to create accounting summaries of LSF Batch system use. The `bacct(1)` command produces one form of summary. The `lsb.acct` file is a text file suitable for processing with `awk`, `perl`, or similar tools. See the `lsb.acct(5)` manual page for details of the contents of this file. Additionally, the LSF Batch API supports calls to process the `lsb.acct` records. See the *LSF Programmer's Guide* for details of LSF Batch API.

You should move the `lsb.acct` file to a backup location, and then run your accounting on the backup copy. The daemon automatically creates a new `lsb.acct` file to replace the moved file. This prevents problems that might occur if the daemon writes new log entries while the accounting programs are running. When the accounting is complete, you can remove or archive the backup copy.

LSF Batch Event Log

The LSF Batch daemons keep an event log in the `lsb.events` file. The `mbatchd` daemon uses this information to recover from server failures, host reboots, and LSF Batch reconfiguration. The `lsb.events` file is also used by the `bhist` command to display detailed information about the execution history of batch jobs, and by the `badmin` command to display the operational history of hosts, queues, and LSF Batch daemons.

For performance reasons, the `mbatchd` automatically backs up and rewrites the `lsb.events` file after every 1000 batch job completions (this is the default; the value is controlled by the `MAX_JOB_NUM` parameter in the `lsb.params` file). The old `lsb.events` file is moved to `lsb.events.1`, and each old `lsb.events.n` file is moved to `lsb.events.n+1`. The `mbatchd` never deletes these files. If disk storage is a concern, the LSF administrator should arrange to archive or remove old `lsb.events.n` files occasionally.

CAUTION!

Do not remove or modify the `lsb.events` file. Removing or modifying the `lsb.events` file could cause batch jobs to be lost.

Duplicate Event Logging

By default, LSF Batch stores all state information needed to recover from server failures, host reboots, or reconfiguration in a file in the `LSB_SHAREDIR` directory. Typically, the `LSB_SHAREDIR` directory resides on a reliable file server that also contains other critical applications necessary for running user's jobs. This is performed because, if the central file server is unavailable, user's applications cannot run, and the failure of LSF Batch to continue processing user's jobs is a secondary issue.

For sites not wishing to rely solely on a central file server for recovery information, LSF can be configured to maintain a replica of the recovery file. The replica is stored on the file server, and used if the primary copy is unavailable—referred to as *duplicate event logging*. When LSF is configured this way, the primary event log is stored on the first master host, and re-synchronized with the replicated copy when the host recovers.

Configuring Duplicate Event Logging

To enable the replication feature, define `LSB_LOCALDIR` in the `lsf.conf` file. `LSB_LOCALDIR` should be a local directory and it should exist *only* on the first master host (that is, the first host configured in the `lsf.cluster.cluster` file).

`LSB_LOCALDIR` is used to store the primary copy of the batch state information. The contents of `LSB_LOCALDIR` are copied to a replica in `LSB_SHAREDIR` which resides on a central file server. As before, `LSB_SHAREDIR` is assumed to be accessible from all hosts which can potentially become the master.

How Duplicate Event Logging Works

With the replication feature enabled the following scenarios can occur:

Failure of File Server

If the file server containing `LSB_SHAREDIR` goes down, LSF will continue to process jobs. Client commands such as `bhist(1)` and `bacct(1)` which directly read `LSB_SHAREDIR` will not work. When the file server recovers, the replica in `LSB_SHAREDIR` will be updated.

3 Managing LSF Batch

Failure of First Master Host

If the first master host fails, then the primary copy of the recovery file in the `LSB_LOCALDIR` directory becomes unavailable. A new master host will be selected which will use the recovery file replica in `LSB_SHAREDIR` to restore its state and to log future events. Note that there is no replication by the second master.

Recovery of First Master Host

When the first master host becomes available again, it will update the primary copy in `LSB_LOCALDIR` from the replica in `LSB_SHAREDIR` and continue operations as before.

The replication feature improves the reliability of LSF Batch operations provided that the following assumptions hold:

- Failure of the LSF master host only occurs from the first master to the second master. The replication feature is not active if the second master also fails and a third master takes over.
- The master host containing `LSB_LOCALDIR` and the file server containing `LSB_SHAREDIR` do not fail simultaneously. In this situation, LSF Batch will be unavailable.
- Network partitioning causing a cluster to split into two independent clusters each simultaneously running a `mbatchd` does not occur. This may happen given certain network topologies and failure modes. For example, connectivity is lost between the first master, M1, and both the file server and the secondary master, M2. Both M1 and M2 will run the `mbatchd` service with M1 logging events to `LSB_LOCALDIR` and M2 logging to `LSB_SHAREDIR`. When connectivity is restored, the changes made by M2 to `LSB_SHAREDIR` will be lost when M1 updates `LSB_SHAREDIR` from its copy in `LSB_LOCALDIR`.

Controlling LSF Batch Servers

The `lsadmin` command is used to control LSF Base daemons, LIM, and RES. LSF Batch has the `badmin` command to perform similar operations on LSF Batch daemons.

LSF Batch System Status

To check the status of LSF Batch server hosts and queues, use the `bhosts` and `bqueues` commands:

`% bhosts`

HOST_NAME	STATUS	JL/U	MAX	NJOBS	RUN	SSUSP	USUSP	RSV
hostA	ok	2	1	0	0	0	0	0
hostB	closed	2	2	2	2	0	0	0
hostD	ok	-	8	1	1	0	0	0

`% bqueues`

QUEUE_NAME	PRIO	STATUS	MAX	JL/U	JL/P	JL/H	NJOBS	PEND	RUN	SUSP
night	30	Open:Inactive	-	-	-	-	4	4	0	0
short	10	Open:Active	50	5	-	-	1	0	1	0
simulation	10	Open:Active	-	2	-	-	0	0	0	0
default	1	Open:Active	-	-	-	-	6	4	2	0

If the status of a batch server is 'closed', then it will not accept more jobs. The LSF administrator can force a job to run using the `brun(1)` command. See *'Forcing Job Execution — `brun -f` on page 98* for details.

Use the `bhosts -l` command to see more information about the status of closed servers. One of the following conditions will be indicated:

- `closed_Lock`
The host is locked by the LSF administrator or root. All batch jobs on the host are suspended by the `lsbatch` system.
- `closed_Adm`
The host is closed by the LSF administrator or root. No job can be dispatched to it but jobs that are executing on it will not be affected.
- `closed_Wind`
The host is closed by its dispatch windows, which are defined in the configuration file `lsb.hosts`.
- `closed_Full`
The configured maximum number of batch jobs on the host has been reached.

3 Managing LSF Batch

- `closed_Busy`
The host is busy because some load indices go beyond the configured thresholds. The overloaded thresholds are identified by an asterisk (*).
- `closed_LIM`
The LIM on the host is unreachable, but the `sbatchd` is ok.

An inactive queue will accept new job submissions, but will not dispatch any new jobs. A queue can become inactive if the LSF cluster administrator explicitly inactivates it via `badmin` command, or if the queue has a dispatch or run window defined and the current time is outside the time window.

`mbatchd` automatically logs the history of the LSF Batch daemons in the LSF Batch event log. You can display the administrative history of the batch system using the `badmin` command.

The `badmin hhist` command displays the times when LSF Batch server hosts are opened and closed by the LSF administrator.

The `badmin qhist` command displays the times when queues are opened, closed, activated, and inactivated.

The `badmin mbdhist` command displays the history of the `mbatchd` daemon, including the times when the master starts, exits, reconfigures, or changes to a different host.

The `badmin hist` command displays all LSF Batch history information, including all the events listed above.

Remote Start-up of `sbatchd`

You can use `badmin hstartup` command to start `sbatchd` on some or all remote hosts from one host:

```
% badmin hstartup all  
Start up slave batch daemon on <hostA> .....done  
Start up slave batch daemon on <hostB> .....done  
Start up slave batch daemon on <hostD> .....done
```

Note that you do not have to be *root* to use the `badadmin` command to start LSF Batch daemons.

For remote startup to work, file `/etc/lsf.sudoers` has to be set up properly and you have to be able to run `rsh` across all LSF hosts without having to enter a password. See *'The lsf.sudoers File' on page 189* for configuration details of `lsf.sudoers`.

Restarting `sbatchd`

`mbatchd` is restarted by the `badadmin reconfig` command. `sbatchd` can be restarted using the `badadmin hrestart` command:

```
% badadmin hrestart hostD
Restart slave batch daemon on <hostD> ..... done
```

You can specify more than one host name to restart `sbatchd` on multiple hosts, or use `all` to refer to all LSF Batch server hosts. Restarting `sbatchd` on a host does not affect batch jobs that are running on that host.

Shutting Down LSF Batch Daemons

The `badadmin hshutdown` command shuts down the `sbatchd`.

```
% badadmin hshutdown hostD
Shut down slave batch daemon on <hostD> .... done
```

If `sbatchd` is shutdown, that particular host will not be available for running new jobs. Existing jobs running on that host will continue to completion, but the results will not be sent to the user until `sbatchd` is later restarted.

To shut down `mbatchd` you must first use the `badadmin hshutdown` command to shut down the `sbatchd` on the master host, and then run the `badadmin reconfig` command. The `mbatchd` is normally restarted by `sbatchd`; if there is no `sbatchd` running on the master host, `badadmin reconfig` causes `mbatchd` to exit.

If `mbatchd` is shut down, all LSF Batch services will be temporarily unavailable. However, all existing jobs will not be affected. When `mbatchd` is later restarted, previous status will be restored from the event log file and job scheduling will continue.

3 Managing LSF Batch

Opening and Closing of Batch Server Hosts

Occasionally, you might want to drain a batch server host for the purposes of rebooting, maintenance, or host removal. This can be achieved by running the `badadmin hclose` command:

```
% badadmin hclose hostB  
Close <hostB> ..... done
```

When a host is open, LSF Batch can dispatch jobs to that host. When a host is closed, no new batch jobs are dispatched, but jobs already dispatched to the host continue to execute. To reopen a batch server host, run `badadmin hopen` command:

```
% badadmin hopen hostB  
Open <hostB> ..... done
```

To view the history of a batch server host, run `badadmin hhist` command:

```
% badadmin hhist hostB  
Wed Nov 20 14:41:58: Host <hostB> closed by administrator <lsf>.  
Wed Nov 20 15:23:39: Host <hostB> opened by administrator <lsf>.
```

Controlling LSF Batch Queues

Each batch queue can be open or closed, active or inactive. Users can submit jobs to open queues, but not to closed queues. Active queues start jobs on available server hosts, and inactive queues hold all jobs. The LSF administrator can change the state of any queue. Queues can also become active or inactive because of queue run or dispatch windows.

bqueues — Queue Status

The current status of a particular queue or all queues is displayed by the `bqueues(1)` command. The `bqueues -l` option also gives current statistics about the jobs in a particular queue such as the total number of jobs in this queue, the number of jobs running, suspended, and so on.

```
% bqueues normal
```

QUEUE_NAME	PRIO	STATUS	MAX	JL/U	JL/P	JL/H	NJOBS	PEND	RUN	SUSP
normal	30	Open:Active	-	-	-	2	6	4	2	0

Opening and Closing Queues

When a batch queue is open, users can submit jobs to the queue. When a queue is closed, users cannot submit jobs to the queue. If a user tries to submit a job to a closed queue, an error message is printed and the job is rejected. If a queue is closed but still active, previously submitted jobs continue to be processed. This allows the LSF administrator to drain a queue.

```
% badmin qclose normal
```

```
Queue <normal> is closed
```

```
% bqueues normal
```

QUEUE_NAME	PRIO	STATUS	MAX	JL/U	JL/P	JL/H	NJOBS	PEND	RUN	SUSP
normal	30	Closed:Active	-	-	-	2	6	4	2	0

```
% bsub -q normal hostname
```

```
normal: Queue has been closed
```

```
% badmin qopen normal
```

```
Queue <normal> is opened
```

Activating and Inactivating Queues

When a queue is active, jobs in the queue are started if appropriate hosts are available. When a queue is inactive, jobs in the queue are not started. Queues can be activated and inactivated by the LSF administrator using the `badmin qact` and `badmin qinact` commands, or by configured queue run or dispatch windows.

If a queue is open and inactive, users can submit jobs to the queue but no new jobs are dispatched to hosts. Currently running jobs continue to execute. This allows the LSF administrator to let running jobs complete before removing queues or making other major changes.

```
% badmin qinact normal
```

```
Queue <normal> is inactivated
```

```
% bqueues normal
```

QUEUE_NAME	PRIO	STATUS	MAX	JL/U	JL/P	JL/H	NJOBS	PEND	RUN	SUSP
normal	30	Open:Inctive	-	-	-	-	0	0	0	0

```
% badmin qact normal
```

```
Queue <normal> is activated
```

Managing LSF Batch Configuration

The LSF Batch cluster is a subset of the LSF Base cluster. All servers used by LSF Batch must belong to the base cluster; however, not all servers in the base cluster must provide LSF Batch services.

LSF Batch configuration consists of four files: `lsb.params`, `lsb.hosts`, `lsb.users`, and `lsb.queues`. These files are stored in `LSB_CONFDIR/cluster/configdir`, where `cluster` is the name of your cluster.

All these files are optional. If any of these files do not exist, LSF Batch will assume a default configuration.

The `lsb.params` file defines general parameters about LSF Batch system operation, such as the name of the default queue when the user does not specify one, scheduling intervals for `mbatchd` and `sbatchd`, and so on. Detailed parameters are described in *'The lsb.params File' on page 193*.

The `lsb.hosts` file defines LSF Batch server hosts together with their attributes. Not all LSF hosts defined by LIM configuration have to be configured to run batch jobs. Batch server host attributes include scheduling load thresholds, dispatch windows, and job slot limits. This file is also used to define host groups and host partitions. See *'The lsb.hosts File' on page 202* for details of this file.

The `lsb.users` file contains user-related parameters such as user groups, user job slot limits, and account mapping. See *'The lsb.users File' on page 198* for details.

The `lsb.queues` file defines job queues. Numerous controls are available at the queue level to allow cluster administrators to customize site resource allocation policies. See *'The lsb.queues File' on page 208* for more details.

When you first install LSF on your cluster, some example queues are already configured for you. You should customize these queues or define new queues to meet your site needs.

Note

After changing any of the LSF Batch configuration files, you need to run `badmin reconfig` to tell `mbatchd` to pick up the new configuration. You must also run this every time you change LIM configuration.

Adding a Batch Server Host

You can add a batch server host to LSF Batch configuration following the steps below:

- Step 1** If you are adding a host that has not been added to the LSF Base cluster yet, perform steps described in *'Adding a Host to a Cluster' on page 56*.
- Step 2** Modify `LSB_CONFDIR/cluster/configdir/lsb.hosts` file to add the new host together with its attributes. If you want to limit the added host for use only by some queues, you should also update `lsb.queues` file. Since host types and host models as well as the virtual name 'default' can be used to refer to all hosts of that type, model, or every other LSF host not covered by the definitions, you might not need to change any of the files if the host is already covered.
- Step 3** Run `badmin reconfig` to tell `mbatchd` to pick up the new configuration.
- Step 4** Start `sbatchd` on the added host by running `badmin hstartup` or simply start it manually.

Removing a Batch Server Host

To remove a host as a batch server host, follow the steps below:

- Step 1** If you need to permanently remove a host from your cluster, you should use `badmin hclose` to prevent new batch jobs from starting on the host, and wait for any running jobs on that host to finish. If you want to shut the host down before all jobs complete, use `bkill` to kill the running jobs.
- Step 2** Modify `lsb.hosts` and `lsb.queues` in `LSB_CONFDIR/cluster/configdir` directory and remove the host from any of the sections.
- Step 3** Run `badmin hshutdown` to shutdown `sbatchd` on that host.

CAUTION!

You should never remove the master host from LSF Batch. Change LIM configuration to assign a different default master host if you want to remove your current default master from the LSF Batch server pool.

3 Managing LSF Batch

Adding a Batch Queue

Adding a batch queue does not affect pending or running LSF Batch jobs. To add a batch queue to a cluster:

Step 1 Log in as the LSF administrator on any host in the cluster.

Step 2 Edit the `LSB_CONFDIR/cluster/configdir/lsb.queues` file to add the new queue definition. You can copy another queue definition from this file as a starting point; remember to change the `QUEUE_NAME` of the copied queue. Save the changes to `lsb.queues`. See *'The lsb.queues File' on page 208* for a complete description of LSF Batch queue configuration.

Step 3 Run the command `badmin ckconfig` to check the new queue definition. If any errors are reported, fix the problem and check the configuration again. See *'Overview of LSF Configuration Files' on page 50* for an example of normal output from `badmin ckconfig`.

Step 4 When the configuration files are ready, run `badmin reconfig`. The master batch daemon (`mbatchd`) is unavailable for approximately one minute while it reconfigures. Pending and running jobs are not affected.

Removing a Batch Queue

Before removing a queue, you should make sure there are no jobs in that queue. If you remove a queue that has jobs in it, the jobs are temporarily moved to a *lost and found* queue. Jobs in the lost and found queue remain pending until the user or the LSF administrator uses the `bswitch` command to switch the jobs into regular queues. Jobs in other queues are not affected.

In this example, move all pending and running jobs in the *night* queue to the *idle* queue, and then delete the *night* queue.

Step 1 Log in as the LSF administrator on any host in the cluster.

Step 2 Close the queue to prevent any new jobs from being submitted:

```
% badmin qclose night
Queue <night> is closed
```

Step 3 Move all pending and running jobs into another queue. The `bswitch -q night` argument chooses jobs from the *night* queue, and the job ID number 0 specifies that all jobs should be switched:

```
% bjobs -u all -q night
```

JOBID	USER	STAT	QUEUE	FROM_HOST	EXEC_HOST	JOB_NAME	SUBMIT_TIME
5308	user5	RUN	night	hostA	hostD	sleep 500	Nov 21 18:16
5310	user5	PEND	night	hostA		sleep 500	Nov 21 18:17

```
% bswitch -q night idle 0
```

```
Job <5308> is switched to queue <idle>
```

```
Job <5310> is switched to queue <idle>
```

Step 4 Edit the `LSB_CONFDIR/cluster/configdir/lsb.queues` file. Remove (or comment out) the definition for the queue being removed. Save the changes.

Step 5 Run the command `badmin reconfig`. If any problems are reported, fix them and run `badmin reconfig` again. The batch system is unavailable for about one minute while the system rereads the configuration.

Validating Job Submissions

A user's job can be rejected at submission time if the submission parameters cannot be validated. Sites can implement their own policy to determine valid values or combinations of submission parameters. The validation checking is performed by an external submission program (`esub`) located in `LSF_SERVERDIR` (see '*External Submission and Execution Executables*' on page 42).

The `esub` is invoked at job submission and modification time. It is also invoked when a checkpointed job is restarted. In each of these cases the user is allowed to specify parameters that affect the scheduling or execution of the job. To validate these parameters, the `esub` is invoked with two environment variables set:

- **LSB_SUB_PARM_FILE** is the full pathname of a file containing the submission, modification, or restart parameters specified by the user. The file consists of a set of name-value pairs of the form "option_name=value" with the option names listed in the following table. The `esub` can read this file.

3 Managing LSF Batch

- **LSB_SUB_ABORT_VALUE** is the exit code used by the `esub` whenever an operation is aborted as a result of the parameters encountered by the `esub` when the `LSB_SUB_PARM_FILE` file is read.

The **LSB_SUB_PARM_FILE** Option Names are shown below:

`LSB_SUB_JOB_NAME`

the specified job name

`LSB_SUB_QUEUE`

value is the specified queue name

`LSB_SUB_IN_FILE`

the specified standard input file name

`LSB_SUB_OUT_FILE`

the specified standard output file name

`LSB_SUB_ERR_FILE`

the specified standard error file name

`LSB_SUB_EXCLUSIVE`

"Y" specifies exclusive execution

`LSB_SUB_NOTIFY_END`

ends

`LSB_SUB_NOTIFY_BEGIN`

"Y" specifies email notification when job begins

`LSB_SUB_USER_GROUP`

the specified user group name

`LSB_SUB_CHKPNT_PERIOD`

the specified checkpoint period

`LSB_SUB_CHKPNT_DIR`

the specified checkpoint directory

LSB_SUB_RESTART_FORCE
"Y" specifies forced restart job

LSB_SUB_RESTART
"Y" specifies a restart job.

LSB_SUB_RERUNNABLE
"Y" specifies a rerunnable job.

LSB_SUB_WINDOW_SIG
the specified window signal number

LSB_SUB_HOST_SPEC
the specified hostspec

LSB_SUB_DEPEND_COND
the specified dependency condition

LSB_SUB_RES_REQ
the specified resource requirement string

LSB_SUB_PRE_EXEC
the specified pre-execution command

LSB_SUB_LOGIN_SHELL
the specified login shell

LSB_SUB_MAIL_USER
the specified user for sending email

LSB_SUB_MODIFY
"Y" specifies a modification request

LSB_SUB_MODIFY_ONCE
"Y" specifies a modification-once request

LSB_SUB_PROJECT_NAME
the specified project name

3 Managing LSF Batch

LSB_SUB_INTERACTIVE

"Y" specifies an interactive job

LSB_SUB_PTY

"Y" specifies an interactive job with PTY support

LSB_SUB_PTY_SHELL

"Y" specifies an interactive job with PTY shell support

LSB_SUB_TIME_EVENT

the time event expression

LSB_SUB_HOSTS

the list of execution host names

LSB_SUB_NUM_PROCESSORS

the minimum number of processors requested

LSB_SUB_MAX_NUM_PROCESSORS

the maximum number of processors requested

LSB_SUB_BEGIN_TIME

the begin time, in seconds since 00:00:00 GMT, Jan. 1, 1970

LSB_SUB_TERM_TIME

the termination time, in seconds since 00:00:00 GMT, Jan. 1, 1970

LSB_SUB_OTHER_FILES

always "SUB_RESET" if defined to indicate a bmod is being performed to reset the number of files to be transferred

LSB_SUB_OTHER_FILES_nn

nn is an index number indicating the particular file transfer

value is the specified file transfer expression; for example, for `bsub -f "a > b" -f "c < d"`, the following would be defined:

```
LSB_SUB_OTHER_FILES_0="a > b"
```

```
LSB_SUB_OTHER_FILES_1="c < d"
```

LSB_SUB_EXCEPTION
the specified exception condition

LSB_SUB_RLIMIT_CPU
the specified cpu limit

LSB_SUB_RLIMIT_FSIZE
the specified file limit

LSB_SUB_RLIMIT_DATA
the specified data size limit

LSB_SUB_RLIMIT_STACK
the specified stack size limit

LSB_SUB_RLIMIT_CORE
the specified core file size limit

LSB_SUB_RLIMIT_RSS
the specified resident size limit

LSB_SUB_RLIMIT_RUN
the specified wall clock run limit

Any messages that need to be provided to the user should be directed to the standard error stream and not the standard output stream.

One use of this feature is to support project-based accounting. The user can request that the resources used by a job be charged to a particular project. Projects are defined outside of the LSF configuration files, so LSF will accept any arbitrary string for a project name. In order to ensure that only valid projects are entered and the user is eligible to charge to that project, an `esub` can be written.

The following is an example of an external submission program written in UNIX bare shell to do this.

3 Managing LSF Batch

```
#!/bin/sh
. $LSB_SUB_PARM_FILE

# Redirect stderr to stdout so echo can be used for error messages
exec 1>&2

# Check valid projects
if [ $LSB_PROJECT -ne "proj1" -o $LSB_PROJECT -ne "proj2" ] then
    echo "Invalid project name specified"
    exit $LSB_SUB_ABORT_VALUE
fi

USER=`whoami`
if [ $LSB_PROJECT -eq "proj1" ]; then
    # Only user1 and user2 can charge to proj1
    if [ $USER -ne "user1" -a $USER -ne "user2" ]; then
        echo "You are not allowed to charge to this project"
        exit $LSB_SUB_ABORT_VALUE
    fi
fi
```

Controlling LSF Batch Jobs

The LSF administrator can control batch jobs belonging to any user. Other users can control only their own jobs. Jobs can be suspended, resumed, killed, and moved within and between queues.

Moving Jobs — `bswitch`, `btop`, and `bbot`

The `bswitch` command moves pending and running jobs from queue to queue. The `btop` and `bbot` commands change the dispatching order of pending jobs within a queue. The LSF administrator can move any job. Other users can move only their own jobs.

The `btop` and `bbot` commands do not allow users to move their own jobs ahead of those submitted by other users. Only the execution order of the user's own jobs is changed. The LSF administrator can move one user's job ahead of another user's. The

`btop`, `bbot`, and `bswitch` commands are described in the *LSF Batch User's Guide* and in the `btop(1)` and `bswitch(1)` manual pages.

Signalling Jobs — `bstop`, `bresume`, and `bkill`

The `bstop`, `bresume`, and `bkill` commands send signals to batch jobs. See the `kill(1)` manual page for a discussion of the signals on UNIX.

`bstop` sends `SIGSTOP` to sequential jobs and `SIGTSTP` to parallel jobs.

`bresume` sends a `SIGCONT`.

`bkill` sends the specified signal to the process group of the specified jobs. If the `-s` option is not present, the default operation of `bkill` is to send a `SIGKILL` signal to the specified jobs to kill these jobs. Twenty seconds before `SIGKILL` is sent, `SIGTERM` and `SIGINT` are sent to give the job a chance to catch the signals and clean up.

On Windows NT, 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.

Users are only allowed to send signals to their own jobs. The LSF administrator can send signals to any job. See the *LSF Batch User's Guide* and the manual pages for more information about these commands.

This example shows the use of the `bstop` and `bkill` commands:

```
% bstop 5310
Job <5310> is being stopped

% bjobs 5310
JOBID USER  STAT  QUEUE      FROM_HOST  EXEC_HOST  JOB_NAME    SUBMIT_TIME
5310  user5  PSUSP  night      hostA                sleep 500   Nov 21 18:17

% bkill 5310
Job <5310> is being terminated
```

3 Managing LSF Batch

```
% bjobs 5310
```

JOBID	USER	STAT	QUEUE	FROM_HOST	EXEC_HOST	JOB_NAME	SUBMIT_TIME
5310	user5	EXIT	night	hostA		sleep 500	Nov 21 18:17

Forcing Job Execution — `brun -f`

A pending batch job can be forced to run by using the `brun` command. This operation can only be performed by an LSF administrator. To force a job to run, you must specify the host on which that job will run. For parallel jobs, a list of hosts can be specified. The number of host names in the list must be at least equal to the minimum number of processors requested by the job. For example, the following command will force the sequential job 104 to run on `hostA`:

```
% brun -m hostA 104
```

The following command will force the parallel job 105 to run on `hostA`, `hostB`, `hostC`, and `hostD`.

```
% brun -m "hostA hostB hostC hostD" 105
```

If the job had requested more than 4 processors at a minimum, the request would have been rejected. If the number of hosts specified for a parallel job is greater than the maximum number of processors the job requested, the extra hosts are ignored.

When a job is forced to run, any other constraints associated with the job (such as resource requirements or dependency conditions) are ignored. Moreover, any scheduling policy (such as fairshare or job limit) specified in the batch configuration is also ignored. In this situation you might see some job slot limits, such as the maximum number of jobs that can run on a host, being violated. See *Job Slot Limits* on page 26 for details on job slot limits. However, after a job is forced to run, it can still be suspended due to the underlying queue's run window and threshold conditions and the job's execution hosts' threshold conditions. To override these so that a job can be run until completion, ignoring these load conditions, use the `-f` option. An example of a job forced to run until completion is shown below:

```
% brun -f -m hostA 124
```

Managing an LSF Cluster Using `xlsadmin`

Cluster Administrator (`xlsadmin`) is a graphical tool designed to assist you in the management of your LSF cluster. This tool allows you to perform the management tasks described in *Chapter 2, 'Managing LSF Base', on page 45* and preceding portion of *Chapter 3, 'Managing LSF Batch', on page 79*.

`xlsadmin` consists of the following operating modes: management and configuration.

Management mode provides the tools to:

- View base host, batch host and queue status. These tasks would otherwise be done using the `lshosts`, `lsload`, `bhosts`, and `bqueues` commands.
- Perform control tasks (i.e., start, stop, open, close, ...) on base hosts (LIM and RES), batch hosts and queues (`sbatchd`). These tasks would otherwise be done using the `lsadmin` and `badmin` commands.

Configuration mode provides the tools to:

- Add, delete, and modify the configuration of base hosts, batch host, batch queues, global objects, shared objects, and resources. These tasks would otherwise be done by manually editing LSF configuration files.
- Verify configuration changes and reconfigure the LSF Batch system. These tasks would otherwise be done using the `lsadmin reconfig` and `badmin reconfig` commands.

3 Managing LSF Batch

xlsadmin Management Mode

Figure 2 shows the xlsadmin Manage Base tab which displays all cluster hosts defined by LIM configuration. Figure 3 shows the xlsadmin Manage Batch tab which displays all the configured batch hosts and queues.

System messages and LSF command responses are displayed:

- UNIX** In the message area at the bottom of the xlsadmin window.
- NT** In the message dialog activated by choosing **View | Show Message Box...**

Figure 2. xlsadmin Manage Base Tab

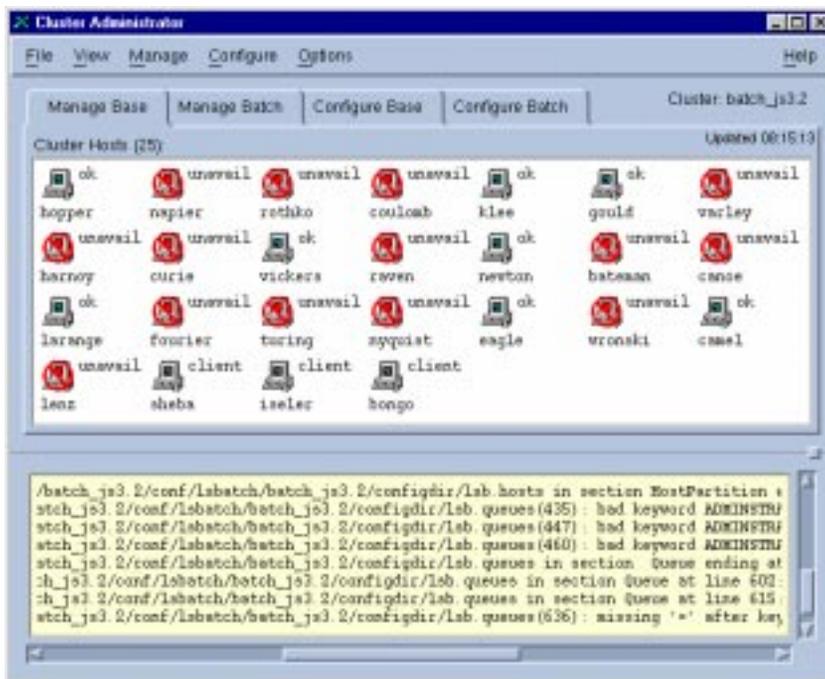


Figure 3. xlsadmin Manage Batch Tab

The screenshot shows the 'Cluster Administrator' window with the 'Manage Batch' tab selected. It displays two tables: 'Batch Hosts' and 'Batch Queues'.

Host Name	Status	NODES	MSK	SLD
couloirb	ok	0	unlimited	unlimited
gould	ok	0	unlimited	unlimited
larange	ok	0	unlimited	unlimited
canoe	unavail	0	unlimited	unlimited
eagle	ok	0	unlimited	unlimited
idea	ok	1	unlimited	unlimited
cure	unavail	0	unlimited	unlimited
huing	ok	0	unlimited	unlimited
wronski	unavail	0	unlimited	unlimited
canal	ok	0	unlimited	unlimited
newton	closed	0	unlimited	unlimited

Queue Name	Status	PRIO	MSK	NODES	PEND
avang	Open/Active	70	unlimited	0	0
owners	Open/Active	43	unlimited	0	0
priority	Open/Active	43	unlimited	0	0
dkr_q	Open/Active	43	unlimited	0	0
right	Open/Inactive	40	unlimited	0	0
exclusive	Open/Active	40	unlimited	0	0
q40	Open/Active	40	2	0	0
JobAccept_0	Open/Active	35	unlimited	0	0
JobAccept_4	Open/Active	35	unlimited	0	0
short	Open/Active	35	unlimited	0	0
BackF StkR	Open/Active	35	5	0	0

Updated: 14:47:55

On the Manage Base and Manage Batch tabs, double-click any item to display status dialogs.

Right-click any item to perform the following control tasks:

- **Base host:** start, restart, lock, unlock and shutdown LIM, and start, restart, and shutdown RES.
- **Batch host:** start, restart, and shutdown sbatchd, and open and close host.
- **Queue:** activate, deactivate, open, and close.

3 Managing LSF Batch

xlsadmin Configuration Mode

Figure 4 shows the xlsadmin Configure Base tab. This tab displays the base hosts defined by LIM configuration and provides tools to add, modify, and delete base hosts and global objects (host types, host models, and resources).

Figure 4. xlsadmin Configure Base Tab

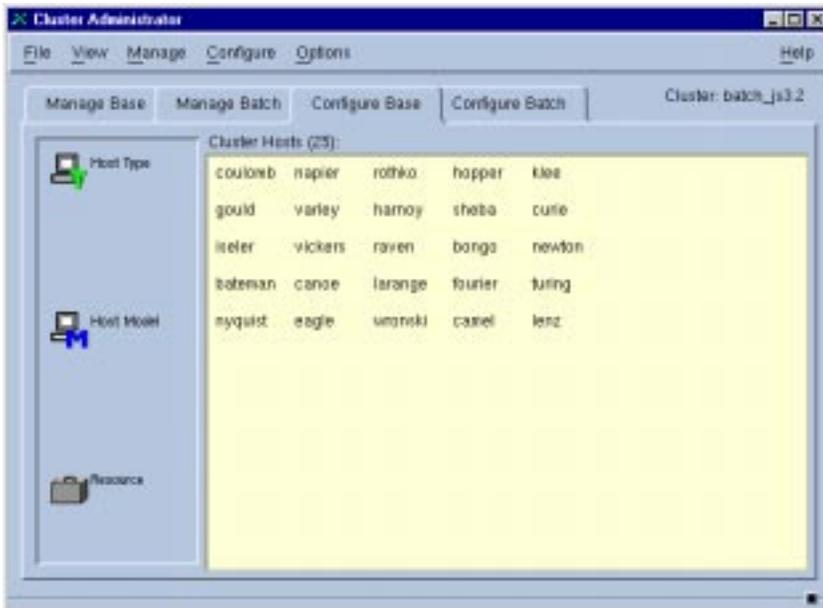
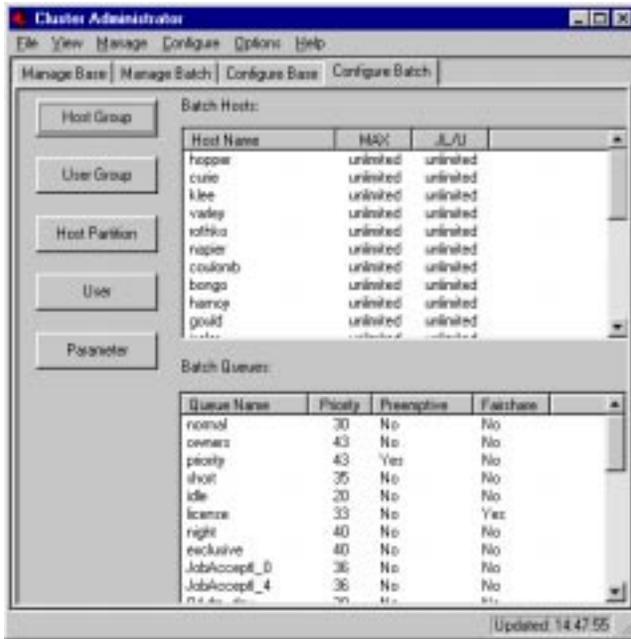


Figure 5 shows the `xlsadmin` Configure Batch tab. This tab displays the configured batch hosts and queues and provides the tools to add, modify, and delete batch hosts, queues, host groups, user groups, host partitions, and batch parameters.

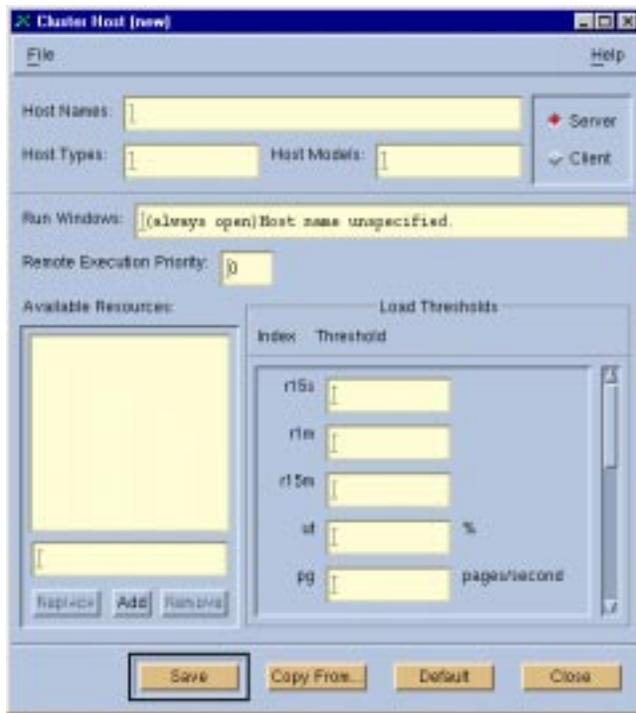
Figure 5. `xlsadmin` Configure Batch Tab



3 Managing LSF Batch

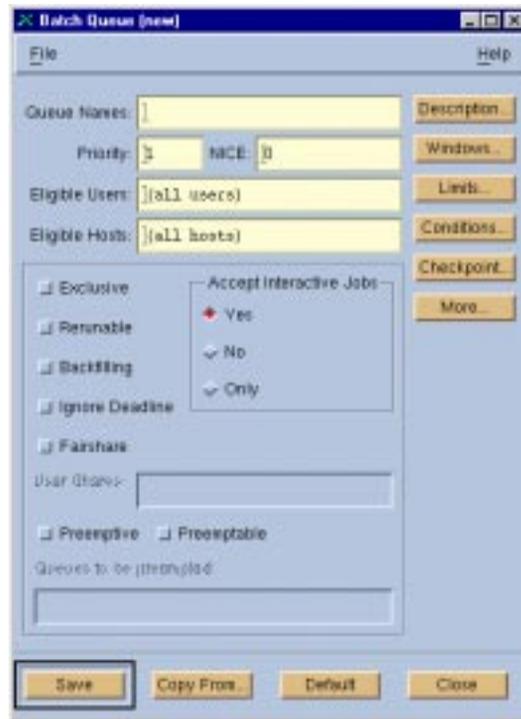
To add, modify and delete base hosts, batch hosts and queues use the right-click menu and choose the appropriate command. *Figure 6* shows the Cluster Host dialog used to edit and add base hosts, and *Figure 7* shows the Batch Queue dialog used to edit and add queues.

Figure 6. Cluster Host Dialog



To add, modify and delete host types, host models, resources (configured in `lsf.shared`), host groups, host partitions (configured in `lsb.hosts`), user groups (configured in `lsb.users`), and batch parameters (configured in `lsb.params`) choose the appropriate tool button. For example, *Figure 7* shows the Resources dialog used to edit, add, and delete resources.

Figure 7. Batch Queue Dialog



After making modifications to the cluster configuration, complete the following steps:

- Step 1** Choose **File | Check**. View the messages displayed in the message area and correct any errors.
- Step 2** Save all modifications by choosing **File | Save to Files...**
- Step 3** Reconfigure the LSF cluster using the modified configuration by choosing **File | Commit**.

3 Managing LSF Batch

4. Tuning LSF Batch

This chapter describes the operating concepts and maintenance tasks of the batch queuing system, LSF Batch. This chapter requires you to understand concepts from *'Managing LSF Base'* on page 45. The topics covered in this chapter are:

- tuning LSF Batch
- controlling interference via load conditions
- understanding suspended jobs
- controlling fairshare
- hierarchical fairshare
- understanding how fairshare works
- limits and windows
- controlling job execution
- using licensed software with LSF Batch
- sample LSF Batch configuration files

Tuning LSF Batch

Each batch job has its resource requirements. Batch server hosts that match the resource requirements are the *candidate hosts*. When the batch daemon wants to schedule a job, it first asks the LIM for the load index values of all the candidate hosts.

4 Tuning LSF Batch

The load values for each host are compared to the scheduling conditions. Jobs are only dispatched to a host if all load values are within the scheduling thresholds.

When a job is running on a host, the batch daemon periodically gets the load information for that host from the LIM. If the load values cause the suspending conditions to become true for that particular job, the batch daemon performs the `SUSPEND` action to the process group of that job. The batch daemon allows some time for changes to the system load to register before it considers suspending another job.

When a job is suspended, the batch daemon periodically checks the load on that host. If the load values cause the scheduling conditions to become true, the daemon performs the `RESUME` action to the process group of the suspended batch job.

The `SUSPEND` and `RESUME` actions are configurable as described in '*Configurable Job Control Actions*' on page 228.

LSF Batch has a wide variety of configuration options. This section describes only a few of the options to demonstrate the process. For complete details, see '*LSF Batch Configuration Reference*' on page 193. The algorithms used to schedule jobs and concepts involved are described in '*How LSF Batch Schedules Jobs*' on page 19.

Controlling Interference via Load Conditions

LSF is often used on systems that support both interactive and batch users. On one hand, users are often concerned that load sharing will overload their workstations and slow down their interactive tasks. On the other hand, some users want to dedicate some machines for critical batch jobs so that they have guaranteed resources. Even if all your workload is batch jobs, you still want to reduce resource contentions and operating system overhead to maximize the use of your resources.

Numerous parameters in LIM and LSF Batch configurations can be used to control your resource allocation and to avoid undesirable contention.

Since interferences are often reflected from the load indices, LSF Batch responds to load changes to avoid or reduce contentions. LSF Batch can take actions on jobs to reduce interference before or after jobs are started. These actions are triggered by different load conditions. Most of the conditions can be configured at both the queue level and at the host level. Conditions defined at the queue level apply to all hosts used by the queue, while conditions defined at the host level apply to all queues using the host.

- Scheduling conditions. These conditions, if met, trigger the start of more jobs. The scheduling conditions are defined in terms of load thresholds or resource requirements.

At the queue level, scheduling conditions are configured as either resource requirements or scheduling load thresholds, as described in *'The lsb.queues File' on page 208*. At the host level, the scheduling conditions are defined as scheduling load thresholds, as described in *'The lsb.hosts File' on page 202*.

- Suspending conditions. These conditions affect running jobs. When these conditions are met, a `SUSPEND` action is performed to a running job.

At the queue level, suspending conditions are defined as `STOP_COND` as described in *'The lsb.queues File' on page 208*, or as suspending load threshold as described in *'Load Thresholds' on page 216*. At the host level, suspending conditions are defined as stop load threshold as described in *'The lsb.hosts File' on page 202*.

- Resuming conditions. These conditions determine when a suspended job can be resumed. When these conditions are met, a `RESUME` action is performed on a suspended job.

At the queue level, resume conditions are defined as either `RESUME_COND`, or the scheduling load conditions if `RESUME_COND` is not defined.

To effectively reduce interference between jobs, correct load indices should be used properly. Below are examples of a few frequently used parameters.

Paging Rate (`pg`)

The paging rate (`pg`) load index relates strongly to the perceived interactive performance. If a host is paging applications to disk, the user interface feels very slow.

The paging rate is also a reflection of a shortage of physical memory. When an application is being paged in and out frequently, the system is spending a lot of time performing overhead, resulting in reduced performance.

The paging rate load index can be used as a threshold to either stop sending more jobs to the host, or to suspend an already running batch job so that interactive users will not be interfered.

4 Tuning LSF Batch

This parameter can be used in different configuration files to achieve different purposes. By defining paging rate threshold in `lsf.cluster.cluster`, the host will become busy from LIM's point of view; therefore, no more jobs will be advised by LIM to run on this host.

By including paging rate in LSF Batch queue or host scheduling conditions, batch jobs can be prevented from starting on machines with a heavy paging rate, or can be suspended or even killed if they are interfering with the interactive user on the console.

A batch job suspended due to `pg` threshold will not be resumed even if the resume conditions are met unless the machine is interactively idle for more than `PG_SUSP_IT` seconds, as described in *'Parameters' on page 193*.

Interactive Idle Time (`it`)

Strict control can be achieved using the idle time (`it`) index. This index measures the number of minutes since any interactive terminal activity. Interactive terminals include hard wired `tty`s, `rlogin` and `lslogin` sessions, and X shell windows such as `xterm`. On some hosts, LIM also detects mouse and keyboard activity.

This index is typically used to prevent batch jobs from interfering with interactive activities. By defining the suspending condition in LSF Batch queue as `'it==0 && pg >50'`, a batch job from this queue will be suspended if the machine is not interactively idle and paging rate is higher than 50 pages per second. Furthermore, by defining resuming condition as `'it>5 && pg <10'` in the queue, a suspended job from the queue will not resume unless it has been idle for at least five minutes and the paging rate is less than ten pages per second.

The `it` index is only non-zero if no interactive users are active. Setting the `it` threshold to five minutes allows a reasonable amount of think time for interactive users, while making the machine available for load sharing, if the users are logged in but absent.

For lower priority batch queues, it is appropriate to set an `it` scheduling threshold of ten minutes and suspending threshold of two minutes in the `lsb.queues` file. Jobs in these queues are suspended while the execution host is in use, and resume after the host has been idle for a longer period. For hosts where all batch jobs, no matter how important, should be suspended, set a per-host suspending threshold in the `lsb.hosts` file.

CPU Run Queue Length (`r15s`, `r1m`, `r15m`)

Running more than one CPU-bound process on a machine (or more than one process per CPU for multiprocessors) can reduce the total throughput because of operating system overhead, as well as interfering with interactive users. Some tasks such as compiling can create more than one CPU intensive task.

Batch queues should normally set CPU run queue scheduling thresholds below 1.0, so that hosts already running compute-bound jobs are left alone. LSF Batch scales the run queue thresholds for multiprocessor hosts by using the effective run queue lengths, so multiprocessors automatically run one job per processor in this case. For concept of effective run queue lengths, see `lsfintr(1)`.

For short to medium-length jobs, the `r1m` index should be used. For longer jobs, you might want to add an `r15m` threshold. An exception to this are high priority queues, where turnaround time is more important than total throughput. For high priority queues, an `r1m` scheduling threshold of 2.0 is appropriate.

CPU Utilization (`ut`)

The `ut` parameter measures the amount of CPU time being used. When all the CPU time on a host is in use, there is little to gain from sending another job to that host unless the host is much more powerful than others on the network. The `lsload` command reports `ut` in percent, but the configuration parameter in the `lsf.cluster.cluster` file and the LSF Batch configuration files is set as a fraction in the range from 0 to 1. A `ut` threshold of 0.9 prevents jobs from going to a host where the CPU does not have spare processing cycles.

If a host has very high `pg` but low `ut`, then it may be desirable to suspend some jobs to reduce the contention.

The commands `bhist` and `bjobs` are useful for tuning batch queues. `bhist` shows the execution history of batch jobs, including the time spent waiting in queues or suspended because of system load. `bjobs -p` shows why a job is pending.

Understanding Suspended Jobs

A batch job is suspended when the load level of the execution host causes the suspending condition to become true. The `bjobs -lp` command shows the reason why the job was suspended together with the scheduling parameters. Use `bhosts -l`

4 Tuning LSF Batch

to check the load levels on the host, and adjust the suspending conditions of the host or queue if necessary.

The `bhosts -l` gives the most recent load values used for the scheduling of jobs.

```
% bhosts -l hostB
```

```
HOST: hostB
```

STATUS	CPUF	JL/U	MAX	NJOBS	RUN	SSUSP	USUSP	RSV	DISPATCH_WINDOWS
ok	20.00	2	2	0	0	0	0	0	-

```
CURRENT LOAD USED FOR SCHEDULING:
```

	r15s	rlm	r15m	ut	pg	io	ls	t	tmp	swp	mem
Total	0.3	0.8	0.9	61%	3.8	72	26	0	6M	253M	297M
Reserved	0.0	0.0	0.0	0%	0.0	0	0	0	0M	0M	0M

```
LOAD THRESHOLD USED FOR SCHEDULING:
```

	r15s	rlm	r15m	ut	pg	io	ls	it	tmp	swp	mem
loadSched	-	-	-	-	-	-	-	-	-	-	-
loadStop	-	-	-	-	-	-	-	-	-	-	-

A '-' in the output indicates that the particular threshold is not defined. If no suspending threshold is configured for a load index, LSF Batch does not check the value of that load index when deciding whether to suspend jobs. Normally, the `swp` and `tmp` indices are not considered for suspending jobs, because suspending a job does not free up the space being used. However, if `swp` and `tmp` are specified by the `STOP_COND` parameter in your queue, these indices are considered for suspending jobs.

The load indices most commonly used for suspending conditions are the CPU run queue lengths, paging rate, and idle time. To give priority to interactive users, set the suspending threshold on `it` load index to a non-zero value. Batch jobs are stopped (within about 1.5 minutes) when any user is active, and resumed when the host has been idle for the time given in the `it` scheduling condition.

To tune the suspending threshold for paging rate, it is desirable to know the behaviour of your application. On an otherwise idle machine, check the paging rate using `lsload`, and then start your application. Watch the paging rate as the application runs. By subtracting the active paging rate from the idle paging rate, you get a number for the paging rate of your application. The suspending threshold should allow at least 1.5 times that amount. A job can be scheduled at any paging rate up to the scheduling

threshold, so the suspending threshold should be at least the scheduling threshold plus 1.5 times the application paging rate. This prevents the system from scheduling a job and then immediately suspending it because of its own paging.

The effective CPU run queue length condition should be configured like the paging rate. For CPU-intensive sequential jobs, the effective run queue length indices increase by approximately one for each job. For jobs that use more than one process, you should make some test runs to determine your job's effect on the run queue length indices. Again, the suspending threshold should be equal to at least the scheduling threshold plus 1.5 times the load for one job.

Suspending thresholds can also be used to enforce inter-queue priorities. For example, if you configure a low-priority queue with an `r1m` (1 minute CPU run queue length) scheduling threshold of 0.25 and an `r1m` suspending threshold of 1.75, this queue starts one job when the machine is idle. If the job is CPU intensive, it increases the run queue length from 0.25 to roughly 1.25. A high-priority queue configured with a scheduling threshold of 1.5 and an unlimited suspending threshold will send a second job to the same host, increasing the run queue to 2.25. This exceeds the suspending threshold for the low priority job, so it is stopped. The run queue length stays above 0.25 until the high priority job exits. After the high priority job exits the run queue index drops back to the idle level, so the low priority job is resumed.

Controlling Fairshare

By default, LSF Batch schedules user jobs according to the First-Come-First-Serve (FCFS) principle. If your sites have many users contending for limited resources, the FCFS policy is not enough. For example, a user could submit 1000 long jobs in one morning and occupy all the resources for a whole week, while other users's urgent jobs wait in queues.

LSF Batch provides fairshare scheduling to give you control on how resources should be shared by competing users. Fairshare can be configured so that LSF Batch can schedule jobs according to each user or user group's configured shares. When fairshare is configured, each user or user group is assigned a priority based on the following factors:

- configured share for the user or user group
- current number of job slots in use by the user

4 Tuning LSF Batch

- cumulative CPU time used by the user over the past configurable number of hours (controlled by the `HIST_HOURS` parameter in the `lsb.params` file)
- cumulative run time.

If a user or group has used less than their share of the processing resources, their pending jobs (if any) are scheduled first, jumping ahead of other jobs in the batch queues. The CPU times used for fairshare scheduling are not normalised for the host CPU speed factors.

The special user names `others` and `default` can also be assigned shares. The name `others` refers to all users not explicitly listed in the `USER_SHARES` parameter. The name `default` refers to each user not explicitly named in the `USER_SHARES` parameter. Note that `default` represents a single user name while `others` represents a user group name.

Fairshare affects job scheduling only if there are resource contentions among users such that users with more shares will run more jobs than users with less shares. If there is only one user having jobs to run, then fairshare has no effect on job scheduling.

Fairshare in LSF Batch can be configured at either queue level or host level. At queue level, the shares apply to all users who submit jobs to the queue and all hosts that are configured as hosts for the queue. It is possible that several queues share some hosts as servers, but each queue can have its own fairshare policy.

Queue level fairshare is defined using the keyword `FAIRSHARE`.

If you want strict resource allocation control on some hosts for all workload, configure fairshare at the host level. Host level fairshare is configured as a host partition. Host partition is a configuration option that allows a group of server hosts to be shared by users according to configured shares. In a host partition each user or group of users is assigned a share. The `bhpart` command displays the current cumulative CPU usage and scheduling priority for each user or group in a host partition.

Below are some examples of configuring fairshare at both queue level and host level. Details of the configuration syntax are described in *'Host Partitions'* on page 206 and *'Scheduling Policy'* on page 221.

Note

Do not define fairshare at both the host and the queue level if the queue uses some or all hosts belonging to the host partition, because this results in policy conflicts. Doing so will result in undefined scheduling behaviour.

Favouring Critical Users

If you have a queue that is shared by critical users and non-critical users, you can configure fairshare so that as long as there are jobs from key users waiting for resource, non-critical users' jobs will not be dispatched.

First you can define a user group `key_users` in `lsb.users` file. You can then define your queue such that `FAIRSHARE` is defined:

```
Begin Queue
QUEUE_NAME = production
FAIRSHARE = USER_SHARES[[key_users@, 2000] [others, 1]]
...
End Queue
```

By this configuration, `key_users` each have 2000 shares, while other users together have only 1 share. This makes it virtually impossible for other users' jobs to get dispatched unless no user in the `key_users` group has jobs waiting to run.

Note that a user group followed by an '@' refers to each user in that group, as you could otherwise configure by listing every user separately, each having 2000 shares. This also defines equal shares among the `key_users`. If '@' is not present, then all users in the user group share the same share and there will be no fairshare among them.

You can also use host partition to achieve similar results if you want the same fairshare policy to apply to jobs from all queues.

Sharing Hosts Between Two Groups

Suppose two departments contributed to the purchase of a large system. The engineering department contributed 70 percent of the cost, and the accounting department 30 percent. Each department wants to get (roughly) their money's worth from the system.

4 Tuning LSF Batch

You would configure two user groups in the `lsb.users` file, one listing all the users in the engineering group, and one listing all the members in the accounting group:

```
Begin UserGroup
Group_Name    Group_Member
eng_users     (user6 user4)
acct_users    (user2 user5)
End UserGroup
```

You would then configure a host partition for the host, listing the appropriate shares:

```
Begin HostPartition
HPART_NAME = big_servers
HOSTS = hostH
USER_SHARES = [eng_users, 7] [acct_users, 3]
End HostPartition
```

Note the difference in defining `USER_SHARES` in a queue and in a host partition. Alternatively, the shares can be configured for each member of a user group by appending an '@' to the group name:

```
USER_SHARES = [eng_users@, 7] [acct_users@, 3]
```

If a user is configured to belong to two user groups, the user can specify which group the job belongs to with the `-P` option to the `bsub` command.

Similarly, you can define the same policy at the queue level if you want to enforce this policy only within a queue.

Round-Robin Scheduling

Round-robin scheduling balances the resource usage between users by running one job from each user in turn, independent of what order the jobs arrived in. This can be configured by defining an equal share for everybody. For example:

```
Begin HostPartition
HPART_NAME = even_share
HOSTS = all
USER_SHARES = [default, 1]
End HostPartition
```

Hierarchical Fairshare

For both queues and host partitions, the specification of how resources are allocated to users can be performed in a hierarchical manner. Groups of users can collectively be allocated a share, and that share can be further divided and given to subgroups, resulting in a share tree. For a discussion of the terminology associated with hierarchical fairsharing, see *'Hierarchical Fairshare'* on page 60 of the *LSF Batch User's Guide*.

Configuring Hierarchical Fairshare

There are two steps in configuring hierarchical fairshare:

- Define a share tree by defining a hierarchical user group in file `lsb.users`
- Reference the share tree in the `USER_SHARES` definition of the queue or host partition.

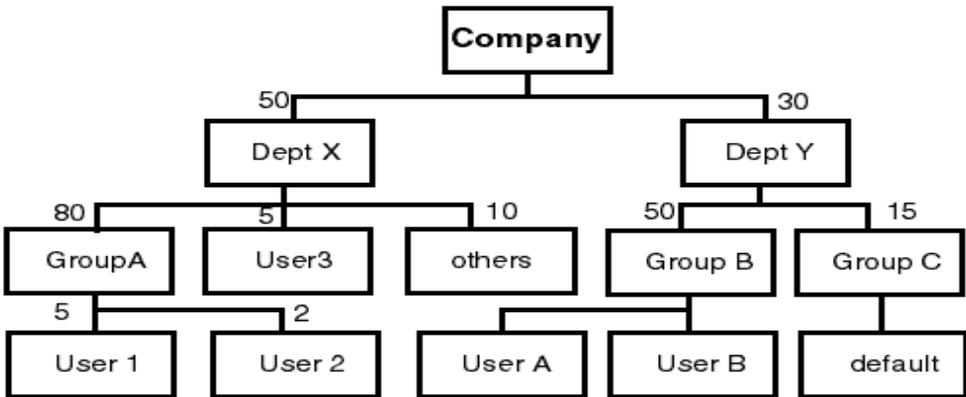
The following example shows how you can configure a share tree in the `lsb.users` file. User groups must be defined in the share tree before they can be used (in the `GROUP_MEMBER` column) to define other groups. The `USER_SHARES` column describes how the shares are distributed in a hierarchical manner.

```
Begin UserGroup
GROUP_NAME  GROUP_MEMBER          USER_SHARES
GroupA      (User1 User2)           ([User1, 5] [User2, 2])
GroupB      (UserA UserB)         ( )
GroupC      (UserC UserD UserE UserF) ([default, 1])
DeptX       (GroupA User3 User4 User5) ([GroupA, 80] [User3, 5] [others, 10])
DeptY       (GroupB GroupC)       ([GroupB, 50] [GroupC, 15])
Company     (DeptX DeptY)         ([DeptX, 50] [DeptY, 30])
End UserGroup
```

4 Tuning LSF Batch

The share distribution tree described by the preceding configuration is shown below.

Figure 8. Example Share Tree



There are a few special cases in the above tree that should be noted. The keyword "others" is used to refer to a special group representing all other users in GroupA that are not explicitly listed in the share allocation. For example, DeptX subdivides its shares among GroupA, User3 and "others".

In the above example, there is no definition of how users in group "others" should divide the shares. Also note that there is no specification of how users in groupB should subdivide the shares. If share distribution is not defined for a group, all members of the group collectively own the shares. In this case group members compete for resources allocated to the group on a First-Come-First-Serve (FCFS) basis.

To implement equal share at the group level, you should define `USER_SHARES` for the group as "[default, 1]", as is the case with GroupC in the above example.

The hierarchical shares defined in `lsb.users` file have no effect unless the group names are referenced in a share provider's `USER_SHARES` definition.

To associate the share tree defined by the above in a share provider (queue or host partition) simply use the group in the `USER_SHARES` definition.

The following example shows how a host partition might use the share tree “company” in its definition:

```
Begin HostPartition
HPART_NAME = hpartest
HOSTS = all
USER_SHARES = ([company, 1])
End HostPartition
```

The `USER_SHARES` parameter in the host partition definition references the top-level group of the share tree. Each share provider will maintain a copy of the share tree and adjust the priority of users based on the resource consumption of jobs using the provider. This might result in, for example, a user having a low priority in one fairshare queue and a high priority in another queue, even though the static shares they have been allocated are the same.

If hierarchical fairshare is not required, the `USER_SHARES` parameter in the `UserGroup` section of the `lsb.users` file can be omitted and the `USER_SHARE` parameter in the queue or host partition can directly list the shares. In this case, the share tree is essentially flat, and the share assigned to any group cannot be further divided.

Understanding How Fairshare Works

LSF Batch uses an account to maintain information about shares and resource consumption of every user or user group. Each account keeps the following information:

- The static share assigned to the user or user group (`u_share`)
- Current number of job slots (both reserved and started) in use by the user or user group (`run_j`)
- The cumulative CPU time used by the user or user group in the past `HIST_HOURS` hours (`cpu_t`)
- The cumulative run time of current jobs submitted by the user or user group (`run_t`).

4 Tuning LSF Batch

LSF Batch uses a decay factor in calculating the cumulative CPU time `cpu_t`. This decay factor scales the CPU time used by jobs so that recently used CPU time is weighted more heavily than CPU time used in the distant past. The decay factor is set such that one hour of CPU time used recently is decayed to 0.1 hours after `HIST_HOURS` hours. See *‘The lsb.params File’ on page 193* for the definition of `HIST_HOURS`.

A dynamic priority is calculated for each account according to the following formula:

```
priority = u_share  
          / (0.01 + cpu_t*CPU_TIME_FACTOR + run_t*RUN_TIME_FACTOR + run_j*RUN_JOB_FACTOR)
```

where `CPU_TIME_FACTOR`, `RUN_TIME_FACTOR`, and `RUN_JOB_FACTOR` are system-wide configuration parameters defined in `lsb.params` file. See *‘The lsb.params File’ on page 193* for a description and default values for these parameters. These parameters allow for customization of the fairshare formula to ignore or give greater weight to certain terms. For example, if you want to implement static fairshare so that priority is determined by shares only, then you can set all factors as 0.

Dynamic priorities are recalculated whenever a variable in the above formula is changed.

Job Dispatching According to Fairshare

LSF Batch dispatches jobs according to their dynamic priorities. If fairshare is defined at the queue level, the priorities are local to each queue. Among queues, the queue priorities decide which queue should be scanned first. If fairshare is defined at host level through a host partition, then the priorities of users are global across all queues that use hosts in the host partitions to run jobs. In this case, queue priority has no effect because the order is determined by users’ current priorities with regard to the host partition.

Whenever a host becomes available to run a job, LSF Batch tries to dispatch a job of the user with the highest dynamic priority. As soon as a job is dispatched, the user’s `run_j` gets updated and thus the priority gets lowered according to the above formula. In the case of hierarchical fairshare, LSF Batch scans the share tree from the top level down to find out which user’s job to run next. For example, with the share tree shown by *Figure 8*, LSF Batch first decides which department has the highest dynamic priority, then further decides which group has the highest priority. After selecting the highest priority group, a user with the highest priority within the group will be selected. If this

user has a job to run, the job will be dispatched, else the user with the next highest priority will be considered, and so on.

Suppose *User1* is chosen and the job has been started; the priorities of *User1*, *GroupA*, and *DeptX* are immediately updated to reflect the change of variable `run_j` at all levels.

In some special cases, a user could belong to two or more groups simultaneously. This is the case when a user works for several groups at the same time. Thus it is possible to define a share tree with one user appearing multiple times in the same share tree. In this case, the user's priority is determined by the highest priority node the user belongs to. To override this behaviour, a user can use the "-G" option of the `bsub` to advise LSF Batch which user group this user should belong to when dispatching this job.

Limits and Windows

Although LSF Batch makes it easier for users to access all resources of your client, real life constraints require that certain resources be controlled such that users are not stepping on one another. LSF Batch provides ways for you as an administrator to enforce controls in different ways.

Dispatch and Run Windows

The concept of dispatch and run windows for LSF Batch are described in *'How LSF Batch Schedules Jobs'* on page 19.

This can be achieved by configuring dispatch windows for the host in the `lsb.hosts` files, and run windows and dispatch windows for queues in `lsb.queues` file.

Dispatch windows in `lsb.hosts` file cause batch server hosts to be closed unless the current time is inside the time windows. When a host is closed by a time window, no new jobs will be sent to it, but the existing jobs running on it will remain running. Details about this parameter are described in *'Host Section'* on page 202.

Dispatch and run windows defined in `lsb.queues` limit when a queue can dispatch new jobs and when jobs from a queue are allowed to run. A run window differs from

4 Tuning LSF Batch

a dispatch window in that when a run window is closed, jobs that are already running will be suspended instead of remain running. Details of these two parameters are described in *'The lsb.queues File' on page 208*.

Controlling Job Slot Limits

By defining different job slot limits to hosts, queues, and users, you can control batch job processing capacity for your cluster, hosts, and users. For example, by limiting maximum job slot for each of your hosts, you can make sure that your system operates at optimal performance. By defining a job slot limit for some users, you can prevent some users from using up all the job slots in the system at one time. There are a variety of job slot limits that can be used for very different purposes. See *'Job Slot Limits' on page 26* for more concepts and descriptions of job slot limits. Configuration parameters for job slot limits are described in *'LSF Batch Configuration Reference' on page 193*.

Resource Limits

Resource limits control how much resource can be consumed by jobs. By defining such limits, the cluster administrator can have better control of resource usage. For example, by defining a high priority short queue, you can allow short jobs to be scheduled earlier than long jobs. To prevent some users from submitting long jobs to this short queue, you can set CPU limit for the queue so that no jobs submitted from the queue can run for longer than that limit.

Details of resource limit configuration are described in *'Resource Limits' on page 217*.

Reservation Based Scheduling

Most of the Batch policies discussed above support competition based scheduling; that is, users competing for resources on a dynamic basis. It is sometimes desirable to have reservation based scheduling so that people can predict the timing of their jobs.

Resource Reservation

The concept of resource reservation is discussed in *'Resource Reservation' on page 39*.

The resource reservation feature at the queue level allows the cluster administrator to specify the amount of resources the system should reserve for jobs in the queue. It also serves as the upper limits of resource reservation if a user also specifies it when submitting a job.

The resource reservation requirement can be configured at the queue level as part of the queue level resource requirements. For example:

```
Begin Queue
.
RES_REQ = select[type==any] rusage[swap=100:mem=40:duration=60]
.
End Queue
```

will allow a job to be scheduled on any host that the queue is configured to use and will reserve 100 megabytes of swap and 40 megabytes of memory for a duration of 60 minutes. See *'Queue-Level Resource Requirement' on page 213* for detailed configuration syntax for this parameter.

Processor Reservation and Backfilling

The concepts of processor reservation and backfilling were described in *'Processor Reservation' on page 39*. You might want to configure processor reservation if your cluster has a lot of sequential jobs that compete for resources with parallel jobs.

Parallel jobs requiring a large number of processors can often not be started if there are many lower priority sequential jobs in the system. There might not be enough resources at any one instant to satisfy a large parallel job, but there might be enough to allow a sequential job to be started. With the processor reservation feature the problem of starvation of parallel jobs can be reduced.

A host can have multiple 'slots' available for the execution of jobs. The number of slots can be independent of the number of processors and each queue can have its own notion of the number of execution slots available on each host. The number of execution slots on each host is controlled by the `PJOB_LIMIT` and `HJOB_LIMIT` parameters defined in `lsb.queues` file. For details of these parameters defined in `lsb.queues` file, see *'The lsb.queues File' on page 208*. When attempting to schedule parallel jobs requiring N processors (as specified via `bsub -n`), the system will attempt

4 Tuning LSF Batch

to find N execution slots across all eligible hosts. It ensures that each job never receives more slots than there are physical processors on any individual host.

When a parallel job cannot be dispatched because there are not enough execution slots to satisfy its minimum processor requirements, the currently available slots will be reserved for the job. These reserved job slots are accumulated until there are enough available to start the job. When a slot is reserved for a job it is unavailable to any other job.

While processors are being reserved by a parallel job, they cannot be used by other jobs. However, there are situations where the system can determine that the job reserving the processors cannot start before a certain time. In this case it makes sense to run another job that is short enough to fit into the time slot during which the processors are reserved but not used. This notion is termed *backfilling*. Short jobs are said to backfill processors reserved for large jobs. Backfilling requires that users specify how long each job will run so that LSF Batch can estimate when it will start and complete. Backfilling, together with processor reservation, allows large parallel jobs to run while not underutilizing resources.

For the backfill policy to work effectively, each job should have a run limit specified (via `-W bsub` option). In order to enforce that users should specify this option, the external submission executable, `esub`, can be used. See ‘*Validating Job Submissions*’ on page 91.

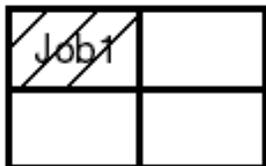
When backfilling is enabled, the system will compute the estimated start time for each job based on the run limits of the currently started jobs. A given job (*jobA*) can backfill the reserved processors of another job (*jobB*) if there is sufficient time for *jobA* to complete, based on its run limit, before the estimated start time of *jobB*.

As an example, consider the sequence of events depicted in the *Figure 9. ‘Example of Backfilling’* on page 125. In this scenario, assume the cluster consists of a 4-CPU multiprocessor host. A sequential job (*job1*) with a run limit of two hours is submitted to a high priority queue and gets started at 8:00 am (figure (a)). Shortly afterwards, a parallel job (*job2*) requiring all four CPUs is submitted. It cannot start right away because of *job1*, so it reserves the remaining three processors (figure (b)). At 8:30 am, another parallel job (*job3*) is submitted requiring only two processors and with a run limit of one hour. Since *job2* cannot start until 10:00am (when *job1* finishes), its reserved processors can be backfilled by *job3* (figure (c)). Therefore *job3* can complete before *job2*’s start time, making use of the idle processors. If *job3*’s run limit was three hours,

for example, it would not be able to backfill *job2*'s reserved slots. *Job 3* will finish at 9:30am and *job1* at 10:00am, allowing *job2* to start shortly after 10:00am.

The estimated start time of a job can be displayed using the `bjobs -l` command or by viewing the detailed information about the job through `xlsbatch`.

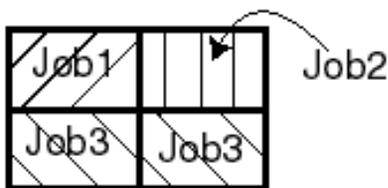
Figure 9. Example of Backfilling



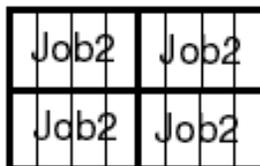
(a) Job1 started at 8:00 am.
Will finish at 10:00 am.



(b) Job2, submitted but can't start
since it needs 4 processors.
Remaining 3 reserved by Job2.



(c) At 8:30 am Job3 submitted.
Job3 backfills Job2.



(d) At 10:00 am, Job2 starts.

See 'Processor Reservation for Parallel Jobs' on page 211 and 'Backfill Scheduling' on page 211 for configuration options for this feature.

Controlling Job Execution

Understanding Job Execution Environment

When LSF Batch runs your jobs, it tries to make it as transparent to the user as possible. By default, the execution environment is maintained to be as close to the current environment as possible. LSF Batch will copy the environment from the submission host to the execution host. It also sets the `umask` and the current working directory.

Since a network can be heterogeneous, it is often impossible or undesirable to reproduce the submission host's execution environment on the execution host. For example, if home directory is not shared between submission and execution host, LSF Batch runs the job in the `/tmp` on the execution host. If the `DISPLAY` environment variable is something like `'Unix:0.0'`, or `':0.0'`, then it must be processed before using on the execution host. These are automatically handled by LSF Batch.

Users can change the default behaviour by using a job starter, or by using the `'-L'` option of the `bsub` command to change the default execution environment. See *'Queue-Level Job Starters'* on page 129 for detailed information on using a job starter at the queue level.

For resource control purpose, LSF Batch also changes some of the execution environment of jobs. These include nice values, resource limits, or any other environment by configuring a job starter.

In addition to environment variables inherited from the user, LSF Batch also sets a few more environment variables for batch jobs. These are:

- `LSB_JOBID`: Batch job ID assigned by LSF Batch.
- `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 `chkpntdir/jobId`, 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 belongs to.
 - `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 `-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.
 - By default, LSF transfers environment variables from the submission to the execution host. However, some environment variables do not make sense when transferred. When submitting a job from an NT to a UNIX machine, the `-L` option of `bsub` can be used to reinitialize the environment variables. If submitting a job from a UNIX machine to an NT machine, you can set the environment variables explicitly in your job script. Alternatively, the Job Starter feature can be used to reset the environment variables before starting the job.

LSF automatically resets the `PATH` on the execution host if the submission host is of a different type. If the submission host is NT and the execution host is UNIX, the `PATH` variable is set to `/bin:/usr/bin:/sbin:/usr/sbin` and `LSF_BINDIR` (if defined in `lsf.conf`) is appended to it. If the submission host is UNIX and the

4 Tuning LSF Batch

execution host is NT, the `PATH` variable is set to the system `PATH` variable with `LSF_BINDIR` appended to it. LSF looks for the presence of the `WINDIR` variable in the job's environment to determine whether the job was submitted from an NT or UNIX host. If `WINDIR` is present, it is assumed that the submission host was NT; otherwise, the submission host is assumed to be a UNIX machine.

NT

Environment Variable Handling

LSF transfers most environment variables between submission and execution hosts. The following environment variables are overridden based on the values on the execution host:

```
COMPSPEC  
COMPUTERNAME  
NTRESKIT  
OS2LIBPATH  
PROCESSOR_ARCHITECTURE  
PROCESSOR_LEVEL  
SYSTEMDRIVE  
SYSTEMROOT  
WINDIR
```

These must be defined as system environment variables on the execution host.

If the `WINDIR` on the submission and execution host are different, then the system `PATH` variable on the execution host is used instead of that from the submission host.

Avoid using drive names in environment variables (especially the `%PATH` variable) for drives that are connected over the network. It is preferable to use the UNC form of the path. This is because drive maps are shared between all users logged on to a particular machine. For example, if an interactive user has drive `F:` mapped to `\\serverX\share`, then any batch job will also see drive `F:` mapped to `\\serverX\share`. However, drive `F:` might have been mapped to a different share on the submission host of the job.

The Job Starter feature can be used to perform more site-specific handling of environment variables. See *'Job Starters' on page 16* for more details.

NICE Value

Many LSF tools use LSF Remote Execution Server (RES) to run jobs such as `lsrun`, `lsmake`, `lstcsh`, and `lsgrun`. You can control the execution priority of jobs started via RES by modifying your LIM configuration file `lsf.cluster.cluster`. This can be done by defining the `REXPRI` parameter for individual hosts. See *'Descriptive Fields' on page 182* for details of this parameter.

LSF Batch jobs can be run with a nice value as defined in your `lsb.queues` file. Each queue can have a different nice value. See *'NICE = integer' on page 209* for details of this parameter.

Pre-execution and Post-execution commands

Your batch jobs can be accompanied with a pre-execution and a post-execution command. This can be used for many purposes. For example, you can use these commands to create or delete scratch directories, or check for necessary conditions before running the real job. Details of these concepts are described in *'Pre- and Post-execution Commands' on page 36*.

The pre-execution and post-execution commands can be configured at the queue level as described in *'Queue-Level Pre-/Post-Execution Commands' on page 224*.

Queue-Level Job Starters

Some jobs have to be started in a particular environment, or require some type of setup to be performed before they are executed. In a shell environment, this situation is often handled by writing such preliminary procedures into a file that itself contains a call to start the desired job. This is referred to as a *wrapper*.

If users need to submit batch jobs that require this type of preliminary setup, LSF provides a *job starter* function at the queue level. A queue-level job starter allows you to specify an executable that will perform any necessary setup beforehand. One typical use of this feature is to customize LSF for use with Atria ClearCase environment (see *'Support for Atria ClearCase' on page 275*).

4 Tuning LSF Batch

A queue-level job starter is specified in the queue definition (in the `lsb.queues` file) using the `JOB_STARTER` parameter. When a job starter is set up in this way, all jobs executed from this queue will be executed via the job starter (i.e., called by the specified job starter process rather than initiated by the batch daemon process). For example, the following might be defined in a queue:

```
Begin Queue
.
JOB_STARTER = xterm -e
.
End Queue
```

In this case, all jobs submitted into this queue will be run under an `xterm` terminal emulator.

The following are other possible uses of a job starter:

- Set job starter to `'$USER_STARTER'`; enables users to define their own job starters by defining the environment variable `USER_STARTER`. LSF also supports a user-definable job starter at the command level. See the *LSF Batch User's Guide* for detailed information about setting up and using a command-level job starter to run interactive jobs.
- Set job starter to `'make clean;'` causes `make clean` to be run prior to user job.
- Set job starter to `pvmjob` or `mpijob`; allows you to run PVM or MPI jobs with LSF Batch, where `pvmjob` and `mpijob` are job starters for parallel jobs written in PVM or MPI.

A queue-level job starter is configured in the queue definition. See *'Job Starter'* on page 227 for details.

Note

The difference between a job starter and a pre-execution command lies in the effect each can have on the job being executed. A pre-execution command must run successfully and exit, which signals the batch daemon to run the job. Because the pre-execution command is an unrelated process, it does not effect the execution environment of the job. The job starter, however, is the process responsible for invoking the user command, and as such, controls the job's execution environment.

Using Licensed Software with LSF Batch

Software licenses are valuable resources that must be utilized to their full potential. This section discusses how LSF Batch can help manage licensed applications to maximize utilization and minimize job failure due to license problems.

Many applications have restricted access based on the number of software licenses purchased. LSF can help manage licensed software by automatically forwarding jobs to licensed hosts, or by holding jobs in batch queues until licenses are available.

There are three main types of software license: host locked, host locked counted, and network floating.

Host Locked Licenses

Host locked software licenses allow users to run an unlimited number of copies of the product on each of the hosts that has a license. You can configure a boolean resource to represent the software license, and configure your application to require the license resource. When users run the application, LSF chooses the best host from the set of licensed hosts.

See *'Changing LIM Configuration' on page 55* for instructions on configuring boolean resources, and *'The `lsf.task` and `lsf.task.cluster` Files' on page 187* for instructions on configuring resource requirements for an application.

Host Locked Counted Licenses

Host locked counted licenses are only available on specific licensed hosts, but also place a limit on the maximum number of copies available on the host. If an external LIM can get the number of licenses currently available, you can configure an external load index `licenses` giving the number of free licenses on each host. By specifying `licenses>=1` in the resource requirements for the application, you can restrict the application to run only on hosts with available licenses.

See *'Changing LIM Configuration' on page 55* for instructions on writing and using an ELIM, and *'The `lsf.task` and `lsf.task.cluster` Files' on page 187* for instructions on configuring resource requirements for an application.

4 Tuning LSF Batch

If a shell script `check_license` can check license availability and acquires a license if one is available, another solution is to use this script as a pre-execution command when submitting the licensed job.

```
% bsub -m licensed_hosts -E check_license licensed_job
```

An alternative is to configure the `check_license` script as a queue level pre-execution command. See *'Queue-Level Pre-/Post-Execution Commands'* on page 224 for more details.

It is possible that the license becomes unavailable between the time the `check_license` script is run, and when the job is actually run. To handle this case, the LSF administrator can configure a queue so that jobs in this queue will be requeued if they exit with value(s) indicating that the license was not successfully obtained. See *'Automatic Job Requeue'* on page 231.

Floating Licenses

A floating license allows up to a fixed number of machines or users to run the product at the same time, without restricting which host the software can run on. Floating licenses can be thought of as 'cluster resources'; rather than belonging to a specific host, they belong to all hosts in the cluster.

Using LSF Batch to run licensed software can improve the utilization of the licenses - the licenses can be kept in use 24 hours a day, 7 days a week. For expensive licenses, this increases their value to the users. Also, productivity can be increased, as users do not have to wait around for a license to become available.

LSF can be used to manage floating licenses using the shared resources feature together with resource reservation and job queuing. Both situations where all license jobs are run through LSF Batch and when licenses can be used outside of batch control are discussed.

All Licenses Used Through LSF Batch

If all jobs requiring licenses are submitted through LSF Batch, then LSF Batch could regulate the allocation of licenses to jobs and ensure that a job is not started if the required license is not available. A static resource is used to hold the total number of licenses that are available. The static resource is used by LSF Batch as a counter which

is decremented by the resource reservation mechanism each time a job requiring that resource is started.

For example, suppose that there are 10 licenses for the Verilog package shared by all hosts in the cluster. The LSF Base configuration files should be specified as shown below. The resource is static-valued so an ELIM is not necessary.

```
lsf.shared
Begin Resource
RESOURCENAME  TYPE      INTERVAL  INCREASING  DESCRIPTION
verilog       Numeric   ()         N            (Floating licenses for Verilog)
End Resource

lsf.cluster.cluster

Begin ResourceMap
RESOURCENAME  LOCATION
verilog       (10@[all])
End ResourceMap
```

The users would submit jobs requiring Verilog licenses as follows:

```
bsub -R 'rusage[verilog=1]' myprog
```

If a dedicated queue is defined to run Verilog jobs, then the LSF administrator can specify the resources requirements at the queue-level:

```
Begin Queue
QUEUE_NAME = q_verilog
RES_REQ=rusage[verilog=1]
End Queue
```

If the Verilog licenses are not cluster-wide and can only be used by some hosts in the cluster, then the resource requirement string should be modified to include the 'defined()' tag in the `select` section, as follows:

```
select[defined(verilog)] rusage[verilog=1]
```

For each job in the queue “q_verilog”, LSF Batch will reserve a Verilog license before dispatching a job, and release the license when the job completes. The number of licenses being reserved can be shown using the `bhosts -s` command. One limitation

4 Tuning LSF Batch

of this approach is that if a job does not actually use the license then the licenses will be under-utilized. This could happen if the user mistakenly specifies that their application needs a Verilog license, or submits a non-Verilog job to a Verilog queue. LSF Batch assumes that each job indicating that it requires a Verilog license will actually use it, and simply subtracts the total number of jobs requesting Verilog licenses from the total number available to decide whether an additional job can be dispatched.

Licenses Used Outside of LSF Batch

To handle the situation where application licenses are used by jobs outside of LSF Batch, an ELIM should be used to collect the actual number of licenses available instead of relying on a statically configured value. LSF Batch is periodically informed of the number of available licenses and takes this into consideration when scheduling jobs. Assuming there are a number of licenses for the Verilog package that can be used by all the hosts in the cluster, the LSF Base configuration files could be set up to monitor this resource as follows:

lsf.shared

Begin Resource

RESOURCENAME	TYPE	INTERVAL	INCREASING	DESCRIPTION
verilog	Numeric	60	N	(Floating licenses for Verilog)

End Resource

lsf.cluster.cluster

Begin ResourceMap

RESOURCENAME	LOCATION
verilog	([all])

End ResourceMap

The INTERVAL in the `lsf.shared` file would indicate how often the ELIM was expected to update the value of the 'Verilog' resource (in this case every 60 seconds). Since this resource is shared by all hosts in the cluster, the ELIM would only need to be started on the master host. If the Verilog licenses can only be accessed by some hosts in the cluster, the LOCATION field of the "ResourceMap" section should be specified as `([hostA hostB hostC ...])`. In this case an ELIM is only started on hostA.

The users would submit jobs requiring Verilog licenses as follows:

```
bsub -R 'rusage[verilog=1:duration=1]' myprog
```

LSF administrators can set up a queue dedicated to jobs that require Verilog licenses:

```
Begin Queue
QUEUE_NAME = q_verilog
RES_REQ=rusage[verilog=1:duration=1]
End Queue
```

The queue named `q_verilog` contains jobs that will reserve one Verilog license when it is started. Notice the duration specified (in minutes) is used to avoid the under utilization of shared resources. When duration is specified, the shared resource will be released after the specified duration expires. The reservation prevents the multiple jobs which are started in a short interval from over-using the available licenses. By limiting the duration of the reservation and using the actual license usage as reported by the ELIM, underutilization is also avoided and licenses used outside of LSF can be accounted for.

In situations where an interactive job outside the control of LSF Batch competes with batch jobs for a software license, it is possible that a batch job, having reserved the software license, may fail to start as the very license is intercepted by an interactive job. To handle this situation it is required that LSF Batch requeue the job for future execution. Job requeue can be achieved by using `REQUEUE_EXIT_VALUES` keyword in a queue's definition (see `lsb.queues(5)`). If a job exits with one of the values in the `REQUEUE_EXIT_VALUES`, LSF Batch will requeue the job. For example, jobs submitted to the following queue will use Verilog licenses:

```
Begin Queue
QUEUE_NAME = q_verilog
RES_REQ=rusage[verilog=1:duration=1]
# application exits with value 99 if it fails to get license
REQUEUE_EXIT_VALUE = 99
JOB_STARTER = lic_starter
End Queue
```

All jobs in the queue are started by `lic_starter`, which checks if the application failed to get a license and exits with an exit code of 99. This will cause the job to be requeued and the system will attempt to reschedule it at a later time. `lic_starter` can be coded as follows:

```
#!/bin/sh
# lic_starter: If application fails with no license, exit 99,
```

4 Tuning LSF Batch

```
# otherwise, exit 0. The application displays
# "no license" when it fails without license available.
$* 2>&l | grep "no license"
if [ $? != "0" ]
then
    exit 0      # string not found, application got the license
else
    exit 99
fi
```

Example LSF Batch Configuration Files

Example Queues

There are numerous ways to build queues. This section provides some examples.

Idle Queue

You want to dispatch large batch jobs only to those hosts that are idle. These jobs should be suspended as soon as an interactive user begins to use the machine. You can (arbitrarily) define a host to be idle if there has been no terminal activity for at least 5 minutes and the 1 minute average run queue is no more than 0.3. The idle queue does not start more than one job per processor.

```
Begin Queue
QUEUE_NAME = idle
NICE       = 20
RES_REQ    = it>5 && r1m<0.3
STOP_COND  = it==0
RESUME_COND = it>10
PJOB_LIMIT = 1
End Queue
```

Owners Queue

If a department buys some fast servers with its own budget, they may want to restrict the use of these machines to users in their group. The owners queue includes a `USERS` section defining the list of users and user groups that are allowed to use these machines. This queue also defines fairshare policy so that users can have equal sharing of resources.

```
Begin Queue
QUEUE_NAME = owners
PRIORITY   = 40
rlm        = 1.0/3.0
FAIRSHARE  = USER_SHARES[[default, 1]]
USERS      = server_owners
HOSTS      = server1 server2 server3
End Queue
```

Night Queue

On the other hand, the department might want to allow other people to use its machines during off hours so that the machine cycles are not wasted. The night queue only schedules jobs after 7 p.m. and kills jobs around 8 a.m. every day. Jobs are also allowed to run over the weekend.

To ensure jobs in the night queue do not hold up resources after the run window is closed, `TERMINATE_WHEN` is defined as `WINDOW` so that when the run window is closed, jobs that have been started but have not finished will be killed.

Because no `USERS` section is given, all users can submit jobs to this queue. The `HOSTS` section still contains the server host names. By setting `MEMLIMIT` for this queue, jobs that use a lot of real memory automatically have their time sharing priority reduced on hosts that support the `RLIMIT_RSS` resource limit.

This queue also reserves swp memory of 40MB for the job and this reservation decreases to 0 over 20 minutes after the job starts.

4 Tuning LSF Batch

```
Begin Queue
QUEUE_NAME      = night
RUN_WINDOW      = 5:19:00-1:08:00 19:00-08:00
PRIORITY        = 5
RES_REQ         = ut<0.5 && swp>50 rusage[swp=40:duration=20:decay=1]
rlm             = 0.5/3.0
MEMLIMIT        = 5000
TERMINATE_WHEN  = WINDOW
HOSTS           = server1 server2 server3
DESCRIPTION     = Low priority queue for overnight jobs
End Queue
```

License Queue

Some software packages have fixed licenses and must be run on certain hosts. Suppose a package is licensed to run only on a few hosts that are tagged with `product` resource. Also suppose that on each of these hosts, only one license is available.

To ensure correct hosts are chosen to run jobs, a queue level resource requirement `'type==any && product'` is defined. To ensure that the job gets a license when it starts, the `HJOB_LIMIT` has been defined to limit one job per host. Since software licenses are expensive resources that should not be under-utilized, the priority of this queue has been defined to be higher than any other queues so that jobs in this queue are considered for scheduling first. It also has a small nice value so that more CPU time is allocated to jobs from this queue.

```
Begin Queue
QUEUENAME       = license
NICE            = 0
PRIORITY        = 80
HJOB_LIMIT      = 1
RES_REQ         = type==any && product
rlm             = 2.0/4.0
DESCRIPTION     = Licensed software queue
End Queue
```

Short Queue

The short queue can be used to give faster turnaround time for short jobs by running them before longer jobs.

Jobs from this queue should always be dispatched first, so this queue has the highest `PRIORITY` value. The `r1m` scheduling threshold of 2 and no suspending threshold mean that jobs are dispatched even when the host is being used and are never suspended. The `CPULIMIT` value of 15 minutes prevents users from abusing this queue; jobs running more than 15 minutes are killed.

Because the short queue runs at a high priority, each user is only allowed to run one job at a time.

```
Begin Queue
QUEUE_NAME = short
PRIORITY   = 50
r1m        = 2/
CPULIMIT   = 15
UJOB_LIMIT = 1
DESCRIPTION = For jobs running less than 15 minutes
End Queue
```

Because the short queue starts jobs even when the load on a host is high, it can preempt jobs from other queues that are already running on a host. The extra load created by the short job can make some load indices exceed the suspending threshold for other queues, so that jobs from those other queues are suspended. When the short queue job completes, the load goes down and the preempted job is resumed.

Front End Queue

Some special-purpose computers are accessed through front end hosts. You can configure the front end host in `lsb.hosts` so that it accepts only one job at a time, and then define a queue that dispatches jobs to the front end host with no scheduling constraints.

Suppose *hostD* is a front end host:

```
Begin Queue
QUEUE_NAME = front
PRIORITY   = 50
HOSTS      = hostD
JOB_STARTER = pload
DESCRIPTION = Jobs are queued at hostD and started with pload command
End Queue
```

4 Tuning LSF Batch

NQS Forward Queue

To interoperate with NQS, you must configure one or more LSF Batch queues to forward jobs to remote NQS hosts. An NQS forward queue is an LSF Batch queue with the parameter `NQS_QUEUES` defined. The following queue forwards jobs to the NQS queue named `pipe` on host `cray001`:

```
Begin Queue
QUEUE_NAME = nqsUse
PRIORITY   = 30
NICE       = 15
QJOB_LIMIT = 5
CPULIMIT   = 15
NQS_QUEUES = pipe@cray001
DESCRIPTION = Jobs submitted to this queue are forwarded to NQS_QUEUES
USERS      = all
End Queue
```

Example `lsb.hosts` file

The `lsb.hosts` file defines host attributes. Host attributes also affect the scheduling decisions of LSF Batch. By default LSF Batch uses all server hosts as configured by LIM configuration files. In this case you do not have to list all hosts in the `Host` section. For example:

```
Begin Host
HOST_NAME    MXJ    JL/U    swp    # This line is keyword(s)
default      2      1      20
End Host
```

The virtual host name `default` refers to each of the other hosts configured by LIM but is not explicitly mentioned in the `Host` section of the `lsb.hosts` file. This file defines a total allowed job slot limit of 2 and a per user job limit of 1 for every batch server host. It also defines a scheduling load threshold of 20MB of swap memory.

In most cases your cluster is heterogeneous in some way, so you might have different controls for different machines. For example:

```
Begin Host
HOST_NAME      MXJ      JL/U      swp      # This line is keyword(s)
hostA          8        2         ( )
hppa           2        ( )       ( )
default        2        1         20
End Host
```

In this file you add host type `hppa` in the `HOST_NAME` column. This will include all server hosts from LIM configuration that have host type `hppa` and are not explicitly listed in the `Host` section of this file. You can also use a host model name for this purpose. Note the `()` in some of the columns. It refers to undefined parameters and serves as a place-holder for that column.

`lsb.hosts` file can also be used to define host groups and host partitions, as exemplified in *'Sharing Hosts Between Two Groups'* on page 115.

4 Tuning LSF Batch

5. Managing LSF MultiCluster

What is LSF MultiCluster?

Within a single organization, divisions, departments, or sites may have separate LSF clusters managed independently. Many organizations have realized it is desirable to allow their multitude of clusters to cooperate to reap the benefits of global load sharing:

- Users can access a diverse collection of computing resources and get better performance as well as computing capabilities. Many machines that would otherwise be idle can be used to process jobs. Multiple machines can be used to process a single parallel job. All these lead to increased user productivity.
- The demands for computing resources fluctuate widely across departments and over time. Partitioning the resources of an organization along user and departmental boundaries forces each department to plan for computing resources according to its maximum demand. Load sharing makes it possible for an organization to plan computing resources globally based on total demand. Resources can be added anywhere and made available to the entire organization. Global policies for load sharing can be implemented. With efficient resource sharing, the organization can realize increased computer usage in an economical manner.

LSF MultiCluster enables a large organization to form multiple cooperating clusters of computers so that load sharing happens not only within the clusters but also among them. It enables load sharing across large numbers of hosts, allows resource ownership and autonomy to be enforced, non-shared user accounts and file systems to be supported, and communication limitations among the clusters to be taken into consideration in job scheduling.

LSF MultiCluster is a separate product in the LSF product suite. You must obtain a specific license for LSF MultiCluster before you can use it.

5 Managing LSF MultiCluster

This chapter describes the configuration and operational details of LSF MultiCluster. The topics covered are:

- Monitoring of load and host information of remote clusters
- Accessing control of inter-cluster interactive tasks
- Executing batch jobs transparently on remote clusters
- Account mapping between clusters not sharing a uniform username/user ID space.

Enabling MultiCluster Functionalities

The following steps should be followed to enable the sharing of load information, interactive tasks and batch jobs between clusters:

- 1) Define the multicluster feature in the `lsf.cluster.cluster` file. Your licence must have multicluster support.
- 2) Configure LIM to specify the sharing of load information and interactive job access control.
- 3) Configure LSF Batch to specify the queues sharing jobs and account mapping between the users.

The LIM configuration files `lsf.shared` and `lsf.cluster.cluster` (stored in `LSF_CONFDIR`) are affected by multicluster operation. For sharing to take place between clusters, they must share common definitions in terms of host types, models, and resources. For this reason, it is desirable to make the `lsf.shared` file the same on each cluster, often by putting it into a shared file system, or replicating it across all clusters.

Where it is not possible to maintain a common `lsf.shared` file, and each cluster maintains its own, the exchange of system information and jobs between clusters is based on the common definitions. A resource, host type, or model defined in one cluster is considered to be equivalent to that defined in another cluster if the name is the same. It is possible, for example, to define a host model with the same name but with different CPU factors so that each cluster considers the relative CPU speed differently.

In such cases, each cluster will interpret resource, host type or model information received from another cluster based on its local `lsf.shared` file. If the definition is not found locally, then it is ignored.

For example, if the remote cluster defines a static boolean resource `local_res` and associates it with `hostA`, then when `hostA` is viewed from the local cluster, `local_res` will not be associated with it. Similarly, a user will not be able to submit a job locally specifying a resource which is only defined in a remote cluster.

Each LIM reads the `lsf.shared` file and its own `lsf.cluster.cluster` file. All information about a remote cluster is retrieved dynamically by the master LIM's on each cluster communicating with each other. However, before this can occur a master LIM must know the name of at least some of the LSF server hosts in each remote cluster with which it will interact. The names of the servers in a remote cluster are used to locate the current master LIM on that cluster as well as to ensure that any remote master is a valid host for that cluster. The latter is necessary to ensure security and prevent a bogus LIM from interacting with your cluster.

The `lsf.shared` File

The `lsf.shared` file in `LSF_CONFDIR` should list the names of all clusters. For example:

```
Begin Cluster
ClusterName
clus1
clus2
End Cluster
```

The LIM will read the `lsf.cluster.cluster` file in `LSF_CONFDIR` for each remote cluster and save the first ten host names listed in the `Host` section. These will be considered as valid servers for that cluster, that is, one of these servers must be up and running as the master.

If `LSF_CONFDIR` is not shared or replicated then it is necessary to specify a list of valid servers in each cluster using the option `Servers` in the `Cluster` section. For example,

```
Begin Cluster
ClusterName      Servers
clus1            (hostC hostD hostE)
```

5 Managing LSF MultiCluster

```
clus2          (hostA hostB hostF)
End Cluster
```

The hosts listed in the `servers` column are the contacts for LIMs in remote clusters to get in touch with the local cluster. One of the hosts listed in the `Servers` column must be up and running as the master for other clusters to contact the local cluster.

The `lsf.cluster.cluster` File

To enable the multicluster feature, insert the following section into the `lsf.cluster.cluster` file.

```
Begin Parameters
PRODUCTS=LSF_Base LSF_MultiCluster LSF_Batch
End Parameters
```

Note

The license file must support the LSF MultiCluster feature. If you have configured the cluster to run LSF MultiCluster on all hosts, and the license file does not contain the LSF MultiCluster feature, then the hosts will be unlicensed, even if you have valid licenses for other LSF components. See ‘Setting Up the License Key’ on page 36 of the LSF Installation Guide for more details.

By default, the local cluster can obtain information about all other clusters specified in `lsf.shared`. However, if the local cluster is only interested in certain remote clusters, you can use the following section in `lsf.cluster.cluster` to limit which remote clusters your cluster is interested in. For example,

```
Begin RemoteClusters
CLUSTERNAME
clus3
clus4
End RemoteClusters
```

This means local applications will not know anything about clusters other than clusters `clus3` and `clus4`. Note that this also affects the way RES behaves when it is authenticating a remote user. Remote execution requests originating from users outside of these clusters are rejected. The default behaviour is to accept any request from all the clusters in `lsf.shared`.

The `RemoteClusters` section may be used to specify the following parameters associated with each cluster in addition to the `CLUSTERNAME` parameter.

CACHE_INTERVAL

Load and host information is requested on demand from the remote cluster and cached by the local master LIM. Clients in the local cluster receives the cached copy of the remote cluster information. This parameter controls how long load information from the remote cluster is cached in seconds. The default is 60 seconds. Upon a request from a command, the cached information is used if it is less than `CACHE_INTERVAL` second old otherwise fresh information is retrieved from the relevant remote cluster by the local master LIM and returned to the user. Host information is cached twice as long as load information is.

EQUIV

The LSF utilities such as `lsload`, `lshosts`, `lsplace`, and `lsrun` normally only return information about the local cluster. To get information about or run tasks on hosts in remote clusters, you must explicitly specify a cluster name (see sections below). To make resources in remote clusters as transparent as possible to the user, you can specify a remote cluster as being *equivalent* to the local cluster. The master LIM will then consider all equivalent clusters when servicing requests from clients for load, host or placement information. Therefore, you do not have to explicitly specify remote cluster names. For example, `lsload` will list hosts of the local cluster as well as the remote clusters.

RECV_FROM

By default, if two clusters are configured to access each other's load information, they also accept interactive jobs from each other. If you want your cluster to access load information of another cluster but not to accept interactive jobs from the other cluster, you set `RECV_FROM` to 'N'. Otherwise, set `RECV_FROM` to 'Y'.

Example

For cluster *clus1*, *clus2* is equivalent to the local cluster. Load information is refreshed every 30 seconds. However, *clus1* rejects interactive jobs from *clus2*.

```
# Excerpt of lsf.cluster.clus1
Begin RemoteClusters
```

5 Managing LSF MultiCluster

```
CLUSTERNAME      EQUIV  CACHE_INTERVAL  RECV_FROM
clus2             Y      30              N
...
End RemoteClusters
```

Cluster *clus2* does not treat *clus1* as equivalent to the local cluster. Load information is refreshed every 45 seconds. Interactive jobs from *clus1* are accepted.

```
# Excerpt of lsf.cluster.clus1
Begin RemoteClusters
CLUSTERNAME      EQUIV  CACHE_INTERVAL  RECV_FROM
clus1             N      45              Y
...
End RemoteClusters
```

Root Access

By default, root access across clusters is not allowed. To allow root access from a remote cluster, specify `LSF_ROOT_REX=all` in `lsf.conf`. This implies that root jobs from both the local and remote clusters are accepted. This applies to both interactive and batch jobs.

If you want cluster *clus1* and *clus2* to allow root access execution for local jobs only, you insert the line `LSF_ROOT_REX=local` into the `lsf.conf` of both cluster *clus1* and cluster *clus2*. However, if you want *clus2* to also allow root access execution from any cluster, change the line in `lsf.conf` of cluster *clus2* to `LSF_ROOT_REX=all`.

Note

lsf.conf file is host type specific and not shared across different platforms. You must make sure that the `lsf.conf` file for all your host types are changed consistently.

LSF Batch Configuration

To enable batch jobs to flow across clusters the keywords `SNDJOBS_TO` and `RCVJOBS_FROM` are used in the queue definition of the `lsb.queues` file.

The syntax is as follows:

```
Begin Queue
QUEUE_NAME=normal
SNDJOBS_TO=Queue1@Cluster1 Queue2@Cluster2 ... QueueN@ClusterN
RCVJOBS_FROM=Cluster1 Cluster2 ... ClusterN
PRIORITY=30
NICE=20
End Queue
```

Note

You do not specify a remote queue in the RCVJOBS_FROM parameter. The administrator of the remote cluster determines which queues will forward jobs to the normal queue in this cluster.

It is up to you and the administrator of the remote clusters to ensure that the policy of the local and remote queues are *equivalent* in terms of the scheduling behaviour seen by users' jobs.

If a RCVJOBS_FROM queue specifies REQUEUE_EXIT_VALUES, it only applies to jobs submitted locally. Even if a remote job's exit value matches a value specified in the REQUEUE_EXIT_VALUES, the job is not requeued but the job and its exit value are forwarded to the submission cluster.

When accepting a job with a pre-execution command from a remote cluster, the local cluster can configure the maximum number of times it will attempt the pre-execution command before returning the job to the submission cluster. The submission cluster will forward the job to one cluster at a time. The parameter to control the maximum number of times a remote jobs pre-exec command is retried by setting MAX_PREEEXEC_RETRY in lsb.params.

Remote-Only MultiCluster Queues

In order to set up a queue that will forward jobs to remote clusters but will not run any jobs in the local cluster, you can specify that the queue uses no local hosts. This is done by setting the HOSTS parameter in the queue to the keyword "none".

5 Managing LSF MultiCluster

For example, the following definition sets up a queue `remote_only` in cluster `clus1` which sends the job to the `import` queue in cluster `clus2`:

```
Begin Queue
QUEUE_NAME = remote_only
HOSTS = none
SNDJOBS_TO = import@clus2
PRIORITY = 30
DESCRIPTION = A remote only queue
End Queue
```

Any jobs submitted to queue `remote_only` will be forwarded to the queue `import` in cluster `clus2`. This is done without attempting to schedule the job locally which reduces the latency of multicluster queues.

For `clus2`, the queue `import` can be specified as follows:

```
Begin Queue
QUEUE_NAME = import
RCVJOBS_FROM = clus1
PRIORITY = 50
DESCRIPTION = A queue that imports jobs from clus1
End Queue
```

Inter-cluster Load and Host Information Sharing

The information collected by LIMs on remote clusters can be viewed locally. The list of clusters and associated resources can be viewed with the `lsclusters` command.

```
% lsclusters
CLUSTER_NAME  STATUS  MASTER_HOST  ADMIN  HOSTS  SERVERS
clus2         ok      hostA        user1  3      3
clus1         ok      hostC        user2  3      3
```

If you have defined `EQUIV` to be 'Y' for cluster `clus2` in your `lsf.cluster.clus1` file, you will see all hosts in cluster `clus2` if you run `lsload` or `lshosts` from cluster `clus1`. For example:

```
% lshosts
HOST_NAME  type  model  cpuf  ncpus  maxmem  maxswp  server  RESOURCES
hostA      NTX86 PENT200 10.0  1    64M    100M    Yes (pc nt)
hostF      HPPA  HP735  14.0  1    58M    94M     Yes (hpux cs)
hostB      SUN41 SPARC3SLC 8.0   1    15M    29M     Yes (sparc bsd)
hostD      HPPA  A900   30.0  4    264M   512M    Yes (hpux cs bigmem)
hostE      SGI   ORIGIN2K 36.0  32   596M   1024M   Yes (irix cs bigmem)
hostC      SUNSOL SunSparc 12.0  1    56M    75M     Yes (solaris cs)
```

You can use a cluster name in place of a host name to get information specific to a cluster. For example:

```
% lshosts clus1
HOST_NAME  type  model  cpuf  ncpus  maxmem  maxswp  server  RESOURCES
hostD      HPPA  A900   30.0  4    264M   512M    Yes (hpux cs bigmem)
hostE      SGI   ORIGIN2K 36.0  32   596M   1024M   Yes (irix cs bigmem)
hostC      SUNSOL SunSparc 12.0  1    56M    75M     Yes (solaris cs)
```

```
% lshosts clus2
HOST_NAME  type  model  cpuf  ncpus  maxmem  maxswp  server  RESOURCES
hostA      NTX86 PENT200 10.0  1    64M    100M    Yes (pc nt)
hostF      HPPA  HP735  14.0  1    58M    94M     Yes (hpux cs)
hostB      SUN41 SPARC3SLC 8.0   1    15M    29M     Yes (sparc bsd)
```

```
% lsload clus1 clus2
HOST_NAME  status  r15s  r1m  r15m  ut    pg  ls    it  tmp  swp  mem
hostD      ok      0.2  0.3  0.4  19%  6.0  6    3   146M 319M 52M
hostC      ok      0.1  0.0  0.1  1%   0.0  3    43  63M  44M  7M
hostA      ok      0.3  0.3  0.4  35%  0.0  3    1   40M  42M  10M
hostB      busy   *1.3  1.1  0.7  68% *57.5 2    4   18M  25M  8M
hostE      lockU  1.2  2.2  2.6  30%  5.2  35   0   10M  293M 399M
hostF      unavail
```

LSF commands `lshosts`, `lsload`, `lsmon`, `lsrun`, `lsgrun`, and `lsplace` can accept a cluster name in addition to host names.

Running Interactive Jobs on Remote Clusters

The `lsrun` and `lslogin` commands can be used to run interactive jobs both within and across clusters. See *Running Batch Jobs across Clusters* on page 189 of the *LSF Batch User's Guide* for examples.

You can configure the multicluster environment so that one cluster accepts interactive jobs from the other cluster, but not vice versa. For example, to make *clus1* reject interactive jobs from *clus2*, you need to specify the `RECV_FROM` field in file `lsf.cluster.clus1`:

```
Begin RemoteClusters
CLUSTERNAME  EQUIV  CACHE_INTERVAL  RECV_FROM
clus2        Y      30              N
End RemoteClusters
```

When a user in *clus2* attempts to use the cluster *clus1*, an error will result. For example:

```
% lsrun -m clus1 -R - hostname
ls_placeofhosts: Not enough host(s) currently eligible
```

Cluster *clus2* will not make any placement of jobs on *clus1* and therefore `lsrun` will return an error about not being able to find enough hosts.

```
% lsrun -m hostC -R - hostname
ls_rsetenv: Request from a non-LSF host rejected
```

In this case, the job request is sent to the host *hostC* and the RES on *hostC* rejects the job as it is not considered a valid LSF host.

Note

`RECV_FROM` *only controls accessibility of interactive jobs. It does not affect jobs submitted to LSF Batch.*

Distributing Batch Jobs Across Clusters

As the administrator, you can configure a queue to send jobs to a queue in a remote cluster. Jobs submitted to the local queue can automatically get sent to remote clusters. The following commands can be used to get information about multiple clusters:

bclusters

The `bclusters` command displays a list of queues together with their relationship with queues in remote clusters.

```
% bclusters
LOCAL_QUEUE      JOB_FLOW  REMOTE      CLUSTER     STATUS
testmc           send      testmc      clus2       ok
testmc           recv      -           clus2       ok
```

The `JOB_FLOW` field describes whether the local queue is to send jobs to or receive jobs from the remote cluster.

If the value of `JOB_FLOW` is `send` (that is, `SNDJOBS_TO` is defined in the local queue), then the `REMOTE` field indicates a queue name in the remote cluster. If the remote queue in the remote cluster does not have `RCVJOBS_FROM` defined to accept jobs from the cluster, the status field will never be `ok`. It will either be `disc`, or `reject`, where `disc` means that the communication between the two clusters has not been established yet. This could occur if there are no jobs waiting to be dispatched or the remote master cannot be located. If remote cluster agrees to accept jobs from the local queue and communication has been successfully established, the status will be `ok`, otherwise the status will be `rejected`.

If the value of `JOB_FLOW` is `recv` (that is, `RCVJOBS_FROM` is defined in the local queue), then the `REMOTE` field is always `-`. The `CLUSTER` field then indicates the cluster name from which jobs will be accepted. The `status` field will be `ok` if a connection with the remote cluster has established.

```
% bclusters
LOCAL_QUEUE      JOB_FLOW  REMOTE      CLUSTER     STATUS
testmc           send      testmc      clus2       disc
testmc           recv      -           clus2       disc
```

5 Managing LSF MultiCluster

bqueues

The `-m host_name` option can optionally take a cluster name to display the queues in a remote cluster.

```
% bqueues -m clus2
```

QUEUE_NAME	PRIO	STATUS	MAX	JL/U	JL/P	JL/H	NJOBS	PEND	RUN	SUSP
fair	3300	Open:Active	5	-	-	-	0	0	0	0
interactive	1055	Open:Active	-	-	-	-	0	0	0	0
testmc	55	Open:Active	-	-	-	-	5	2	2	1
priority	43	Open:Active	-	-	-	-	0	0	0	0

bjobs

The `bjobs` command can display the cluster name in the `FROM_HOST` and `EXEC_HOST` fields. The format of these fields can be `'host@cluster'` to indicate which cluster the job originated from or was forwarded to. Use the `-w` option to get the full cluster name. To query the jobs in a specific cluster, use the `-m` option and specify the cluster name.

```
% bjobs
```

JOBID	USER	STAT	QUEUE	FROM_HOST	EXEC_HOST	JOB_NAME	SUBMIT_TIME
101	user7	RUN	testmc	hostC	hostA@clus2	simulate	Oct 8 18:32
102	user7	USUSP	testmc	hostC	hostB@clus2	simulate	Oct 8 18:56
104	user7	RUN	testmc	hostA@clus2	hostC	verify	Oct 8 19:20

```
% bjobs -m clus2
```

JOBID	USER	STAT	QUEUE	FROM_HOST	EXEC_HOST	JOB_NAME	SUBMIT_TIME
521	user7	RUN	testmc	hostC@clus1	hostA	simulate	Oct 8 18:35
522	user7	USUSP	testmc	hostC@clus1	hostB	simulate	Oct 8 19:23
520	user7	RUN	testmc	hostA	hostC@clus1	verify	Oct 8 19:26

Note that jobs forwarded to a remote cluster are assigned new job IDs. You only need to use local job IDs when manipulating local jobs. The `SUBMIT_TIME` field displays the real job submission time for local jobs, and job forwarding time for jobs from remote clusters.

bhosts

To view the hosts of a specific cluster you can use a cluster name in place of a host name.

```
% bhosts clus2
```

HOST_NAME	STATUS	JL/U	MAX	NJOBS	RUN	SSUSP	USUSP	RSV
hostA	ok	-	10	1	1	0	0	0
hostB	ok	-	10	1	1	0	0	0
hostF	closed	-	3	3	3	0	0	0

bhist

The `bhist` command displays the history of events about when a job is forwarded to another cluster or was accepted from another cluster.

```
% bhist -l 101
```

```
Job Id <101>, User <user7>, Project <default>, Command <simulate>
Tue Oct 08 18:32:11: Submitted from host <hostC> to Queue <testmc>, CWD <
    /homes/user7>, Requested Resources <type!=ALPHA>
    ;
Tue Oct 08 18:35:07: Forwarded job to cluster clus2;
Tue Oct 08 18:35:25: Dispatched to <hostA>;
Tue Oct 08 18:35:35: Running with execution home </homes/user7>, Execution C
    WD </homes/user7>, Execution Pid <25212>;
Tue Oct 08 20:30:50: USER suspend action initiated (actpid 25672);
Tue Oct 08 20:30:50: Suspended by the user or administrator.
```

```
Summary of time in seconds spent in various states by Tue Oct 08 20:35:24 1996
```

PEND	PSUSP	RUN	USUSP	SSUSP	UNKWN	TOTAL
176	0	6943	274	0	0	7393

Account Mapping Between Clusters

By default, LSF assumes a uniform user name space within a cluster and between clusters, but it is not uncommon for an organization to fail to satisfy this assumption. LSF Batch supports the execution of batch jobs across non-uniform user name spaces

5 Managing LSF MultiCluster

between clusters by allowing user account mapping between such clusters—at both the system level and the individual user level.

User Level Account Mapping

Individual users of the LSF cluster can set up their own account mapping by setting up a `.lsfhosts` file in their home directories. The `.lsfhosts` file used to support account mapping can be used to specify cluster names in place of host names.

Example #1

A user has accounts on two clusters, *clus1* and *clus2*. On cluster *clus1*, the user name is *userA* and on *clus2* the user name is *user_A*. To run jobs in either cluster under the appropriate user name, the `.lsfhosts` files should be set up as follows:

On machines in cluster *clus1*:

```
% cat ~userA/.lsfhosts
clus2 user_A
```

On machines in cluster *clus2*:

```
% cat ~user_A/.lsfhosts
clus1 userA
```

Example #2

A user has the account *userA* on cluster *clus1* and wants to use the *lsfguest* account when running jobs on cluster *clus2*. The `.lsfhosts` files should be set up as follows:

On machines in cluster *clus1*:

```
% cat ~userA/.lsfhosts
clus2 lsfguest send
```

On machines in cluster *clus2*:

```
% cat ~lsfguest/.lsfhosts
clus1 userA recv
```

Example #3

A site has two clusters, *clus1* and *clus2*. A user has a uniform account name as *userB* on all hosts in *clus2*. However, in *clus1*, this user has a uniform account name as *userA*, except on *hostX*, on which he has the account name *userA1*. This user would like to use both clusters transparently.

To implement this mapping, the user should set the `.lsfhosts` files in his home directories on different machines as follows:

On *hostX* of *clus1*:

```
% cat ~userA1/.lsfhosts
clus1    userA
hostX    userA1
clus2    userB
```

On any other machine in *clus1*:

```
% cat ~userA/.lsfhosts
clus2    userB
hostX    userA1
```

On the *clus2* machines:

```
% cat ~userB/.lsfhosts
clus1    userA
hostX    userA1
```

System Level Account Mapping

An LSF administrator can set up system level account mapping in the `lsb.users` file.

For a job submitted as one user at the submission cluster to run as another user in a remote execution cluster, the LSF Batch system requires that both clusters agree with this account mapping. The submission cluster can propose a set of user mappings and the execution cluster decides whether to accept these settings or not.

5 Managing LSF MultiCluster

The system level account mapping is defined in the “UserMap” section of the `lsb.users` file. It contains multiple account mapping entries, where each entry contains three fields:

- LOCAL: defines a list of local users
- REMOTE: defines a list of remote users in the form of `username@clustername`
- DIRECTION: two values can be used for this field: “export” and “import”. The “export” keyword indicates that exported jobs of users defined in the LOCAL column are running as the users in the REMOTE column. The “import” keyword indicates that imported jobs belonging to remote users specified in the REMOTE column are running as the users specified in the LOCAL column.

Example #1

For *userA* on cluster *clus1* to map to *userB* on cluster *clus2*, at *clus1*, the `lsb.users` file can be set up as follows:

```
Begin UserMap
LOCAL      REMOTE          DIRECTION
.
userA      userB@clus2     export
.
End UserMap
```

At *clus2*, the `lsb.users` file is set up as:

```
Begin UserMap
LOCAL      REMOTE          DIRECTION
.
userB      userA@clus1     import
.
End UserMap
```

Example #2

As another example, *userA* on *clus1* wants to run as *userB* or *userC* on *clus2*. The *clus1s* `lsb.users` file should be set up as follows:

```
Begin UserMap
LOCAL      REMOTE          DIRECTION
.
userA      (userB@clus2 userC@clus2)    export
.
End UserMap
```

At *clus2*, *userA* is allowed to run as both *userB* or *userD*:

```
Begin UserMap
LOCAL      REMOTE          DIRECTION
(userB userD)    userA@clus1    import
End UserMap
```

Despite the fact that *clus2* allows *userA* to also map to *userD*, *clus1* does not propose such a mapping and hence the common agreeable account mapping between *clus1* and *clus2* for *userA* is *userA@clus1* running as *userB@clus2*.

5 Managing LSF MultiCluster

6. LSF Base Configuration Reference

This chapter contains a detailed description of the contents of the LSF Base configuration files. These include the installation file `lsf.conf`; the LIM configuration files `lsf.shared`, `lsf.cluster.cluster`, `lsf.task`, and `lsf.task.cluster`; and the optional LSF hosts file for additional host name information.

The `lsf.conf` File

Installation of and operation of LSF is controlled by the `lsf.conf` file. The `lsf.conf` file is created during installation, and records all the settings chosen when LSF is installed. This information is used by LSF daemons and commands to locate other configuration files, executables, and network services.

`lsf.conf` contains LSF installation settings as well as some system-wide options. This file is initially created by the `lsfsetup` utility during LSF installation and updated, if necessary, when you upgrade to a new version. Many of the parameters are set during the installation. This file can also be expanded to include LSF application specific parameters.

LSB_CONFDIR

LSF Batch configuration directories are installed under `LSB_CONFDIR`. Configuration files for each LSF cluster are stored in a subdirectory of `LSB_CONFDIR`. This subdirectory contains several files that define the LSF Batch user and host lists, operation parameters, and batch queues.

All files and directories under `LSB_CONFDIR` must be readable from all hosts in the cluster. `LSB_CONFDIR/cluster/configdir` must be owned by the LSF administrator.

6 LSF Base Configuration Reference

Default: `LSF_CONFDIR/lSBATCH`

You should not try to redefine this parameter once LSF has been installed. If you want to move these directories to another location, you must make sure the permissions of directories and files are set properly. See *Appendix B, 'LSF Directories'*, beginning on page 255 for details.

LSB_DEBUG

If this is defined, LSF Batch will run in single user mode. In this mode, no security checking is performed, so the LSF Batch daemons should not run as root. When `LSB_DEBUG` is defined, LSF Batch will not look in the system services database for port numbers. Instead, it uses port number 40000 for `mbatchd` and port number 40001 for `sbatchd` unless `LSB_MBD_PORT/LSB_SBD_PORT` are defined in the file `lsf.conf`. The valid values for `LSB_DEBUG` are 1 and 2. You should always choose 1 unless you are testing LSF Batch.

Default: undefined

LSB_MAILPROG

LSF Batch normally uses `/usr/lib/sendmail` as the mail transport agent to send mail to users. If your site does not use `sendmail`, configure `LSB_MAILPROG` with the name of a `sendmail`-compatible transport program. LSF Batch calls `LSB_MAILPROG` with the following arguments:

```
LSB_MAILPROG -F "LSF Batch system" -f Manager@host dest_addr
```

The `-F "LSF Batch System"` argument sets the full name of the sender; the `-f Manager@host` argument gives the return address for LSF Batch mail, which is the LSF administrator's mailbox. `dest_addr` is the destination address, generated by the rules given for `LSB_MAILTO` above.

`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.

If this parameter is modified, the LSF administrator must restart the `sbatchd` daemons on all hosts to pick up the new value.

Default: `/usr/lib/sendmail`

LSB_MAILTO

LSF Batch 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 Batch system. The default is to send mail to the user who submitted the job, on the host where 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 Batch. `LSB_MAILTO` is a format string that is used to build the mailing address. The substring `!U`, if found, is replaced with the user's account name; the substring `!H` is replaced with the name of the submission host. All other characters (including any other `!`) are copied exactly. Common formats are:

`!U`

Mail is sent to the submitting user's account name on the local host.

`!U@!H`

Mail is sent to `user@submission_hostname`

`!U@company_name.com`

Mail is sent to `user@company_name.com`

If this parameter is modified, the LSF administrator must restart the `sbatchd` daemons on all hosts to pick up the new value.

Default: `!U`

LSB_SHAREDIR

LSF Batch keeps job history and accounting log files 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/cluster/logdir` directory must be owned by the LSF administrator.

Default: `LSF_INDEP/work`

6 LSF Base Configuration Reference

Note

All files and directories under `LSB_SHARED_DIR` must allow read and write access from the LSF master host. See 'Fault Tolerance' on page 5 and 'Resource and Resource Requirements' on page 8.

LSF_AFS_CELLNAME

This must be defined to AFS cell name if the AFS file system is in use.

Default: undefined

LSB_LOCALDIR

This parameter needs to be defined if you want to use the duplicate event logging feature. This parameter specifies a directory that is local to the default master lost, that is, the first host configured in your `lsf.cluster.<cluster>` file. See 'Duplicate Event Logging' on page 81 for more information about this topic.

LSF_AUTH

This is an optional definition. By default, external user authentication is used, and `LSF_AUTH` is defined to be `eauth`. External authentication is the only way to provide security for clusters that contain Windows NT hosts. See 'External Authentication' on page 11 for details.

If this parameter is changed, all the LSF daemons must be shut down and restarted by running `lsf_daemons start` on each of the LSF server hosts so that the daemons will use the new authentication method.

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. See 'Resource Requirements' on page 24 for instructions on registering service names.

If `LSF_AUTH` is not defined, LSF uses privileged ports for user authentication. LSF commands must be installed `setuid` to `root` to operate correctly. If the LSF 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 manual page for `mount` for more details.

Windows NT does not have the concept of setuid binaries and does not restrict access to privileged ports, so this method does not provide any security on Windows NT.

Default: `eauth`

LSF_EAUTH_KEY

This defines a key the `eauth` uses to encrypt and decrypt the user authentication data. 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 (like choosing a normal UNIX password).

If this parameter is not defined, then `eauth` will use an internal key.

Default: `undefined`

LSF_BINDIR

Directory where all user commands are installed.

Default: `LSF_MACHDEP/bin`

LSF_CONFDIR

The directory where all LIM 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.

Default: `LSF_INDEP/conf`

LSF_CROSS_UNIX_NT

Optional. If this exists and has the value `no`, `No`, or `NO`, all cross-platform job submissions and requests will fail.

This means that in a mixed UNIX/NT cluster, jobs submitted from a UNIX user account on a UNIX host must be run on a UNIX host, and requests to stop or modify the job must be also submitted from a UNIX user account. Windows NT jobs can only be started, stopped, or modified by Windows NT user accounts on Windows NT hosts.

6 LSF Base Configuration Reference

If this parameter is undefined, or defined as any other value, mixed UNIX/NT clusters operate properly, and only the user name is used for authentication of the user account.

Default: undefined

LSF_ECHKPNTDIR

Optional. Specifies the directory where the `echkpnt` and `erestart` executable files are installed, if they are not in the default location `LSF_SERVERDIR`.

Default: undefined

LSF_ENVDIR

LSF normally installs the `lsf.conf` file in the `/etc` directory. The `lsf.conf` file is installed by creating a shared copy in `LSF_SERVERDIR` 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`.

Default: `/etc`

LSF_INCLUDEDIR

Directory under which the LSF API header file `<lsf/lsf.h>` is installed.

Default: `LSF_INDEP/include`

LSF_INDEP

Specifies the default top-level directory for all host-type independent LSF files. This includes manual pages, configuration files, working directories, and examples. For example, defining `LSF_INDEP` as `/usr/local/lsf` places manual pages in `/usr/local/lsf/man`, configuration files in `/usr/local/lsf/conf`, and so on.

Default: `/usr/local/lsf`

LSF_LIBDIR

Directory where the LSF application programming interface library `liblsf.a` is installed.

Default: `LSF_MACHDEP/lib`

LSF_LICENSE_FILE

Either the full path name of the FLEXlm license file used by LSF, or the host name of the license server host machine and port number of the license service (format: `port_number@host_name`). If this variable is not defined, on UNIX LIM looks for the license in `/usr/local/flexlm/licenses/license.dat`. On NT, LIM looks for license in `C:\flexlm\licensed`.

Default: `LSF_CONFDIR/license.dat`

LSF_LIM_DEBUG

If `LSF_LIM_DEBUG` is defined, the Load Information Manager (LIM) will operate 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. The valid values for `LSF_LIM_DEBUG` are 1 and 2. You should always choose 1 unless you are testing LSF.

Default: undefined

LSF_LIM_PORT, LSF_RES_PORT, LSB_MBD_PORT, LSB_SBD_PORT

Internet port numbers to use for communication with the LSF daemons. The port numbers are normally obtained by looking up the LSF service names in the `/etc/services` file or the NIS (UNIX). If it is not possible to modify the service database, these variables can be defined 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.

6 LSF Base Configuration Reference

Default: get port numbers from services database on UNIX. On NT, these parameters are mandatory.

LSF_LOGDIR

This is an optional definition on UNIX and a mandatory parameter on NT.

Error messages from all servers are logged into files in this directory. If a server is unable to write in this directory, then the error logs are created in `/tmp` on UNIX and `C:\temp` on NT.

UNIX

If `LSF_LOGDIR` is not defined, then `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 manual pages for the `syslogd` daemon and the `syslog` function for more information.

Default: log messages go to `syslog`

LSF_LOG_MASK

The message log level for LSF daemons. On UNIX, this is similar to `syslog`. All messages logged at the specified level or higher are recorded; lower level messages are discarded. The log levels in order from highest to lowest are:

- `LOG_ALERT`
- `LOG_SALERT`
- `LOG_EMERG`
- `LOG_ERR`
- `LOG_CRIT`
- `LOG_WARNING`
- `LOG_NOTICE`

- LOG_INFO
- LOG_DEBUG

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.

Note that 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.

Default: LOG_WARNING

LSF_MACHDEP

Specifies the directory where host type dependent files are installed. In clusters with a single host type, LSF_MACHDEP is usually the same as LSF_INDEP. The machine dependent files are the user programs, daemons, and libraries. You should not need to modify this parameter.

Default: /usr/local/lsf

UNIX LSF_MANDIR

Directory under which all manual pages are installed. The manual 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.

Default: LSF_INDEP/man

Note

Manual pages are installed in a format suitable for BSD style man commands.

LSF_MISC

Directory where miscellaneous machine independent files such as LSF example source programs and scripts are installed.

6 LSF Base Configuration Reference

Default: `LSF_CONFDIR/misc`

LSF_RES_ACCT

If defined, RES will log task information by default (see `lsf.acct(5)`). If this parameter is not defined, the LSF administrator must use the `lsadmin` command (see `lsadmin(8)`) to turn task logging on after the RES has started up. A CPU time (in msec) can be specified for the value for this parameter; only tasks that have consumed more than the specified CPU time will be logged. If it is defined as `LSF_RES_ACCT=0`, all tasks will be logged.

Default: undefined

LSF_RES_ACCTDIR

The directory where the RES task log file is stored. If `LSF_RES_ACCTDIR` is not defined, log file is stored in the `/tmp` directory.

Default: `/tmp` on UNIX. `C:\temp` on NT.

LSF_RES_DEBUG

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. The valid values for `LSF_RES_DEBUG` are 1 and 2. You should always choose 1 unless you are testing RES.

Default: undefined

UNIX

LSF_ROOT_REX

This is an optional definition.

If `LSF_ROOT_REX` is defined, RES accepts 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. 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. If the value of this parameter is defined to 'all', then root remote execution across the cluster is enabled. This applies to LSF MultiCluster only. Setting `LSF_ROOT_REX` to any other value only enables root remote execution within the local cluster.

Default: undefined. Root execution is not allowed.

LSF_SERVERDIR

Directory where all server binaries are installed. These include `lim`, `res`, `nios`, `sbatchd`, `mbatchd`, and `eeventd` (for LSF JobScheduler only). If you use `elim`, `eauth`, `eexec`, `esub`, etc, they should also be installed in this directory.

Default: `LSF_MACHDEP/etc`

LSF_SERVER_HOSTS

This defines one or more LSF server hosts that the application should contact to find a Load Information Manager (LIM). This is used on client hosts where no LIM is running on the local host. The 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. See *'Associating Resources with Hosts'* on page 60 for more details about server and client hosts.

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"
```

Default: undefined

LSF_STRIP_DOMAIN

This is an optional definition.

If all 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

6 LSF Base Configuration Reference

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.

Default: undefined

UNIX **LSF_USE_HOSTEQUIV**

This is an optional definition.

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. 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

UNIX **XLSF_APPDIR**

The directory where 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 LSF X applications. The application default files are platform-independent.

Default: `LSF_INDEP/misc`

UNIX **XLSF_UIDDIR**

The directory where Motif User Interface Definition files are stored. These files are platform-specific.

Default: `LSF_LIBDIR/uid`

LSF_RES_RLIMIT_UNLIM

By default, the 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.

For example:

```
LSF_RES_RLIMIT_UNLIM="cpu core stack"
```

will set the `cpu`, core size, and stack hard limits to be unlimited for all remote tasks.

Default: undefined

Note

The `LSF_RES_RLIMIT_UNLIM` parameter applies to LSF Base only.

The `lsf.shared` File

The `lsf.shared` file contains definitions that are used by all load sharing clusters. This includes lists of cluster names, host types, host models, the special resources available, and external load indices.

Clusters

The mandatory `Cluster` section defines all cluster names recognized by the LSF system, with one line for each cluster.

The `ClusterName` keyword is mandatory. 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.

6 LSF Base Configuration Reference

```
Begin Cluster
ClusterName
cluster1
cluster2
End Cluster
```

Host Types

The mandatory `HostType` section lists the valid host type names in the cluster. Each host is assigned a host type in the `lsf.cluster.cluster` file. All hosts that can run the same binary programs should have the same host type, even if they have different models of processor. LSF uses the host type as a default requirement for task placement. Unless specified otherwise, jobs are always run on hosts of the same type.

The `TYPENAME` keyword is mandatory. Host types are usually based on a combination of the hardware name and operating system. For example, a HP-PA system runs the HP-UX operating system, so you could assign the host type `HPPA`. If your site already has a system for naming host types, you can use the same names for LSF.

```
Begin HostType
TYPENAME
SUN41
SOLSPARC
ALPHA
HPPA
NTX86
End HostType
```

Host Models

The mandatory `HostModel` section lists the various models of machines and gives the relative CPU speed for each model. LSF uses the relative CPU speed to normalize the CPU load indices so that jobs are more likely to be sent to faster hosts. The `MODELNAME` and `CPUFACTOR` keywords are mandatory.

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.

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.

```
Begin HostModel
MODELNAME  CPUFACTOR
SparcIPC   1.0
Sparc10    2.0
End HostModel
```

The CPU factor affects the calculation of job execution time limits and accounting. Using large values for the CPU factor can cause confusing results when CPU time limits or accounting are used. See *'Resource Limits' on page 217* for more information.

Resources

The section Resource is optional. This section is used to define resource names. The following keywords are supported:

RESOURCENAME

This parameter is mandatory for each resource to be configured. A resource name is an arbitrary character string, except the following reserved names:

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.

cpu

Alias for r1m.

ut

The CPU utilization, exponentially averaged over the last minute, between 0 and 1.

6 LSF Base Configuration Reference

<code>pg</code>	The memory paging rate, exponentially averaged over the last minute, in pages per second.
<code>io</code>	The disk I/O rate exponentially averaged over the last minute, in KBytes per second.
<code>ls</code>	The number of current login users.
<code>logins</code>	Alias for <code>ls</code> .
<code>it</code>	The idle time of the host (keyboard not touched on all logged in sessions), in minutes.
<code>idle</code>	Alias for <code>it</code> .
<code>tmp</code>	The amount of free space in <code>/tmp</code> , in MBytes.
<code>swp</code>	The amount of currently available swap space, in MBytes.
<code>swap</code>	Alias for <code>swp</code> .
<code>mem</code>	The amount of currently available memory, in MBytes.
<code>n_cpus</code>	The number of CPUs on the host.
<code>ndisks</code>	The number of local disks on the host.

`maxmem`

The maximum physical memory, in MBytes.

`maxswp`

The maximum swap space, in MBytes.

`maxtmp`

The maximum space in the disk partition containing the `/tmp` directory, in MBytes.

`cpuf`

The processor CPU factor.

`type`

The host type.

`model`

The host model.

`status`

The host status.

A resource name cannot begin with a number, and cannot contain any of the following characters:

`: . () [+ - * / ! & | < > @ =`

`TYPE`

The `TYPE` is either boolean, numeric, or string. A boolean resource has a value of 1 on hosts which have that resource, and 0 otherwise. If `TYPE` is not given, the default type is boolean. Examples of boolean resource names include `sparc` (architecture), `sysv` (System V Unix), `fs` (file server), `cs` (compute server), and `solaris` (operating system).

`INTERVAL`

This parameter defines the time interval (in seconds) at which the resource is sampled by the external `LIM`. This keyword applies to dynamic resources only. A dynamic resource changes its value over time. An `ELIM` needs to be configured to sample and report this value to the `LIM`. If the resource has type numeric and has `INTERVAL` defined, then this resource becomes an external load index. This way of defining an external load index obsoletes the

6 LSF Base Configuration Reference

NewIndex section in the `lsf.shared` file. If `INTERVAL` is not given, the resource is considered static.

INCREASING

This parameter applies to numeric resources only. If a larger value means a greater load, then `INCREASING` should be defined as 'Y', otherwise 'N'.

DESCRIPTION

This is a 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.

RELEASE

This parameter controls whether a shared resource is released when a job is suspended. `RELEASE` applies to numeric resources only, such as floating licenses. 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.

Specify `Y` to release the resource.

Default: `Y`

The `lsf.cluster.cluster` File

This is the load-sharing cluster configuration file. There is one such file for each load-sharing cluster in the system. The *cluster* suffix must agree with the name defined in the `Cluster` section of the `lsf.shared` file.

Parameters

The `Parameters` section is optional. This section contains miscellaneous parameters for the LIM.

PRODUCTS

The `PRODUCTS` line specifies which LSF product(s) will be enabled in the cluster. The `PRODUCTS` line can specify any combination of the strings `'LSF_Base'`, `'LSF_Batch'`, `'LSF_JobScheduler'`, `'LSF_MultiCluster'`, and `'LSF_Analyzer'` to enable the operation of LSF Base, LSF Batch, LSF JobScheduler, LSF MultiCluster, and LSF Analyzer, respectively. If any of `'LSF_Batch'`, `'LSF_JobScheduler'`, or `'LSF_MultiCluster'` are specified then `'LSF_Base'` is automatically enabled as well. Specifying the `PRODUCTS` line enables the feature for all hosts in the cluster. Individual hosts can be configured to run as LSF Batch servers or LSF JobScheduler servers within the same cluster. LSF MultiCluster is either enabled or disabled for multicluster operation for the entire cluster.

The `PRODUCTS` line is created automatically by the installation program `lsfsetup`. For example:

```
Begin Parameters
PRODUCTS=LSF_Base LSF_Batch
End Parameters
```

If the `PRODUCTS` line is not specified, the default is to enable the operation of `'LSF_Base'` and `'LSF_Batch'`.

Note

The features defined by the `PRODUCTS` line must match the license file used to serve the cluster. A host will be unlicensed if the license is unavailable for the component it was configured to run. For example, if you configure a cluster to run LSF JobScheduler on all hosts, and the license file does not contain the LSF JobScheduler feature, then the hosts will be unlicensed, even if there are licenses for LSF Base or LSF Batch.

Default: `LSF_Base LSF_Batch`

ELIMARGS

The `ELIMARGS` parameter specifies any necessary command line arguments for the external LIM. This parameter is ignored if no external load indices are configured.

Default: none

6 LSF Base Configuration Reference

EXINTERVAL

The time interval (in seconds) at which the LIM daemons exchange load information. On extremely busy hosts or networks, 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.

Default: 15 seconds

ELIM_POLL_INTERVAL

The 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

HOST_INACTIVITY_LIMIT

An integer reflecting a multiple of `EXINTERVAL`. This parameter controls the maximum time a slave LIM will take to send its load information to the master LIM as well as the frequency at which the master LIM will send a heartbeat message to its slaves. A slave LIM can send its load information any time from `EXINTERVAL` to `(HOST_INACTIVITY_LIMIT-2)*EXINTERVAL` seconds. A master LIM will send a master announce to each host at least every `EXINTERVAL*HOST_INACTIVITY_LIMIT` seconds.

Default: 5

MASTER_INACTIVITY_LIMIT

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 + hostNo*MASTER_INACTIVITY_LIMIT)*EXINTERVAL` seconds where *hostNo* is the position of the host in the `lsf.cluster.cluster` file.

Default: 2

PROBE_TIMEOUT

Before taking over as the master, a slave LIM will try to connect to the last known master via TCP. This parameter specifies the time-out in seconds to be used for the `connect(2)` system call.

Default: 2 seconds

RETRY_LIMIT

An integer reflecting a multiple of `EXINTERVAL`. This parameter controls the number of retries a master (slave) LIM makes before assuming the slave (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 the master is down.

Default: 2

LSF Administrators

The `ClusterAdmins` section defines the LSF administrator(s) for this cluster. Both UNIX user and group names may be specified with the `ADMINISTRATORS` keyword. The LIM will expand the definition of a group name using the `getgrnam(3)` call. 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`. If the primary administrator is changed, make sure the owner of the configuration files and the files under `LSB_SHAREDIR/cluster` are changed as well. All LSF administrators have the same authority to perform actions on LSF daemons, jobs, queues, or hosts in the system.

For backwards compatibility, `ClusterManager` and `Manager` are synonyms for `ClusterAdmins` and `ADMINISTRATOR` respectively. It is possible to have both sections present in the same `lsf.cluster.cluster` file to allow daemons from different LSF versions to share the same file.

If this section is not present, the default LSF administrator is `root`. For flexibility, each cluster may have its own LSF administrator(s), identified by a user name, although the same administrator(s) can be responsible for several clusters.

6 LSF Base Configuration Reference

The `ADMINISTRATOR` parameter is normally set during the installation procedure.

Use the `-l` option of the `lsclusters(1)` command to display all the administrators within a cluster.

The following gives an example of a cluster with three LSF administrators. The user listed first, `user2`, is the primary administrator.

```
Begin ClusterAdmins
ADMINISTRATORS = user2 lsfgrp user7
End ClusterAdmins
```

Hosts

The `Host` section is the last section in `lsf.cluster.cluster` 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. 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.

Since the master LIM makes all placement decisions for the cluster, you want it on a fast machine. Also, to avoid the delays involved in switching masters if the first machine goes down, you want the master to 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. Both types are listed below.

Descriptive Fields

The `HOSTNAME`, `model`, `type`, and `RESOURCES` fields must be defined in the `Host` section. The `server`, `nd`, `RUNWINDOW` and `REXPRI` fields are optional.

HOSTNAME

The official name of the host as returned by `hostname(1)`. Must be listed in `lsf.shared` as belonging to this cluster.

model

Host model. Must be one of those defined in the `lsf.shared` file. This determines the CPU speed scaling factor applied in load and placement calculations.

type

A host type as defined in the `HostType` section of `lsf.shared`. The strings used for host types are decided by the system administrator. For example, `SPARC`, `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 then 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 `SUN41` might be used for any computer with a SPARC processor running SunOS 4.1. This would include many Sun models and quite a few from other vendors as well.

server

1 if the host can receive jobs from other hosts, 0 otherwise. If `server` is set to 0, the host is an LSF client. Client hosts do not run the LSF daemons. Client hosts can submit interactive and batch jobs to an LSF cluster, but cannot execute jobs sent from other hosts. If this field is not defined, then the default is 1.

nd

The 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

6 LSF Base Configuration Reference

RESOURCES

Boolean resources available on this host. The resource names are strings defined in the `Resource` section of the file `lsf.shared`. You may list any number of resources, enclosed in parentheses and separated by blanks or tabs. For example, `(fs frame hpux)`.

RUNWINDOW

Dispatch window during which the LIM recommends this host for task execution. 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. Note that LSF Batch uses its own (optional) host dispatch windows to control batch job processing on batch server hosts.

A dispatch window consists of one or more time windows. See *'How LSF Batch Schedules Jobs'* on page 19 for a description of the format of time window specifications.

Default: always accept remote jobs

UNIX

REXPRI

The default execution priority for interactive remote jobs run under the `RES`. Range: -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

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.

Note

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 you wish to use for determining whether hosts are busy. Fields that are not configured are not considered when determining host status.

Thresholds can be set for any load index supported internally by the LIM, and for any external load index (see *'Load Thresholds' on page 216*).

This example `Host` section contains descriptive and threshold information for two hosts.

```
Begin Host
HOSTNAME model      type  server rlm pg tmp RESOURCES      RUNWIN
DOW
hostA     SparcIPC Sparc   1  3.5 15   0 (sunos)      ( )
hostD     Sparc10  Sparc   1  3.5 15   0 (sunos frame) (18:00
-08:00)
End Host
```

Resource Map

`ResourceMap` section is needed when you define 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` file, 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 `ResourceMap` section, you can define resources that are shared by all hosts in the cluster, or resources that are shared by only some of the hosts in the cluster.

This section must appear after the `Host` section of the `lsf.cluster.cluster` file because it has a dependency on host names defined in the `Host` section. The following parameters must be defined in the `ResourceMap` section:

RESOURCENAME

The name of the resource. This resource name must be defined in the `Resource` section of the `lsf.shared` file.

6 LSF Base Configuration Reference

LOCATION

This defines the hosts that share the resource. For a static resource, the value must be defined here as well. The syntax is:

```
(value@[instance] ...) ...
```

You must 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.

`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@[apple] 4@[others]
```

Indicates that there are 2 units of the resource on `apple`, 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.

The `ResourceMap` section may be specified as demonstrated in the following example:

```
Begin ResourceMap
RESOURCENAME  LOCATION
verilog       [all]
```

```
local          ([apple orange] [others])  
End ResourceMap
```

The resources, "verilog" and "synopsys" must already have been defined in the RESOURCE section of the `lsf.shared` file. "verilog" is a static numeric resource shared by all hosts. The value for `verilog` is 5. "local" is a numeric shared resource that contains two instances in the cluster. The first instance is shared by two machines, `apple` and `orange`. The second instance is shared by all other hosts.

Resources defined in the `ResourceMap` section can be viewed by "-s" option of the `lshosts` (for static resource) and `lsload` (for dynamic resource) commands.

The `lsf.task` and `lsf.task.cluster` Files

Users should not have to specify a resource requirement each time they submit a job. LSF supports the concept of a task list.

A task list is a list maintained by LSF that keeps track of the default resource requirements for different applications. 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 together with the 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, then 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.

LSF's task list can be configured at three levels: a system-wide task list that applies to all clusters and all users, a cluster-wide task list that applies to all users in the same cluster, and a user task list that applies only to the user. The system-wide task list and the cluster-wide task list are configured by the `lsf.task` and `lsf.task.cluster` files and are only modified by the cluster administrator. The user-specific task list is maintained in the `.lsftask` file in the user's home directory. Users use the `lsrtasks` command to manipulate his/her own task list.

6 LSF Base Configuration Reference

LSF combines the system-wide, cluster-wide, 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 name in different lists, the cluster-wide list overrides system-wide list, and user-specific list overrides both.

Each task list file contains a `RemoteTasks` section that maps task names to resource requirements, one task per line. Each line in the section is an entry consisting of a task name and a resource requirement string separated by a slash `/`. A plus sign `+` or a minus sign `-` can optionally precede each entry. If no `+` or `-` is specified, then `+` 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.

Below is an example of a task list file:

```
Begin RemoteTasks
+ "newjob/mem>25"
+ "verilog/select[type==any && swp>100]"
+ "f77/type==any"
+ "compressdir/fs"
End RemoteTasks
```

The hosts File

If your LSF clusters include hosts that have more than one interface and are configured with more than one official host name, you must either modify the host name configuration or create a private hosts file for LSF to use. The LSF hosts file is stored in `LSF_CONFDIR`. The format of `LSF_CONFDIR/hosts` is the same as for the `/etc/hosts` file.

For every host that has more than one official name, you must duplicate the `hosts` database information except that all entries for the host should use the same official name. Configure all the other names for the host as aliases so that people can still refer to the host by any name. For example, if your `/etc/hosts` file contains

```
AA.AA.AA.AA  host-AA host # first interface
BB.BB.BB.BB  host-BB      # second interface
```

then the `LSF_CONFDIR/hosts` file should contain:

```
AA.AA.AA.AA  host host-AA # first interface
BB.BB.BB.BB  host host-BB # second interface
```

The LSF hosts file should only contain entries for host with more than one official name. All other hosts names and addresses are resolved using the default method for your hosts. See *'Hosts, Machines, and Computers'* on page 3 for a detailed discussion of official host names.

The `lsf.sudoers` File

The format of this file is very similar to that of the `lsf.conf` file (see *'The lsf.conf File'* on page 161). Each line of the file is a `NAME=VALUE` statement, where `NAME` describes an authorized operation and `VALUE` is a single string or multiple strings enclosed in quotes. On UNIX, lines starting with '#' are comments and are ignored. On UNIX, the `lsf.sudoers` file is optional.

On Windows NT, except for the `LSF_LOCAL_ADMIN_GROUP` variable, the parameters described in the `lsf.sudoers` file are described in a Registry key located at `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\LSF Service\lsf.sudoers`.

The following variables are defined:

`LSF_LOCAL_ADMIN_GROUP`

Windows NT only. This is a Registry key that defines the local LSF administrators group. Members of this user group are assigned privileges that allow them to start and stop the LSF services.

The location of this value in the Registry is:

`HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\LSF Service`

Default value: LSF Local Admins

6 LSF Base Configuration Reference

LSF_STARTUP_USERS

UNIX only. This variable is equivalent to the local LSF administrators group in Windows NT, and it enables a list of specified users to start LSF daemons using the LSF administration commands `lsadmin` and `badmin`. By default, `root` is the only user who can start up the LSF daemons as `root`, and `lsadmin` and `badmin` must be installed as *setuid root* programs.

```
LSF_STARTUP_USERS="user1 user2"
```

This allows listed users to perform the startup operations. If this list contains only one user, quotes are not necessary.

```
LSF_STARTUP_USERS = all_admins
```

This allows all the LSF administrators configured in the `lsf.cluster.cluster` file to start up LSF daemons using the `lsadmin` and `badmin` commands.

CAUTION!

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`.

LSF_STARTUP_PATH

The absolute pathname of the directory where the server binaries, namely `lim`, `res`, `sbatchd`, are installed. This is normally `LSF_SERVERDIR` as defined in your `lsf.conf` file. LSF will allow the specified administrators (see `LSF_STARTUP_USERS` or `LSF_LOCAL_ADMIN_GROUP`) to start the daemons installed in the `LSF_STARTUP_PATH` directory.

On UNIX, both `LSF_STARTUP_USERS` and `LSF_STARTUP_PATH` must be defined for this feature to work.

LSB_PRE_POST_EXEC_USER

This parameter defines the authorized user for the LSF Batch queue level pre-execution and post-execution commands. These commands can be configured at the queue level by the LSF administrator. If `LSB_PRE_POST_EXEC_USER` is defined, the queue level pre-execution and post-execution commands will be run as the user defined. If this parameter is not defined, the commands will be

run as the user who submitted the job. In particular, you can define this variable if you need to run commands as root on UNIX.

See *'Pre- and Post-execution Commands'* on page 36 for details of pre-execution and post-execution.

You can only define a single username in this parameter.

LSF_EAUTH_USER

This defines the username to run the external authentication executable, `eauth`. If this parameter is not defined, then `eauth` will be run as the primary LSF administrator. See *'External Authentication'* on page 11 for an explanation of external authentication.

LSF_EAUTH_KEY

This defines a key the `eauth` uses to encrypt and decrypt the user authentication data. If this parameter is not defined, then `eauth` will encrypt and decrypt authentication data using an internal key.

This parameter gives the user site a chance to improve their security. The rule of choosing the key is same as choosing the password. If you want to change the key, you should modify the `lsf.sudoers` file on every host. For the hosts to work together, they must all use the same key.

See *'External Authentication'* on page 11 for an explanation of external authentication.

LSF_EEXEC_USER

This defines the user name to run the external execution command, `eexec`. If this parameter is not defined, then `eexec` will be run as the user who submitted the job. See *'External Submission and Execution Executables'* on page 42 for an explanation of external execution.

6 LSF Base Configuration Reference

7. LSF Batch Configuration Reference

This chapter describes the LSF Batch configuration files `lsb.params`, `lsb.users`, `lsb.hosts`, and `lsb.queues`. These files use the same horizontal and vertical section structure as the LIM configuration files (see ‘*Configuration File Formats*’ on page 52). All LSF Batch configuration files are found in the `LSB_CONFDIR/cluster/configdir` directory.

The `lsb.params` File

The `lsb.params` file defines general parameters used by the LSF Batch cluster. This file contains only one section.

Most of the parameters that can be defined in the `lsb.params` file control timing within the LSF Batch system. The default settings provide good throughput for long running batch jobs while adding a minimum of processing overhead in the batch daemons.

Parameters

This section and all the keywords in this section are optional. If keywords are not present, LSF Batch assumes default values for the corresponding keywords. The valid keywords for this section are:

`DEFAULT_QUEUE = queue ...`

`DEFAULT_QUEUE` lists the names of LSF Batch queues defined in the `lsb.queues` file. When a user submits a job to the LSF Batch system without explicitly specifying a queue and the user’s environment variable `LSB_DEFAULTQUEUE` is not set, LSF Batch queues the job in the first default queue listed that satisfies the job’s specifications and other restrictions.

7 LSF Batch Configuration Reference

If this keyword is not present or no valid value is given, then LSF Batch automatically creates a default queue named `default` with all the default parameters (see *'The lsb.queues File' on page 208*).

`DEFAULT_HOST_SPEC = host_spec`

`host_spec` must be a host name defined in the `lsf.cluster.cluster` file, or a host model defined in the `lsf.shared` file.

The CPU time limit defined by the `CPULIMIT` parameter in the `lsb.queues` file or by the user through the `-c cpu_limit` option of the `bsub` command is interpreted as the maximum number of minutes of CPU time that a job may run on a host of the default specification. When a job is dispatched to a host for execution, the CPU time limit is then normalized according to the execution host's CPU factor.

If `DEFAULT_HOST_SPEC` is defined in both the `lsb.params` file and the `lsb.queues` file for an individual queue, the value specified for the queue overrides the global value. If a user explicitly gives a host specification with the CPU limit when submitting a job, the user specified host or host model overrides the values defined in both the `lsb.params` and the `lsb.queues` files.

Default: the fastest batch server host in the cluster.

`DEFAULT_PROJECT = proj_name`

The default project name for jobs. When a user submits a job without specifying any project name, and the user's environment variable `LSB_DEFAULTPROJECT` is not set, LSF Batch automatically assigns the job to this default project name. On IRIX 6, the project name must be one of the projects listed in the `/etc/project(4)` file. On all other platforms, the project name is a string used for accounting purposes.

Default: If this parameter is not present, LSF Batch uses `default` as the default project name.

`MBD_SLEEP_TIME = integer`

The LSF Batch job dispatching interval. It determines how often the LSF Batch system tries to dispatch pending batch jobs.

Default: 60 (seconds).

`SBD_SLEEP_TIME = integer`

The LSF Batch job checking interval. It determines how often the LSF Batch

system checks the load conditions of each host to decide whether jobs on the host must be suspended or resumed.

Default: 30 (seconds).

`JOB_ACCEPT_INTERVAL` = *integer*

The number of `MBD_SLEEP_TIME` periods to wait after dispatching a job to a host, before dispatching a second job to the same host. If

`JOB_ACCEPT_INTERVAL` is zero, a host may accept more than one job in each job dispatching interval (`MBD_SLEEP_TIME`).

Default: 1.

`MAX_SBD_FAIL` = *integer*

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 the master batch daemon fails to reach a host, and has retried `MAX_SBD_FAIL` times, the host is considered unavailable. When a host becomes unavailable the `mbatchd` assumes that all jobs running on that host have exited, and all rerunnable jobs (jobs submitted with the `bsub -r` option) are scheduled to be rerun on another host.

Default: 3.

`CLEAN_PERIOD` = *integer*

The amount of time that job records for jobs that have finished or have been killed are kept in-core in the master batch daemon 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 (seconds).

`MAX_JOB_NUM` = *integer*

The maximum number of finished jobs whose events are to be stored in an event log file (see the `lsb.events(5)` manual page). Once the limit is reached, the `mbatchd` switches the event log file. See ‘*LSF Batch Event Log*’ on page 80.

Default: 1000.

`HIST_HOURS` = *integer*

The number of hours of resource consumption history taken into account when calculating the priorities of users in a host partition (see ‘*Host Partitions*’

7 LSF Batch Configuration Reference

on page 206) or a fairshare queue (see ‘*The lsb.queues File*’ on page 208). This parameter is meaningful only if a fairshare queue or a host partition is defined. In calculating a user’s priority, LSF Batch uses a decay factor which scales the CPU time used by the user’s jobs such that 1 hour of CPU time used is equivalent to 0.1 hour after HIST_HOURS have elapsed.

Default: five (hours).

PG_SUSP_IT = *integer*

The time interval (in seconds) 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).

MAX_JOB_ARRAY_SIZE

The `MAX_JOB_ARRAY_SIZE` parameter is set in the `in lsb.params` file. This parameter specifies the maximum size of a job array that can be created by a user for a single job submission. A large job array allows a user to submit a large number of jobs to the system with a single job submission.

Maximum value: 2046 (jobs)

Default value: 1000 (jobs)

UNIX

JOB_TERMINATE_INTERVAL

This parameter specifies the time interval 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 (seconds).

CPU_TIME_FACTOR

Weighting factor for the CPU time consumed by a user in calculating that user’s fairshare priority in a fairshare queue or host partition.

Default: 0.7.

`RUN_TIME_FACTOR`

Weighting factor for the run time consumed by a user in calculating the user's fairshare priority in a fairshare queue or host partition.

Default: 0.7.

`RUN_JOB_FACTOR`

Weighting factor for the number of job slots used or reserved by a user in calculating the user's fairshare priority in a fairshare queue or host partition.

Default: 3.0.

Handling Cray NQS Incompatibilities

Cray NQS is incompatible with some of the public domain versions of NQS. Even worse, different versions of NQS on Cray are incompatible with each other. If your NQS server host is a Cray, some additional parameters may be needed for LSF Batch to understand the NQS protocol correctly.

If the NQS version on a Cray is NQS 80.42 or NQS 71.3, then no extra setup is needed. For other versions of NQS on a Cray, you need to define `NQS_REQUESTS_FLAGS` and `NQS_QUEUES_FLAGS`.

`NQS_REQUESTS_FLAGS` = *integer*

If the version is NQS 1.1 on a Cray, the value of this flag is 251918848.

For other versions of NQS on a Cray, see '*Handling Cray NQS Incompatibilities*' on page 273 to get the value for this flag.

`NQS_QUEUES_FLAGS` = *integer*

See '*Handling Cray NQS Incompatibilities*' on page 273 to get the value for this flag. This flag is used by LSF Batch to get the NQS queue information.

The `lsb.users` File

The `lsb.users` file contains configuration information about individual users and groups of users in an LSF Batch cluster. This file is optional.

UNIX/NT User Groups

User groups defined by UNIX/NT often reflect certain relationships among users. It is natural to control computer resource access using UNIX/NT user groups.

You can specify a UNIX/NT group anywhere an LSF Batch user group can be specified. On UNIX groups recognized by LSF Batch are the groups that are returned by a `getgrnam(3)` call. Note that only group members listed in the `/etc/group` file or the `group.byname` NIS map are accepted; the user's primary group as defined in the `/etc/passwd` file is ignored. On NT, groups are obtained from the primary domain controller.

If both an individual user and a UNIX/NT group have the same name, LSF assumes that the name refers to the individual user. In this case you can specify the UNIX/NT group name by appending a slash `'/'` to the group name. For example, if you have both a user and a group named `admin` on your system, LSF interprets `admin` as the name of the user, and `admin/` as the name of the group.

Limitations

Although it is convenient to use UNIX groups as LSF Batch user groups, it may produce unexpected results if the UNIX group definitions are not homogeneous across machines. The UNIX groups picked up by LSF Batch are the groups obtained by calling `getgrnam(3)` on the master host. If the master host later changes to another host, the groups picked up might be different.

This will not be a problem if all the UNIX user groups referenced by LSF Batch configuration files are uniform across all hosts in the LSF cluster.

LSF Batch User Groups

A user group is a group of users with a name assigned. User groups can be used in defining the following parameters in LSF Batch configuration files:

- `USERS` in the `lsb.queues` file for authorized queue users.
- `USER_NAME` in the `lsb.users` file for user job slot limits.
- `USER_SHARES` (optional) in the `lsb.hosts` file for host partitions or in the `lsb.queues` file for queue fairshare policies.

The optional `UserGroup` section begins with a line containing the mandatory keywords `GROUP_NAME` and `GROUP_MEMBER`. Each subsequent line defines a single group. The first word on the line is the group name. The rest of the line contains a list of group members, enclosed in parentheses and separated by white space. A group can be included in another group; this means that every member of the first group is also a member of the second.

A user or group can be a member of more than one group. The reserved name `all` can be used to specify all users.

```
Begin UserGroup
GROUP_NAME   GROUP_MEMBER
eng_users    (user1 user4 user5 user6)
tech_users   (eng_users user7)
acct_users   (user2 user3 user1)
End UserGroup
```

Share Tree Defined in User Groups

User groups in LSF Batch can also be configured in a hierarchical way to form a share tree for hierarchical fairshare purpose. See *'Hierarchical Fairshare'* on page 117 for the concept of hierarchical fairshare.

To configure a share tree, the keyword `USER_SHARES` can be used in the `UserGroup` section. The `USER_SHARES` parameter is a list of `[name, shares]` pairs, where `name` is a user name or user group name. `shares` is a positive integer specifying the number of shares this user or user group has.

`shares` determine the static priority of users or user groups relative to each other. So the values of `shares` only make relative sense. The share tree defined in this section has no effect unless it is actually used by a share provider, such as a queue or host partition.

7 LSF Batch Configuration Reference

An example of a share tree configuration is described in ‘*Understanding How Fairshare Works*’ on page 119.

External User Groups

LSF Batch supports the notion of an external user group. An external user group is a user group whose membership users are not statically configured, but are instead retrieved by running an external executable `egroup` in the directory specified by `LSF_SERVERDIR`. This feature allows a site to maintain group definitions outside of LSF and imported them into LSF Batch configuration at initialization time.

`egroup` is invoked with the arguments "`-u user_group_name`" and is run as the LSF administrator during `mbatchd` startup time. `egroup` must write the user names for the group to its standard output, with each name being separated by white space.

To tell LSF Batch that the group members should be retrieved using `egroup`, simply put `(!` in the `GROUP_MEMBER` column of the configuration file. For example:

```
Begin UserGroup
GROUP_NAME      GROUP_MEMBER
regular_users   (user1 user2 user3 user4)
part_time_users (!)
End UserGroup
```

User and Group Job Slot Limits

Each user or user group can have a cluster-wide job slot limit and a per-processor job slot limit. These limits apply to the total number of job slots used by batch jobs owned by the user or group, in all queues. LSF Batch only dispatches the specified number of jobs at one time; if the user submits too many jobs, they remain pending and other users' jobs are run if hosts are available.

Detailed descriptions about job slot limits and how they are enforced by LSF Batch are described in ‘*User Job Slot Limits*’ on page 27.

If a job slot limit is specified for a user group, the total number of job slots used by all users in that group are counted. If a user is a member of more than one group, each of that user's jobs is counted against the limit for all groups to which that user belongs.

This file can also contain a `User` section. The first line of this section gives the keywords that apply to the rest of the lines. The possible keywords include:

`USER_NAME`

Name of a user or user group. This keyword is mandatory. If the name is a group name and the name is appended with an '@', the job slot limits defined apply to each user in that group, as you could otherwise do by listing each user in that group in separate entries in this section.

`MAX_JOBS`

System-wide job slot limits. This limits the total number of job slots this user or user group can use at any time.

`JL/P`

Per processor job slot limit. This limits the maximum number of job slots this user or user group can use per processor. 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 multi-processor host, the user can only use up to 2 job slots at any time. On a uni-processor machine, the user can use 1 job slot.

The reserved user name `default` can be used for `USER_NAME` to set a limit for each user or group not explicitly named. If no default limit is specified, users and groups not listed in this section can run an unlimited number of jobs.

The default per-user job slot limit also applies to groups. If you define groups with many users, you may need to configure a job slot limit for that group explicitly to override the default setting.

```
Begin User
USER_NAME  MAX_JOBS  JL/P
user3      10         -
user2      4          1
eng_users@ 10         1
default    6          1
End User
```

The `lsb.hosts` File

The `lsb.hosts` file contains host related configuration information for the batch server hosts in the cluster. This file is optional.

Host Section

The optional `Host` section contains per-host configuration information. Each host, host model or host type can be configured to run a maximum number of jobs and a limited number of jobs for each user. Hosts, host models or host types can also be configured to run jobs only under specific load conditions or time windows.

If no hosts, host models or host types are named in this section, LSF Batch uses all hosts in the LSF cluster as batch server hosts. Otherwise, only the named hosts, host models and host types are used by LSF Batch. If a line in the `Host` section lists the reserved host name `default`, LSF Batch uses all hosts in the cluster and the settings on that line apply to every host not referenced in the section, either explicitly or by listing its model or type.

Note

When you modify the cluster by adding or removing hosts, no changes are made to the `lsb.hosts` file. 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.

The first line of this section gives the keywords that apply to the rest of the lines. The keyword `HOST_NAME` must appear. Other supported keywords are optional.

`HOST_NAME`

The name of a host defined in the `lsf.cluster.cluster` file, a host model or host type defined in the `lsf.shared` file, or the reserved word `default`.

`MXJ`

The maximum number of job slots for the host. On multiprocessor hosts `MXJ` should be set to at least the number of processors to fully use the CPU resource.

Default: unlimited.

JL/U

The maximum number of job slots any single user can use on this host at any time. See *'Job Slot Limits'* on page 26 for details of job slot limits.

Default: unlimited.

DISPATCH_WINDOW

Times when this host will accept batch jobs.

Dispatch windows are specified as a series of time windows. See *'How LSF Batch Schedules Jobs'* on page 19 for detailed format of time windows.

Default: always open.

Note

Earlier versions of LSF used the keyword RUN_WINDOW instead of DISPATCH_WINDOW in the `lsb.hosts` file. This keyword is still accepted to provide backward compatibility.

MIG

Migration threshold in minutes. If a checkpointable or rerunable job dispatched to this host is suspended for more than MIG minutes, the job is migrated. The suspended job is checkpointed (if possible) and killed. Then LSF restarts or reruns the job on another suitable host if one is available. If LSF is unable to rerun or restart the job immediately, the job reverts to PENDING status and is queued with a higher priority than any other submitted job, so it is rerun or restarted before other queued jobs are dispatched.

Each LSF Batch queue can also specify a migration threshold. Jobs are migrated if either the host or the queue specifies a migration threshold. If MIG is defined both here and in `lsb.queues`, the lower threshold is used.

Jobs that are neither checkpointable nor rerunable are not migrated.

Default: no automatic migration.

r15s, r1m, r15m, ut, pg, io, ls, it, tmp, swp, mem, *name*

Scheduling and suspending thresholds for the dynamic load indices supported by LIM, including external load index names. 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. See *Section 4*,

7 LSF Batch Configuration Reference

'Resources', beginning on page 35 of the *LSF Batch User's Guide* for complete descriptions of the load indices.

Each LSF Batch queue also can specify scheduling and suspending thresholds in `lsb.queues`. If both files specify thresholds for an index, those that apply are the most restrictive ones apply.

Default: no threshold.

CHKPNT

Defines the form of checkpointing available. Currently, only the value 'c' is accepted. This indicates that checkpoint copy is supported. With checkpoint copy, all opened files are automatically copied to the checkpoint directory by the operating system when a process is checkpointed. Checkpoint copy is currently supported only on ConvexOS.

Default: no checkpoint copy.

The keyword line should name only the load indices that you wish to configure on a per-host basis. Load indices not listed on the keyword line do not affect scheduling decisions.

Each following line contains the configuration information for one host, host model or host type. This line must contain one entry for each keyword on the keywords line. Use empty parentheses '()' or a dash '-' to specify the default 'don't care' value for an entry. The entries in a line for a host override the entries in a line for its model or type.

```
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
```

This example `Host` section shows host-specific configuration for a host and a host type, along with default values for all other load-sharing hosts. Host `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. Host `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. `SUNSOL` must be a host type defined in the `lsf.shared` file.

The entry with host name `default` applies to each of the other hosts in the LSF 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.

Host Groups

The `HostGroup` section is optional. This section defines names for sets of hosts. The host group name can then be used in other host group, host partition, and batch queue definitions, as well as on an LSF Batch command line. When a host group name is used, it has exactly the same effect as listing all of the host names in the group.

Host groups are specified in the same format as user groups in the `lsb.users` file.

The host group section must begin with a line containing the mandatory keywords `GROUP_NAME` and `GROUP_MEMBER`. Each other line in this section must contain an alphanumeric string for the group name, and a list of host names or previously defined group names enclosed in parentheses and separated by white space.

Host names and host group names can appear in more than one host group. The reserved name `all` specifies all hosts in the cluster.

```
Begin HostGroup
GROUP_NAME  GROUP_MEMBER
licencel   (hostA hostD)
sys_hosts  (hostF licensel hostK)
End HostGroup
```

This example section defines two host groups. The group `licensel` contains the hosts `hostA` and `hostD`; the group `sys_hosts` contains `hostF` and `hostK`, along with all hosts in the group `licensel`. Group names must not conflict with host names.

External Host Groups

LSF Batch allows host membership to be maintained outside LSF Batch and imported into LSF Batch configuration at initialization time. An executable `egroup` in the `LSF_SERVERDIR` directory is invoked to obtain the list of members for a given host group. The `egroup` should be invoked with argument `"-m host_hgroup_name"`. The group members, separated by spaces, should be written to the standard output stream of `egroup`. The `egroup` is run as the LSF administrator during `mbatchd` startup time.

To tell LSF Batch that the group membership should be retrieved via `egroup`, simply put `"!"` in the `GROUP_MEMBER` column of the `HostGroup` section. For example:

```
Begin HostGroup
GROUP_NAME      GROUP_MEMBER
Big_servers     (!)
desk_tops       (host1 host2 host3 host4)
End HostGroup
```

Host Partitions

The `HostPartition` section is optional, and you can configure more than one such section. See *'Controlling Fairshare'* on page 113 for more discussions of fairshare and host partitions.

Each `Host Partition` section contains a list of hosts and a list of user shares. Each host can be named in at most one host partition. Hosts that are available for batch jobs, but not included in any host partition are shared on a first-come, first-served basis. The special host name `all` can be specified to configure a host partition that applies to all hosts in a cluster.

Each user share contains a single user name or user group name, and an integer defining the shares available to that user. The special user name `'others'` can be used to configure total shares for all users not explicitly listed. The special name `'default'` configures the default per-user share for each user not explicitly named. Only one of `others` or `default` may be configured in a single host partition.

Note

Host partition fairshare scheduling is an alternative to queue level fairshare scheduling. You cannot use both in the same LSF cluster.

The following example shows a host partition applied to hosts *hostA* and *hostD*:

```
Begin HostPartition
HPART_NAME = part1
HOSTS = hostA hostD
USER_SHARES = [eng_users, 7] [acct_users, 3] [others, 1]
End HostPartition
```

In the example, the total of all the shares is $7 + 3 + 1 = 11$. This host partition specifies that all users in the user group *eng_users* should get 7/11 of the resources, the *acct_users* group should get 3/11, and all other users together get 1/11.

Note that the shares for a group specify the total share for all users in that group, unless the group name has a trailing '@'. In this case, the share is for each individual user in the group. If you want to further divide shares allocated to a group among group members, you can define a share tree for the group in the *lsb.users* file. See 'Hierarchical Fairshare' on page 117 for more details.

Fairshare is only enforced when jobs from more than one user or group are pending. If only one user or group is submitting jobs, those jobs can take all the available time on the partitioned hosts. If another user or group begins to submit jobs, those jobs are dispatched first until the shares reach the configured proportion.

The following example shows a host partition that gives users in the *eng_users* group very high priority, but allows jobs from other users to run if there are no jobs from the *eng_users* group waiting:

```
Begin HostPartition
HPART_NAME = eng
Hosts = all
User_Shares = ([eng_users, 500] [others, 1])
End HostPartition
```

Hosts belonging to a host partition should not be configured in the *HOSTS* parameter of a queue together with other hosts not belonging to the same host partition. Otherwise, the following two limitations may apply:

7 LSF Batch Configuration Reference

- 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.

The `lsb.queues` File

The `lsb.queues` file contains definitions of the batch queues in an LSF cluster. This file is optional. If no queues are configured, LSF Batch creates a queue named `default`, with all parameters set to default values (see the description of `DEFAULT_QUEUE` in ‘*The lsb.params File*’ on page 193).

Queue definitions are horizontal sections that begin with the line `Begin Queue` and end with the line `End Queue`. You can define at most 40 queues in an LSF Batch cluster. Each queue definition contains the following parameters:

General Parameters

`QUEUE_NAME` = *string*

The name of the queue. This parameter must be defined, and has no default. The queue name can be any string of non-blank characters up to 40 characters long. It is best to use 6 to 8 character names made up of letters, digits, and possibly underscores ‘_’ or dashes ‘-’.

`PRIORITY` = *integer*

This parameter indicates the priority of the queue relative to other LSF Batch queues. Note that this is an LSF Batch dispatching priority, completely independent of the UNIX scheduler’s priority system for time-sharing processes. The LSF Batch `NICE` parameter is used to set the UNIX time-sharing priority for batch jobs.

LSF Batch tries to schedule jobs from queues with larger `PRIORITY` values first. This does not mean that jobs in lower priority queues are not scheduled unless higher priority queues are empty. Higher priority queues are checked

first, but not all jobs in them are necessarily scheduled. For example, a job might be held because no machine with the right resources is available, or all jobs in a queue might be held because the queue's dispatch window or run window (see below) is closed. Lower priority queues are then checked and, if possible, their jobs are scheduled.

If more than one queue is configured with the same `PRIORITY`, LSF Batch schedules jobs from all these queues in first-come, first-served order.

Default: 1.

`NICE` = *integer*

Adjusts the UNIX scheduling priority at which jobs from this queue execute. The default value of 0 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.

Default: 0.

`QJOB_LIMIT` = *integer*

Job slot limit for the queue. This limits the total number of job slots that this queue can use at any time.

Default: unlimited.

`UJOB_LIMIT` = *integer*

Per user job slot limit for the queue. This limits the total number of job slots any user of this queue can use at any time.

Default: unlimited.

`PJOB_LIMIT` = *float*

Per processor job slot limit. This limits the total number of job slots this queue can use on any processor at any time. This limit is configured per processor so that multiprocessor hosts automatically run more jobs.

Default: unlimited.

`HJOB_LIMIT` = *integer*

Per host job slot limit. This limits the total number of job slots this queue can use on any host at any time. This limit is configured per host regardless of the number of processors it may have. This may be useful if the queue dispatches jobs which require a node-locked license. If there is only one node-locked

7 LSF Batch Configuration Reference

license per host then the system should not dispatch more than one job to the host even if it is a multiprocessor host. For 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.

`RUN_WINDOW = string`

The time windows in which jobs are run from this queue. Run windows are described in *'How LSF Batch Schedules Jobs'* on page 19.

When the queue run window closes, the queue stops dispatching jobs and suspends any running jobs in the queue. Jobs suspended because the run window closed are restarted when the window reopens. Suspended jobs also can be switched to a queue with run its window open; the job restarts as soon as the new queue's scheduling thresholds are met.

Default: always open.

`DISPATCH_WINDOW = string`

The time windows in which jobs are dispatched from this queue. Once dispatched, jobs are no longer affected by the dispatch window. Queue dispatch windows are analogous to the host dispatch windows described on page 203.

Default: always open.

`ADMINISTRATORS = name . . .`

A list of queue-level administrators. The list of names can include any valid user name in the system, any UNIX user group name, and any user group name configured in the `lsb.users` file. Queue administrators can perform operations on any job in the queue as well as on the queue itself (for example, open/close, activate/deactivate). Switching a job from one queue to another requires the administrator to be authorized for both the current and the destination queues.

The `bqueues(1)` command with the `-l` option will display configured administrators for each queue.

Default: No queue-level administrators are defined.

Processor Reservation for Parallel Jobs

The processor reservation feature is disabled by default. To enable it, specify the `SLOT_RESERVE` keyword in the queue:

```
Begin Queue
.
PJOB_LIMIT=1
SLOT_RESERVE = MAX_RESERVE_TIME[n]
.
End Queue
```

The value of the keyword is `MAX_RESERVE_TIME[n]` where `n` is a multiple of `MBD_SLEEP_TIME` (`MBD_SLEEP_TIME` is defined in `lsb.params`). `MAX_RESERVE_TIME` controls the maximum time a slot is reserved for a job. It is required to avoid deadlock situations in which the system is reserving job slots for multiple parallel jobs such that none of them can acquire sufficient resources to start. The system will reserve slots for a job until `n*MBD_SLEEP_TIME` minutes. If an insufficient number have been accumulated, all slots are freed and made available to other jobs. The maximum reservation time takes effect from the start of the first reservation for a job and a job can go through multiple reservation cycles before it accumulates enough slots to be actually started.

Backfill Scheduling

A queue can be configured to allow its jobs to backfill by using the `BACKFILL` keyword in the queue:

```
Begin Queue
.
.
BACKFILL=y
.
.
End Queue
```

7 LSF Batch Configuration Reference

If the `BACKFILL` keyword is specified, use either `y` or `Y` to enable backfilling. To disable backfilling without removing the keyword, use `n`, `N`, or leave a blank space after the “=” symbol.

Jobs in a backfill queue can make use of slots reserved by jobs in other queues. The `SLOT_RESERVE` parameter can be used to reserve processors for jobs in a queue. If the backfill queue also specifies the `SLOT_RESERVE` parameter, then backfilling can occur among jobs within the queue.

Restrictions

The following restrictions apply to the backfill policy:

- A backfill queue cannot be preemptable.
- Any preemptive queue whose priority is higher than the backfill queue cannot preempt the jobs in backfill queue.
- A job will not have an estimated start time immediately after the `mbatchd` is reconfigured.

Deadline Constraint Scheduling

LSF Batch can take into account the constraints imposed by deadlines when scheduling. By default, LSF Batch does not start a job that is not expected to finish. There are two deadline constraints which affect a job:

- run window of a queue

When a run window configured for a queue closes, jobs running in the queue are suspended. By default, LSF Batch only schedules jobs that can finish before the queue closes.

- termination time of a job

The job's termination time is absolute, and when the termination time is reached, the job is killed. By default, LSF Batch only schedules jobs that can finish before their termination time.

The amount of time that each job is expected to take is specified by the run limit of the queue (see `RUNLIMIT` on page 218).

The run limit must be defined properly for the scheduling feature to work properly. If the run limit is left at the default value, unlimited time, LSF Batch cannot schedule any jobs to finish within the deadline constraints.

The run limit specifies the maximum amount of time allowed for the job to run, but there is no way to know how long the job will actually take. It may not be appropriate to schedule jobs and calculate deadline constraints using this time limit, since not all jobs will take the maximum amount of time to run.

To override the default behaviour and start all jobs no matter what the run limit is, use the queue level parameter `IGNORE_DEADLINE`. In this example, LSF Batch is configured to schedule jobs in queue *liberal* without observing the deadline constraints.

```
Begin Queue
QUEUE_NAME = liberal
.
IGNORE_DEADLINE=y
.
End Queue
```

Flexible Expressions for Queue Scheduling

LSF Batch provides a variety of possibly overlapping options for configuring job scheduling policies.

Queue-Level Resource Requirement

The condition for dispatching a job to a host can be specified through the queue-level `RES_REQ` parameter. Using a resource requirement string you can specify conditions in a more flexible manner than using the `loadSched` thresholds. For example:

```
RES_REQ= select[((type==ALPHA && r1m < 2.0)|| (type==HPPA && r1m < 1.0))]
```

will allow a queue, which contains `ALPHA` and `HPPA` hosts, to have different thresholds for different types of hosts. Using the `hname` resource in the `RES_REQ` string allows you to set up different conditions for different hosts in the same queue, for example:

```
RES_REQ= select[((hname=hostA && mem > 50)|| (hname==hostB && mem > 100))]
```

7 LSF Batch Configuration Reference

When `RES_REQ` is specified in the queue and no job-level resource requirement is specified, then `RES_REQ` becomes the default resource requirement for the job. This allows administrators to override the LSF default of executing only on the same type as the submission host. If a job level resource requirement is specified together with `RES_REQ`, then a host must satisfy both requirements to be eligible for running the job. Similarly, the `loadSched` thresholds, if specified, must also be satisfied for a host to be eligible.

The `order` and `span` sections of the resource requirement string can also be specified in the `RES_REQ` parameter. These sections in `RES_REQ` are ignored if they are also specified by the user in the job level resource requirement.

Queue Level Resource Reservation

The resource reservation feature allows user's to specify that the system should reserve resources after a job starts. This feature is also available at the queue level.

The queue level resource reservation can be configured as part of the `RES_REQ` parameter. The `RES_REQ` can include a `rusage` section to specify the amount of resources a job should reserve after it is started. For example:

```
Begin Queue
.
RES_REQ = swap>50 rusage[swp=40:duration=5h:decay=1]
.
End Queue
```

If *duration* is not specified, the default is to reserve the resource for the lifetime of the job. If *decay* is specified as 1, then the reserved resource will be linearly decreased over the time specified by *duration*. If *decay* is not specified, then the resource reserved will not decrease over time. See 'Resource Reservation' on page 91 of the *LSF Batch User's Guide* and `lsfintro(1)` for detailed syntax of `rusage` parameter.

Note

The use of `RES_REQ` affects the pending reasons as displayed by `bjobs`. If `RES_REQ` is specified in the queue and the `loadSched` thresholds are not specified the pending reasons for each individual load index will not be displayed.

Suspending Condition

The condition for stopping a job can be specified using a resource requirement string in the queue level `STOP_COND` parameter. If `loadStop` thresholds have been specified, then a job will be suspended if either the `STOP_COND` is `TRUE` or the `loadStop` thresholds are violated. For example, the following will suspend a job based on the idle time for desktop machines and based on availability of swap and memory on compute servers. Note that `cs` is a boolean resource defined in the `lsf.shared` file and configured in the `lsf.cluster.cluster` file to indicate that a host is a compute server:

```
Begin Queue
.
STOP_COND= select[(!cs && it < 5) || (cs && mem < 15 && swap < 50)]
.
End Queue
```

Note

Only the `select` section of the resource requirement string is considered when stopping a job. All other sections are ignored.

The use of `STOP_COND` affects the suspending reasons as displayed by the `bjobs` command. If `STOP_COND` is specified in the queue and the `loadStop` thresholds are not specified, the suspending reasons for each individual load index will not be displayed.

Note

LSF Batch 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`).

Resume Condition

A separate `RESUME_COND` allows you to specify the condition that must be satisfied on a host if a suspended job is to be resumed. If `RESUME_COND` is not defined, then the `loadSched` thresholds are used to control resuming of jobs. The `loadSched` thresholds are ignored if `RESUME_COND` is defined.

Note that only the `select` section of the resource requirement string is considered when resuming a job. All other sections are ignored.

7 LSF Batch Configuration Reference

Load Thresholds

The queue definition can contain thresholds for 0 or more of the load indices. Any load index that does not have a configured threshold has no effect on job scheduling. A description of all the load indices is given in *Section 4, 'Resources', beginning on page 35 of the LSF Batch User's Guide.*

Each load index is configured on a separate line with the format:

```
index = loadSched/loadStop
```

index is the name of the load index, for example `r1m` for the 1-minute CPU run queue length or `pg` for the paging rate. *loadSched* is the scheduling threshold for this load index. *loadStop* is the suspending threshold.

The `loadSched` and `loadStop` thresholds permit the specification of conditions using simple AND/OR logic. For example, the specification:

```
MEM=100/10  
SWAP=200/30
```

translates into a `loadSched` condition of `mem >= 100 && swap >= 200` and a `loadStop` condition of `mem < 10 || swap < 30`. The `loadSched` condition must be satisfied by a host before a job is dispatched to it and also before a job suspended on a host can be resumed. If the `loadStop` condition is satisfied, a job is suspended.

Note

LSF Batch 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 scheduling threshold also defines the host load conditions under which suspended jobs in this queue may be resumed.

When LSF Batch suspends or resumes a job, it invokes the `SUSPEND` or `RESUME` action as described in *'Configurable Job Control Actions' on page 228*. The default `SUSPEND` action is to send signal `SIGSTOP`, while default action for `RESUME` is to send signal `SIGCONT`.

Note

The `r15s`, `r1m`, and `r15m` CPU run queue length conditions are compared to the effective queue length as reported by `lsload -E`, which is normalised for multiprocessor hosts. Thresholds for these parameters should be set at appropriate levels for single processor hosts.

Resource Limits

Batch queues can enforce resource limits on jobs. LSF Batch supports most of the resource limits that the underlying operating system supports. In addition, LSF Batch also supports a few limits that the underlying operating system does not support.

`CPULIMIT = [hour:]minute[/host_spec]`

Maximum CPU time allowed for a job running in this queue. This limit applies to the whole job, no matter how many processes the job may contain. If a job consists of multiple processes, the `CPULIMIT` parameter applies to all processes in a job. 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 less than 30 seconds may be ignored.

The limit is scaled; the job is allowed to run longer on a slower host, so that a job can do roughly the same amount of work no matter what speed of host it is dispatched to.

The time limit is given in the form `[hour:]minute[/host_spec]`. `minute` may be greater than 59. Three and a half hours can be specified either as 3:30, or 210. `host_spec` is shared by `CPULIMIT` and `RUNLIMIT` (see below). It may be a host name or a host model name which is used to adjust the CPU time limit or the wall-clock run time limit. In its absence, the `DEFAULT_HOST_SPEC` defined for this queue or defined for the whole cluster is assumed. If `DEFAULT_HOST_SPEC` is not defined, the LSF Batch server host with the largest CPU factor is assumed.

CPU time limits are normalized by multiplying the `CPULIMIT` parameter by the CPU factor of the specified or default host, and then dividing by the CPU factor of the execution host. If the specified host has a CPU factor of 2 and another host has a factor of 1, then a `CPULIMIT` value of 10 minutes allows jobs on the specified host to run for 10 minutes, and jobs on the slower host to run for 20 minutes ($2 * 10 / 1$). See ‘Host Models’ on page 174 and ‘Descriptive Fields’ on page 182 for more discussion of CPU factors.

7 LSF Batch Configuration Reference

Default: unlimited.

`RUNLIMIT = [hour:]minute[/host_spec]`

Maximum wall clock running time allowed for batch jobs in this queue. Jobs that are in the `RUN` state for longer than `RUNLIMIT` are killed by LSF Batch. `RUNLIMIT` is available on all host types. For an explanation of the form of the time limit, see `CPULIMIT` above.

Default: unlimited.

`FILELIMIT = integer`

The per-process (hard) file size limit (in KB) for all the processes belonging to a job from this queue (see `getrlimit(2)`).

Default: unlimited.

`MEMLIMIT = integer`

The per-process (hard) process resident set size limit (in KB) for all the processes belonging to a job from this queue (see `getrlimit(2)`). The process resident set size limit cannot be set on HP-UX and Sun Solaris 2.x, so this limit has no effect on an HP-UX or a Sun Solaris 2.x machine.

Default: unlimited.

`DATALIMIT = integer`

The per-process (hard) data segment size limit (in KB) for all the processes belonging to a job from this queue (see `getrlimit(2)`). The data segment size limit cannot be set on HP-UX, so this limit has no effect on an HP-UX machine.

Default: unlimited.

`STACKLIMIT = integer`

The per-process (hard) stack segment size limit (in KB) for all the processes belonging to a job from this queue (see `getrlimit(2)`). The stack segment size limit cannot be set on HP-UX, so this limit has no effect on an HP-UX machine.

Default: unlimited.

`CORELIMIT = integer`

The per-process (hard) core file size limit (in KB) for all the processes

belonging to a job from this queue (see `getrlimit(2)`). The core file size limit cannot be set on HP-UX, so this limit has no effect on an HP-UX machine.

Default: unlimited.

`PROCLIMIT = integer`

The maximum number of job slots that can be allocated to a parallel job in the queue. Jobs which request more job slots via the `-n` option of `bsub(1)` than the queue can accept will be rejected.

Default: unlimited.

`PROCESSLIMIT = integer`

This limits the number of concurrent processes that can be part of a job.

Default: unlimited.

`SWAPLIMIT = integer`

The amount of total virtual memory limit (in kilobytes) 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`, then `SIGTERM`, and then `SIGKILL` in sequence. For `CPU_LIMIT`, `SIGXCPU` is sent before `SIGINT`, `SIGTERM`, and `SIGKILL`.

Default: unlimited.

`NEW_JOB_SCHED_DELAY = integer`

This parameter controls when a scheduling session should be started after a new job is submitted. For example:

```
Begin Queue
.
NEW_JOB_SCHED_DELAY=0
.
End Queue
```

If `NEW_JOB_SCHED_DELAY` is 0 seconds, a new scheduling session is started as soon as a job is submitted to this queue. This parameter can be used to obtain faster response times for jobs in a queue such as a queue for interactive jobs.

7 LSF Batch Configuration Reference

Setting a value of 0 can cause the `mbatchd` to be busy if there are a lot of submissions.

Default: 10 seconds.

`JOB_ACCEPT_INTERVAL = integer`

This parameter has the same effect as `JOB_ACCEPT_INTERVAL` defined in `lsb.params` file, except that it applies to this queue.

Default: `JOB_ACCEPT_INTERVAL` defined in `lsb.params` or 1 if it is not defined in `lsb.params` file.

`INTERACTIVE = NO | ONLY`

An interactive job can be submitted via the `-I` option of `bsub` command. By default, a queue would accept both interactive and background jobs. This parameter allows LSF cluster administrator to limit a queue to not accept interactive jobs (`NO`), or to only accept interactive jobs (`ONLY`).

Eligible Hosts and Users

Each queue can have a list of users and user groups who are allowed to submit batch jobs to the queue, and a list of hosts and host groups that restricts where jobs from the queue can be dispatched.

`USERS = name . . .`

The list of users who can submit jobs to this queue. The list of names can include any valid user name in the system, any UNIX user group name, and any user group name configured in the `lsb.users` file. The reserved word `all` may be used to specify all users.

Note

LSF cluster administrator can submit jobs to any queue, even if the login name of the cluster administrator is not defined in the `USERS` parameter of the queue. LSF cluster administrator can also switch a user's jobs into this queue from other queues, even if this user's login name is not defined in the `USERS` parameter.

Default: `all`.

`HOSTS = name[+pref_level] . . .`

The list of hosts on which jobs from this queue can be run. Each name in the list must be a valid host name, host group name or host partition name as

configured in the `lsb.hosts` file. The name can be optionally followed by `+pref_level` to indicate the preference for dispatching a job to that host, host group, or host partition. `pref_level` is a positive number specifying the preference level of that host. If a host preference is not given, it is assumed to be 0.

Hosts at the same level of preference are ordered by load. For example:

```
HOSTS = hostA+1 hostB hostC+1 servers+3
```

where `servers` is a host group name referring to all computer servers. This defines three levels of preferences: run jobs on `servers` as much as possible, or else on `hostA` and `hostC`. Jobs should not run on `hostB` unless all other hosts are too busy to accept more jobs.

If you use the reserved word 'others', it means jobs should run on all hosts not explicitly listed. You do not need to define this parameter if you want to use all batch server hosts and you do not need host preferences.

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 (see 'Host Partitions' on page 206).

Default: all batch hosts.

Scheduling Policy

LSF Batch allows many policies to be defined at the queue level. These affect the order jobs in the queue are scheduled.

Queue Level Fairshare

The concept of queue level fairshare was discussed in 'Scheduling Policies' on page 31. The configuration syntax for this policy is:

```
FAIRSHARE = USER_SHARES[ [username, share] [username, share] . . . . . ]
```

Note

These are real square brackets, not syntactic notation.

7 LSF Batch Configuration Reference

`username` is a user login name, a user group name, the reserved word `default`, or the reserved word `others`. `share` is a positive integer specifying the number of shares of resources that a user or user group has in the cluster.

The `USER_SHARES` assignment for a queue is interpreted in the same way as the `USER_SHARES` assignment in a host partition definition in the `lsb.hosts` file. See *'Host Partitions' on page 206* for explanation of `USER_SHARES`. In general, a job has a higher scheduling priority if the job's owner has more shares, fewer running jobs, has used less CPU time and has waited longer in the queue.

Note the differences between the following two definitions:

```
FAIRSHARE = USER_SHARES[[grp1, share1] [grp2, share2]]
FAIRSHARE = USER_SHARES[[grp1, share1] [grp2@, share2]]
```

The '@' immediately after a user group name means that the shares apply to each individual user in the user group. Without '@', the shares apply to the user group as a whole. If you want to further subdivide the shares allocated to a user group, define a share tree for that group in the `lsb.users` file. This implements hierarchical fairshare as discussed in *'Configuring Hierarchical Fairshare' on page 117*.

See *'Controlling Fairshare' on page 113* for examples of fairshare configuration.

Note

Queue level fair share scheduling is an alternative to host partition fair share scheduling. You cannot use both in the same LSF cluster for the same host(s).

Preemptive Scheduling

The concept of preemptive scheduling was discussed in *'Preemptive Scheduling' on page 32*. `PREEMPTION` takes two possible parameters, `PREEMPTIVE` and `PREEMPTABLE`. The configuration syntax is:

```
PREEMPTION = PREEMPTIVE[q1 q2 ...] PREEMPTABLE
```

where `[q1, q2 ...]` are an optional list of queue names of lower priorities.

Note

These are real square brackets, not syntactic notation.

If `PREEMPTIVE` is defined, this defines a preemptive queue that will preempt jobs in [`q1`, `q2`, ...]. Jobs in a preemptive queue can preempt jobs from the specified lower priority queues running on a host by suspending some of them and starting the higher priority jobs on the host.

If `PREEMPTIVE` is specified without a list of queue names, then this queue preempts all lower priority queues.

If the `PREEMPTIVE` policy is not specified, jobs dispatched from this queue will not suspend jobs from lower priority queues.

A queue can be specified as being preemptable by defining `PREEMPTABLE` in the `PREEMPTION` parameter of the queue.

Jobs from a preemptable queue can be preempted by jobs in any higher priority queues even if the higher priority queues do not have `PREEMPTIVE` defined. A preemptable queue is complementary to the preemptive queue. You can define a queue that is both preemptive as well as preemptable by defining both `PREEMPTIVE` and `PREEMPTABLE`. Thus the queue will preempt lower priority queues while it can also be preempted by higher priority queues.

Exclusive Queue

An exclusive queue is created by specifying `EXCLUSIVE` in the policies of a queue.

If the `EXCLUSIVE` policy is specified, this queue performs exclusive scheduling. A job only runs exclusively if it is submitted to a queue with exclusive scheduling, and the job is submitted with the `bsub -x` option. An exclusive job runs by itself on a host—it is dispatched only to a host with no other batch jobs running.

Once an exclusive job is started on a host, the LSF Batch system locks that host out of load sharing by sending a request to the underlying LSF so that the host is no longer available for load sharing by any other task (either interactive or batch) until the exclusive job finishes.

Because exclusive jobs are not dispatched until a host has no other batch jobs running, it is possible for an exclusive job to wait indefinitely if no batch server host is ever completely idle. This can be avoided by configuring some hosts to run only one batch job at a time; that way the host is certain to have no batch jobs running when the previous batch job completes, so the exclusive job can be dispatched there.

7 LSF Batch Configuration Reference

The exclusive scheduling policy is specified using the following syntax:

```
EXCLUSIVE = {Y | N}
```

Migration

The `MIG` parameter controls automatic migration of suspended jobs.

`MIG = number`

If `MIG` is specified, then *number* is the migration threshold in minutes. If a checkpointable or rerunable job is suspended for more than `MIG` minutes and no other job on the same host is being migrated, LSF Batch checkpoints (if possible) and kills the job. Then LSF Batch restarts or reruns the job on another suitable host if one is available. If LSF is unable to rerun or restart the job immediately, the job reverts to `PEND` status and is queued with a higher priority than any submitted job, so it is rerun or restarted before other queued jobs are dispatched.

The `lsb.hosts` file can also specify a migration threshold. Jobs are migrated if either the host or the queue specifies a migration threshold. If `MIG` is defined both here and in `lsb.hosts`, the lower threshold is used.

Jobs that are neither checkpointable nor rerunable are not migrated.

Default: no migration.

Queue-Level Pre-/Post-Execution Commands

Pre- and post-execution commands can be configured on a per-queue basis. These commands are run on the execution host before and after a job from this queue is run, respectively. By configuring appropriate pre- and/or post-execution commands, various situations can be handled such as:

- Creating and deleting scratch directories for the job
- Assigning jobs to run on specific processors on SMP machines
- Customized scheduling
- License availability checking.

Note that the job-level pre-exec specified with the `-E` option of `bsub` is also supported. In some situations (for example, license checking), it is possible to specify a queue-level pre-execution command instead of requiring every job be submitted with the `-E` option.

The execution commands are specified using the `PRE_EXEC` and `POST_EXEC` keywords; for example:

```
Begin Queue
QUEUE_NAME      = priority
PRIORITY        = 43
NICE             = 10
PRE_EXEC        = /usr/people/lsf/pri_prexec
POST_EXEC       = /usr/people/lsf/pri_postexec
End Queue
```

Pre-/Post-Execution Command Setup

The following points should be considered when setting up the pre- and post-execution commands at the queue level.

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. For example, the following is valid:

```
PRE_EXEC = /usr/local/lsf/misc/testq_pre >> /tmp/pre.out
POST_EXEC = /usr/local/lsf/misc/testq_post | grep -v "Hey!"
```

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'`.

NT

The pre- and post-execution commands are run under `cmd.exe/c`. Both the pre- and post-execution commands are run as the user by default. If you must run these commands as a different user, such as `root` (to do privileged operations, if

7 LSF Batch Configuration Reference

necessary), you can configure the parameter `LSB_PRE_POST_EXEC_USER` in the `lsf.sudoers` file. See *'The lsf.sudoers File' on page 189* for details.

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.

- If the pre-execution command exits with a non-zero exit code, then 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.
- When a job is dispatched from a queue which has a pre-execution command, LSF Batch 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, LSF Batch will still run the original post-execution command for this job.
- 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` (see *'Automatic Job Requeue' on page 231*). 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.
- 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.

Default: no pre- and post-execution commands.

Job Starter

A job starter can be defined for an individual queue to create a specific environment for submitted jobs prior to execution. The configuration syntax for the job starter parameter is:

```
JOB_STARTER = starter [starter] [%USRCMD] [starter]
```

where the string *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. A special string %USRCMD can be used to represent the position of the user's job in the job starter command line.

When a batch job is submitted to a queue, LSF Batch holds it in a *job file* until conditions are right for it to be executed.

UNIX The *job file* is just a Bourne shell script run by the batch daemon at execution time.

NT The *job file* is just a batch file processed by the batch daemon at execution time.

Default: no job starter.

If a job starter is specified, the user commands run after the job starter, so the %USRCMD string is not usually required. For example, these two job starters will both give the same results:

```
JOB_STARTER = csh -c
```

```
JOB_STARTER = csh -c %USRCMD
```

However, the %USRCMD string may be enclosed with quotes or followed by additional commands. For 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"
```

Configurable Job Control Actions

Job control in LSF Batch refers to some well-known control actions that will cause a job's status to change. These actions include:

SUSPEND

Change a running job to `SSUSP` or `USUSP`. The default action is to send signal `SIGTSTP` for parallel or interactive jobs and `SIGSTOP` for other jobs.

RESUME

Change a suspended job to `RUN` status. The default action is to send signal `SIGCONT`.

TERMINATE

Terminate a job and possibly cause the job change to `EXIT` status. The default action is to send `SIGINT` first, then send `SIGTERM` 10 seconds after `SIGINT`, then send `SIGKILL` 10 seconds after `SIGTERM`. The 10 second interval can be overridden by the `JOB_TERMINATE_INTERVAL` in the `lsb.params` file (see '`JOB_TERMINATE_INTERVAL`' on page 196 for details).

Note

On Windows NT, 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.

Several situations may require overriding or augmenting the default actions for job control. For example:

- A distributed parallel application requires that it receive a catchable signal when the job is suspended, resumed or terminated to propagate the signal to remote processes.
- Notification to the user when his/her job is suspended.
- An application holds resources (for example, licenses) that are not freed by suspending the job. The administrator can set up an action to be performed that causes the license to be released before the job is suspended and re-acquired when the job is resumed.
- The administrator wants the job checkpointed before being suspended when a run window closes.

It is possible to override the actions used for job control by specifying the `JOB_CONTROLS` parameter in the `lsb.queues` file. The format is:

```
Begin Queue
.
JOB_CONTROLS = SUSPEND[signal | CHPNT | command] \
               RESUME[signal | command] \
               TERMINATE[signal | CHPNT | command]
.
End Queue
```

Here, `signal` is a UNIX signal name (such as, `SIGSTOP`, `SIGTSTP`). `CHKPNT` is a special action, which causes the system to checkpoint the job. Alternatively, `command` specifies an `/bin/sh` command line to be invoked.

When LSF Batch needs to suspend or resume a job it will invoke the corresponding action as specified by the `SUSPEND` or `RESUME` parameters, respectively.

If the action is a signal, then the signal is sent to the job. If the action is a command, then the following points should be considered:

- 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.
- 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_JOBPIIDS` - a list of current process IDs of the job.

For the `SUSPEND` action command, the following environment variable is also set:

`LSB_SUSP_REASONS` - an integer representing a bitmap of suspending reasons as defined in `lsbatch.h`.

7 LSF Batch Configuration Reference

The suspending reason can allow the command to take different actions based on the reason for suspending the job.

The `SUSPEND` action causes the job state to be changed from `RUN` state to the `USUSP` (in response to `bstop`) state or the `SSUSP` (otherwise) state when the action is completed. The `RESUME` action causes the job to go from `SSUSP` or `USUSP` state to the `RUN` state when the action is completed.

If the `SUSPEND` action is `CHKPNT`, then the job is checkpointed and then stopped by sending the `SIGSTOP` signal to the job automatically.

LSF Batch invokes the `SUSPEND` action to bring a job into `SSUSP` or `USUSP` status in the following situations:

- when the user or LSF administrator issued an `bstop` command on the job
- when load condition on the execution host satisfies the suspend condition
- when the queue's run window closes
- when the job is being preempted by a higher priority job

However, in certain situations you may want to terminate the job instead of calling the `SUSPEND` action. For example, you may want to kill jobs if the run window of the queue is closed. This can be achieved by configuring the queue to invoke the `TERMINATE` action instead of `SUSPEND` by specifying the following parameter:

```
TERMINATE_WHEN = WINDOW | LOAD | PREEMPT
```

If the `TERMINATE` action is `CHKPNT`, then the job is checkpointed and killed atomically.

If the execution of an action is in progress, no further actions will be initiated unless it is the `TERMINATE` action. A `TERMINATE` action is issued regardless of the current state of the job.

The following defines 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

```

Note that the command line inside an action definition must not be quoted.

LSF Batch invokes the `TERMINATE` action when a `SUSPEND` action that is redirected to `TERMINATE` with the `TERMINATE_WHEN` parameter should be invoked, or when the job reaches its `RUNLIMIT`, or `PROCESSLIMIT`.

Since the `stdout` and `stderr` of the job control action command are redirected to `/dev/null`, there is no direct way of knowing whether the command runs correctly. You should make sure the command line is correct. If you want to see the output from the command line for testing purposes, redirect the output to a file inside the command line.

Automatic Job Requeue

A queue can be configured to automatically requeue a job if the job exits with particular exit value(s). The parameter `REQUEUE_EXIT_VALUES` is used to specify a list of exit codes that can cause an exited job to be requeued; for example,

```

Begin Queue
PRIORITY           = 43
REQUEUE_EXIT_VALUES = 99 100
End Queue

```

This configuration enables jobs that exit with 99 or 100 to be requeued to the head of the queue from which it was dispatched. When a job is requeued, the output from the failed run is not saved and no mail is sent. The user will only receive notification when the job exits with a value different from the values listed in the `REQUEUE_EXIT_VALUES` parameter. Additionally, a job terminated by a signal is not requeued.

Default: Jobs in a queue are not requeued.

Exclusive Job Requeue

The queue parameter `REQUEUE_EXIT_VALUE` controls job requeue behaviour. It defines a series of exit code values. If batch job exit with one of those values, the job gets requeued. There is a special requeue method called exclusive requeue. If the exit value is defined as `EXCLUDE (value)`, the job will be requeued when it exits with the given value, but it will not be dispatched to the same host where it exited with the value. For example:

```
Begin Queue
.  
REQUEUE_EXIT_VALUES=30 EXCLUDE(20)  
HOSTS=hostA hostB hostC  
.  
End Queue
```

The job in this queue can be dispatched to `hostA`, `hostB` or `hostC`. If the job exits with value 30, it will be dispatched on any of `hostA`, `hostB` or `hostC`. If the job exits with value 20 on `hostA`, when requeued, it will only be dispatched to `hostB` or `hostC`. Similarly, if the job again exits with a value of 20, it will only be dispatched on `hostC`. Finally, if the job exits with value 20 on `hostC`, the job will be pending forever.

Note

If `lsmatchd` is restarted, it will not remember the previous hosts where 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 exclusive exit code.

Default Host Specification for CPU Speed Scaling

LSF runs jobs on heterogeneous machines. To set the CPU time limit for jobs in a platform-independent way, LSF scales the CPU time limit by the CPU factor of the hosts involved.

The `DEFAULT_HOST_SPEC` defines a default host or host model that will be used to normalize the CPU time limit of all jobs, providing consistent behaviour for users.

```
DEFAULT_HOST_SPEC = host_spec  
host_spec must be a host name defined in the lsf.cluster.cluster file, or  
a host model defined in the lsf.shared file.
```

The CPU time limit defined in this file, or by the user through the `bsub -c cpu_limit` option, is interpreted as the maximum number of minutes of CPU time that a job may run on a host of the default specification. When a job is dispatched to a host for execution, the CPU time limit is then normalized according to the execution host's CPU factor.

If `DEFAULT_HOST_SPEC` is defined in both the `lsb.params` file and the `lsb.queues` file for an individual queue, the value specified for the queue overrides the global value. If a user explicitly gives a host specification when submitting a job, the user specified host or host model overrides the values defined in both the `lsb.params` and the `lsb.queues` files.

Default: `DEFAULT_HOST_SPEC` in the `lsb.params` file.

NQS Forward Queues

To interoperate with NQS, you must configure one or more LSF Batch queues to forward jobs to remote NQS hosts. An NQS forward queue is an LSF Batch queue with the parameter `NQS_QUEUES` defined.

`NQS_QUEUES = queue_name@host_name ...`

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)`.

queue_name is the name of an NQS 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, then it is routed to that queue. If no queue accepts the job, it remains pending in the NQS forward queue.

The `lsb.nqsmaps` file (see *'The lsb.nqsmaps File' on page 235*) must be present in order for LSF Batch to route jobs in this queue to NQS systems.

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.

`DESCRIPTION = text`

A brief description of the job queue. This information is displayed by the `bqueues -l` command. The description can include any characters, including

7 LSF Batch Configuration Reference

white space. The description can be extended to multiple lines by ending the preceding line with a back slash '\'. The maximum length for the description is 512 characters.

This description should clearly describe the service features of this queue to help users select the proper queue for each job.

Queue Level Checkpoint and Rerun

A queue can be configured to automatically checkpoint and rerun jobs. Jobs submitted to this queue do not need to specify checkpoint and rerun options, and wrapper scripts do not need to be written for job submission executables. The parameters `CHKPNT` and `RERUNNABLE` are used to configure the queue-level checkpoint and rerun options.

Note

User commands always take precedence over the configured queue options. Jobs submitted and modified using the `-k`, `-x`, and `-kn` options override the queues configured options.

Syntax

```
Begin Queue
QUEUE_NAME = queueName
...
CHKPNT = chkpntDir chkpntPeriod
RERUNNABLE = yes|no # case insensitive
...
End Queue
```

CHKPNT

Automatic checkpointing is enabled when a checkpoint period (`chkpntPeriod`) and checkpoint (`chkpntDir`) directory is specified after the `CHKPNT` parameter.

chkpntDir

The `chkpntDir` specifies the directory where the checkpoint files are created. `chkpntDir` must be string specifying a valid directory.

chkpntPeriod

The `chkpntPeriod` specifies a time interval in minutes for automatic checkpointing. Must be a positive integer.

RERUNNABLE

Automatic job rerun (restart) is enabled when the parameter `RERUNNABLE` is set to yes. Rerun is disabled when `RERUNNABLE` is set to no.

Default: Queues are not configured to checkpoint and rerun jobs.

Example

The following example defines a queue named *myQueue* that automatically saves job checkpoint information in the directory `~/work/chkpnt/` every 10 minutes. The queue is defined to automatically rerun (restart) jobs.

```
Begin Queue
QUEUE_NAME = myQueue
...
CHKPNT = ~/work/chkpnt/ 10 # period is 10 minutes
RERUNNABLE = Y # specify a rerunnable queue
...
End Queue
```

The `lsb.nqsmaps` File

The `lsb.nqsmaps` file contains information on configuring LSF for interoperation with NQS. This file is optional.

Hosts

NQS uses a machine identification number (MID) to identify each host in the network that communicates using the NQS protocol. This MID must be unique and must be the same in the NQS database of each host in the network. The MID is assigned and put into the NQS data base using the NQS program `nmapmgr(1m)` or Cray NQS command `qmgr(8)`. `mbatchd` uses the NQS protocol to talk with NQS daemons for routing,

7 LSF Batch Configuration Reference

monitoring, signalling, and deleting LSF Batch jobs that run on NQS hosts. Therefore, the MIDs of the LSF master host, and any LSF host that might become the master host when the current master host is down, must be assigned and put into the NQS database of each host which may possibly process LSF Batch jobs.

In the mandatory `Hosts` section, list the MIDs of the LSF master host (and potential master hosts) and the NQS hosts that are specified in the `lsb.queues` file. If an NQS destination queue specified in the `lsb.queues` file is a pipe queue, the MIDs of all the destination hosts of this pipe queue must be listed here. If a destination queue of this pipe queue is itself a pipe queue, the MIDs of the destination hosts of this queue must also be listed, and so forth.

There are three mandatory keywords in this section:

HOST_NAME

The name of an LSF or NQS host. It can be the official host name or an alias host name known to the master batch daemon (`mbatchd`) through `gethostbyname(3)`.

MID

The machine identification number of an LSF or NQS host. It is assigned by the NQS administrator to each host communicating using the NQS protocol.

OS_TYPE

The operating system (OS) type of the NQS host. At present, its value can be one of `ULTRIX`, `HPUX`, `AIX`, `SOLARIS`, `SUNOS`, `IRIX`, `OSF1`, `CONVEX` or `UNICOS`. It is used by `mbatchd` to deliver the correct signals to the LSF Batch jobs running on this NQS host. An incorrect OS type would cause unpredictable results. If the host is an LSF host, the type is specified by the `type` field of the `Host` section in the `lsf.cluster.cluster` file. `OS_TYPE` is ignored; '-' must be used as a placeholder.

Begin Hosts

HOST_NAME	MID	OS_TYPE	
cray001	1	UNICOS	#NQS host, must specify OS_TYPE
sun0101	2	SOLARIS	#NQS host
sgi006	3	IRIX	#NQS host
hostA	4	-	#LSF host; OS_TYPE is ignored
hostD	5	-	#LSF host

```

hostC      6      -      #LSF host
End Hosts

```

Users

LSF assumes shared and uniform user accounts on all of the LSF hosts. However, if the user accounts on NQS hosts are not the same as on LSF hosts, account mapping is needed so that the network server on the remote NQS host can take on the proper identity attributes. The mapping is performed for all NQS network conversations. In addition, the user name and the remote host name may need to match an entry either in the `.rhosts` file in the user's home directory, or in the `/etc/hosts.equiv` file, or in the `/etc/hosts.nqs` file on the server host. For Cray NQS, the entry may be either in the `.rhosts` file or in the `.nqshosts` file in the user's home directory.

This optional section defines the user name mapping from the LSF master host to each of the NQS hosts listed in the `Host` section above, that is, the hosts on which the jobs routed by LSF Batch may run. There are two mandatory keywords:

`FROM_NAME`

The name of an LSF Batch user. It is a valid login name on the LSF master host.

`TO_NAME`

A list of user names on NQS hosts to which the corresponding `FROM_NAME` is mapped. Each of the user names is specified in the form `username@hostname`. The `hostname` is the official name or an alias name of an NQS host, while the `username` is a valid login name on this NQS host. The `TO_NAME` of a user on a specific NQS host should always be the same when the user's name is mapped from different hosts. If no `TO_NAME` is specified for an NQS host, LSF Batch assumes that the user has the same user name on this NQS host as on an LSF host.

Begin Users

```

FROM_NAME  TO_NAME
user3      (user3l@cray001 luser3@sgi006)
user1      (suser1@cray001) # assumed to be user1@sgi006
End Users

```

If a user is not specified in the `lsb.nqsmaps` file, jobs are sent to NQS hosts with the same name the user has in LSF.

7 LSF Batch Configuration Reference

A. Troubleshooting and Error Messages

This chapter describes some common problems with LSF and LSF Batch operations, answers some frequently asked questions, and provides some instructions for solving problems.

Error Log Messages

When something goes wrong, the daemons almost always log an error message. The first step is to find the appropriate log and see whether there are any messages.

Specific error log messages are listed in ‘*Error Messages*’ on page 245.

Finding the Error Logs

Error messages of LSF servers are logged in either the `syslog(3)` or specified files. This is determined by the `LSF_LOGDIR` definition in the `lsf.conf` file. For complete instructions on finding the LSF server logs, see ‘*Managing Error Logs*’ on page 45.

UNIX

If you configure LSF to log daemon messages using `syslog`, the destination file is determined by the `syslog` configuration. On most systems, you can find out which file the LSF messages are logged in with the command:

```
grep daemon /etc/syslog.conf
```

Once you have found the `syslog` file, you can select the LSF error messages with the command:

A Troubleshooting and Error Messages

```
egrep 'lim|res|batchd' syslog_file
```

Look at the `/etc/syslog.conf` file and the manual page for `syslog` or `syslogd` for help in finding the system logs.

When searching for log messages from LSF servers, you are more likely to find them on the remote machine where LSF put the task than on your local machine where the command was given.

LIM problems are usually logged on the master host. Run `lsid` to find the master host, and check `syslog` or the `lim.log.hostname` file on the master host. The `res.log.hostname` file contains messages about RES problems, execution problems and setup problems for LSF. Most problems with interactive applications are logged in the remote machine's log files.

Errors from LSF Batch are logged either in the `mbatchd.log.hostname` file on the master host, or the `sbatchd.log.hostname` file on the execution host. The `bjobs` or `bhist` command tells you the execution host for a specific job.

Most LSF log messages include the name of an internal LSF function to help the developers locate problems. Many error messages can be generated in more than one place, so it is important to report the entire error message when you ask for technical support.

Shared File Access

UNIX

A frequent problem with LSF 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.

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.

NT

To share files among NT 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 NT

For file sharing across UNIX and NT, you require a third party NFS product on NT to export directories from NT to UNIX.

Common LSF Base Problems

This section lists some other 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:

A Troubleshooting and Error Messages

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 on the local host.

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` file.

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)`.

NT

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` 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.

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.

Common LSF Batch Problems

This section lists some common problems with LSF Batch. Most problems are due to incorrect installation or configuration. Check the `mbatchd` and `sbatchd` error log files; often the log messages points directly to the problem.

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 Batch 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 Batch'* on page 244.

`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 LSF and LSF Batch are using 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 *'Registering LSF Service Ports'* on page 84 of the *LSF Installation Guide*.

Host Not Used By LSF Batch

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 in the `HostGroup` or `HostPartition` sections of the `lsb.hosts` file, or 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 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
```

LSF daemon message logs are described in *'Managing Error Logs'* on page 45.

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`.

A Troubleshooting and Error Messages

This error is usually caused by incorrect file permissions or missing files. All directories in the path to the configuration files must have 'x' permission for the LSF administrator, and the actual files must have '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 `ident` daemon on *host*. Check that `ident` is defined in *host's* `inetd.conf` and the `ident` 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 *host* is configured correctly.

userok: Request from bad port (<portno>), 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 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. See LSF Guide for help

initSock: Service `lim/udp` is unknown. Read LSF Guide for help

get_ports: <serv> service not registered

The LSF services are not registered. See 'Registering LSF Service Ports' on page 84 of the *LSF Installation Guide*.

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.

A Troubleshooting and Error Messages

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 25 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 FLEXlm is set up correctly, or contact your LSF technical support.

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.

A Troubleshooting and Error Messages

```
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 an 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 *host* is a multi-addressed host. See “*Host Naming*” in the *LSF Installation Guide*.

```
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 an 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 *'Changing LIM Configuration'* on page 55 for a description of the protocol.

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 userID and username on all LSF hosts.

`doacceptconn`: root remote execution permission denied

A Troubleshooting and Error Messages

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 manager, 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 Batch 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 *'Installation on AFS'* on page 97 of the *LSF Installation Guide*).

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 `managerid` has read, write, and execute permissions on the `logdir` directory.

If `logdir` is on AFS, check that the instructions in *'Installation on AFS'* on page 97 of the *LSF Installation Guide* have been followed. Do `fs ls` 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.

A Troubleshooting and Error Messages

`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 Batch.

`do_restartReq: Failed to get hData of host <hostname>/<hostaddr>
mbatchd received a request from sbatchd on host hostname, 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
hostname 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 hostname.`

`% badmin reconfig`

B. LSF Directories

This table lists the directories used by the LSF system, their modes and contents.

Table 2. LSF Directories

Directory	Mode	Contents
<code>\$LSB_CONFDIR</code> <code>\$LSB_CONFDIR/*</code>	755	LSF Batch configuration files, must be owned by the primary LSF administrator, and shared by all potential master hosts
<code>\$LSB_SHAREDIR/</code> <code>cluster/logdir</code>	755	LSF Batch accounting files, must be owned by the primary LSF administrator, and shared by all potential master hosts
<code>\$LSB_LOCALDIR</code>	755	Local directory, on the master host, owned by the primary LSF administrator, used to store the primary copy of LSF Batch event logs
<code>\$LSF_BINDIR</code>	755	User commands, must allow setuid to root, shared by all hosts of the same type
<code>\$LSF_CONFDIR</code>	755	LSF cluster configuration files, must be owned by the primary LSF administrator, and shared by all LSF server hosts
<code>\$LSF_ENVDIR</code>	755	<code>lsf.conf</code> file, must be owned by root
<code>\$LSF_INCLUDEDIR</code>	755	Header files <code>lsf/lsf.h</code> and <code>lsf/lsbatch.h</code>
<code>\$LSF_INDEP</code>	755	Host type independent files shared by all hosts
<code>\$LSF_LIBDIR</code>	755	LSF libraries, shared by all hosts of the same type
<code>\$LSF_LOGDIR</code>	1777	Server error logs, must be owned by root
<code>\$LSF_MACHDEP</code>	755	Host type dependent files shared by all hosts of the same type
<code>\$LSF_MANDIR</code>	755	LSF man pages shared by all hosts

B LSF Directories

Table 2. LSF Directories

Directory	Mode	Contents
<code>\$LSF_MISC</code>	755	Examples and other miscellaneous files shared by all hosts
<code>\$LSF_SERVERDIR</code>	755	Server binaries, must be owned by root, and shared by all hosts of the same type
<code>\$XLSF_APPDIR</code>	755	Window application resource files, shared by all hosts
<code>\$XLSF_UIIDIR</code>	755	GUI UID files, shared by all hosts of the same type

C. Sample System Support

This section describes how LSF can be configured to support specific systems. Several systems are discussed: IRIX 6 processor sets, Solaris 2.6 processor sets, IBM SP-2, Cray Research NQS, and Atria Clearcase.

IRIX 6 Processor Sets

IRIX 6 allows the processors in a multiprocessor system to be divided into groups of processors call *processor sets*. IRIX 6 provides facilities to allow a user to define processor sets, and to run processes onto specific processor sets.

The `pset(1M)` command allows administrators to set up and manipulate processor sets and associate processes with sets. Once a process is associated with a processor set, the process and all its children will be scheduled only on the processors in that set. The definition of the processor set can be dynamically changed to increase or reduce the number of processors a process can be scheduled on.

LSF can be made to interface with processor sets by using `pset(1M)` in the queue-level pre- and post-execution commands (see '*Queue-Level Pre-/Post-Execution Commands*' on page 224). This allows batch jobs to be assigned to certain processors.

Note

*Since the `pset` command must be run as root but, by default, queue-level pre- and post- execution commands are run as the user that submitted the command, you must define `LSB_PRE_POST_EXEC_USER=root` in `/etc/lsf.sudoers`. See '*The `lsf.sudoers` File*' on page 189 for details.*

The following gives examples on how to handle different processor allocation situations for an 8-processor machine.

Time-Based Processor Allocation

During the day you want to ensure that batch jobs only use four processors with the remaining four dedicated for interactive users. During the night you want the batch jobs to be able to use all processors.

This can be accomplished by:

Step 1 Create a 'batch' processor set and an 'interactive' processor set with half the processors in each set. This can be done by setting up an `/etc/pssettab` file as follows:

```
# <symbolic name>    <pset id>    <processor list>
#
batch                100        0,1,2,3
interactive 101        4,5,6,7
```

Run the `pset` command as `root` to tell the operating system to read the `pssettab` file:

```
# pset -i -v
```

Step 2 All batch queues should specify the following `PRE_EXEC` parameter:

```
PRE_EXEC = pset -p $LS_JOBPID batch
```

The `LS_JOBPID` will be set to the process identifier of the batch job process. The `pset` command will tell IRIX to schedule that process and any of its children in the batch processor set.

Step 3 Set up two cron jobs one of which runs every morning at 8 a.m. and the other every evening at 6 p.m. The cron job running at 8 a.m. should execute the command:

```
# Move processors 4,5,6,7 out of the batch processor set
pset -s batch \!4,5,6,7
```

The cron job running at 6 p.m. should execute the command:

```
# Move processors 4,5,6,7 into the batch processor set.
pset -s batch +4,5,6,7
```

User-Based Processor Allocation

You want to give a particular user (Jane) exclusive access to one processor if she has jobs to run. Otherwise, users should be able to use all eight processors for batch jobs.

This can be accomplished by:

Step 1 Create a processor set called 'excl' and another processor set 'batch' for normal batch jobs. The 'batch' processor set can initially contain all processors and the 'excl' set contains only processor 0.

Step 2 Create a queue that will only accept jobs from Jane. The pre- and post-execution commands can be used to shuffle processors between processor sets such that Jane will get exclusive access to a processor:

```
Begin Queue
QUEUE_NAME    = exclusive
PRIORITY      = 43
USERS         = jane
# Move processor 0 from batch processor set to excl processor set
# Associate the job with the excl set.
PRE_EXEC      = pset -s excl +0; pset -s batch -0; pset -p $LS_JOBPID excl
#Move processor 0 back to the batch processor set
POST_EXEC     = pset -s excl -0; pset -s batch +0
DESCRIPTION   = Provides exclusive access to a processor for Jane's jobs
End Queue
```

Other queues should have their pre-execution command set to:

```
PRE_EXEC      = pset -p $LS_JOBPID batch
```

Other Situations

More complicated situations can be handled by using scripts in the pre- and post-execution commands which check for other conditions. For example in the above '*User-Based Processor Allocation*' case, if you wanted to give Jane up to four processors to use (but not more), the pre-execution script could use `pset` to determine how many processors were already in the 'excl' set and move an additional processor from the 'batch' set into the 'excl' set until the 'excl' set has four processors.

Support for Solaris Processor Sets

Solaris 2.6 allows the processors in a multiprocessor system to be divided into groups of processors called processor sets. It provides facilities to allow a user to define processor sets, and to run processes on specific processor sets.

The `psrset(1M)` command allows administrators to set up, manipulate, and associate processes with processor sets. Once a process is associated with a processor set, the process and all its children will be scheduled on the processors in that set. The definition of the processor can be dynamically changed to increase or reduce the number of processors on which a process can be scheduled.

LSF can be made to interface with processor sets by using `psrset(1M)` in the queue-level pre- and post-execution commands (see ‘Queue-Level Pre-/Post-Execution Commands’ on page 224 for detailed information). This allows batch jobs to be dispatched onto certain processors.

Note

Since the `psrset(1M)` command must be run as root but, by default, queue-level pre- and post-execution commands are run as the user that submitted the command, you must define `LSB_PRE_POST_EXEC_USER=root` in `/etc/lsf.sudoers`. See ‘The `lsf.sudoers` File’ on page 189 for more details.

The following gives examples of how to handle different processor allocation situations for an 8-processor machine.

Time-Based Processor Allocation

Suppose that during the day, you want to ensure that batch jobs only use 4 processors with the remaining 4 dedicated for interactive users. During the night, you want the batch jobs to be able to use all processors. This can be accomplished by using the following procedure:

Step 1 Create a processor set with 4 processors (0,1,2,3) for batch jobs—use the following commands (assuming that no processor set has yet been defined, the processor set id will start from 1).

```
%psrset -c 0 # create a processor set. check the  
processor set id from the print out.
```

```
processor 0: was not assigned, now 1
%psrset -a 1 1 # put processor 1 into processor set 1.
processor 1: was not assigned, now 1
%psrset -a 1 2 # put processor 2 into processor set 1.
processor 2: was not assigned, now 1
%psrset -a 1 3 # put processor 3 into processor set 1.
processor 3: was not assigned, now 1
```

Step 2 Create another processor set with 4 processor (4,5,6,7) for interactive jobs—use the following commands.

```
%psrset -c 4 # create a processor set. check the
processor set id from the print out.
processor 4: was not assigned, now 2
%psrset -a 2 5 # put processor 5 into processor set 2.
processor 5: was not assigned, now 2
%psrset -a 2 6 # put processor 6 into processor set 2.
processor 6: was not assigned, now 2
%psrset -a 2 7 # put processor 7 into processor set 2.
processor 7: was not assigned, now 2
```

Step 3 All batch queues should specify the following PRE_EXEC parameter:

```
PRE_EXEC = psrset -b 1 $LS_JOBPID
```

The LS_JOBPID will be set to the process identifier of the batch job process. The psrset command will tell the system to schedule that process and any of its children in the batch processor set.

Step 4 Interactive queues should specify the following PRE_EXEC parameter:

```
PRE_EXEC = psrset -b 2 $LS_JOBPID
```

The LS_JOBPID will be set to the process identifier of the interactive job process. The psrset command will tell the system to schedule that process and any of its children in the interactive processor set.

Step 5 Set up two cron jobs, one of which runs every morning at 8 a.m., and the other every evening at 6 p.m. The cron job running at 8 a.m. should execute the following commands:

```
# Move processors 4,5,6,7 out of the batch processor set.
psrset -r 4
psrset -r 5
psrset -r 6
psrset -r 7
```

The cron job running at 6 p.m. should execute the commands:

```
# Move processors 4,5,6,7 into the batch processor set.
psrset -a 1 4
psrset -a 1 5
psrset -a 1 6
psrset -a 1 7
```

Alternatively, you can use LSF JobScheduler to run this periodic job.

User-Based Processor Allocation

Suppose you want to give a particular user (Bob) exclusive access to one processor if he has jobs to run. Otherwise, users should be able to use all 8 processors for batch jobs. This can be accomplished by:

Step 1 Create a processor set 3 and leave it empty using the following commands:

```
%psrset -c 0 # create a processor set. check the
processor set id from the print out.
processor 0: was 1, now 3
%psrset -a 1 0 # release processor 0 from processor set
3, and put it back to the processor set 1.
processor 0: was 3, now 1
```

Step 2 Create a queue that will only accept jobs from Bob. The pre- and post-execution commands can be used to shuffle processors between processor sets such that Bob will get exclusive access to a processor.

```
Begin Queue
QUEUE_NAME      = exclusive
PRIORITY        = 43
USERS           = bob
# Move processor 0 from batch processor set 1 to
```

```
exclusive processor set 3
# associate the job with the exclusive set
PRE_EXEC           = /usr/local/lsf/etc/procset
$LSB_JOBPID
# Move processor 0 back to the batch processor set
POST_EXEC          = /usr/local/lsf/etc/procset
DESCRIPTION        = Provide an exclusive access to a
processor for Bob's jobs
End Queue
```

The sample script `procset` is included in `examples/solaris-pset` directory of the distribution.

Other Situations

More complicated situations can be handled by using scripts in the pre- and post-execution commands which check for other condition. For example in the above 'User-Based Processor Allocation' case, if you wanted to give Bob up to 4 processors to use (but not more), the pre-execution script could use `psrset (1M)` to determine how many processors were already in processor set 3 and move an additional processor from the batch processor set 1 into processor set 3 until processor set 3 has 4 processors. Remember to release the processors to the batch processor set 1.

IBM SP-2 Support

An IBM SP-2 system consists of multiple nodes running AIX. The system can be configured with a high-performance switch to allow high bandwidth, low latency communication between the nodes. The allocation of the switch to jobs as well as the division of nodes into pools is controlled by the SP-2 Resource Manager.

IBM's Parallel Operating Environment (POE) interfaces with the Resource Manager to allow users to run parallel jobs requiring dedicated access to the high performance switch.

The following are provided in LSF to support POE jobs running under LSF:

- A `poejob` script which translates the nodes allocated by LSF Batch into an appropriate host list and environment variables for use by POE. The `poejob` script also causes the job to be requeued, if the request for dedicated access to the switch fails.
- An external LIM (ELIM) which contacts the SP-2 Resource Manager to determine which pool each node is in and the status of the high-performance switch. Two new load indices, `pool` and `lock`, are introduced to represent the pool the node is in and whether the switch is locked or not, respectively. The SP-2 ELIM uses the `jm_status` command to collect information from the Resource Manager.

The 'poejob' script is installed as part of the standard installation procedure. The SP-2-specific ELIM can be found in the examples directory of the distribution.

The following steps should be followed to allow POE jobs to run under LSF:

- Step 1** Ensure that the SP-2 node names are the same as their host names. That is, `jm_status` should return the same names for the nodes that `lsload` returns.
- Step 2** Build and install the SP-2 ELIM. A `README.sp2` file is provided in the examples directory with specific instructions.
- Step 3** Add the following to the `lsf.shared` file in the `LSF_CONFDIR` directory to pick up the indices reported by the ELIM

```
Begin Resource
NAME  TYPE      INTERVAL INCREASING DESCRIPTION
lock  Numeric  60        Y           (IBM SP2 Node lock status)
pool  Numeric  60        N           (IBM SP2 POWER parallel system pool)
End NewIndex Resource
```

- For all queues which accept POE jobs define a requeue exit value and a threshold for the `lock` index.

For example:

```
Begin Queue
NAME=poejobs
lock=0
REQUEUE_EXIT_VALUES = 133
...
End Queue
```

The `poejob` script will exit with 133 if it is necessary to requeue the job. Note that other types of jobs should not be submitted to the same queue. Otherwise, they will get requeued if they happen to exit with 133. The scheduling threshold on the `lock` index prevents dispatching to nodes which are being used in exclusive mode by other jobs.

Note that it is only necessary to enable requeuing of POE jobs if some users submit jobs requiring exclusive access to the node.

Support for HP Exemplar Technical Servers

The HP Exemplar Technical Server is a high performance cache coherent Non Uniform Memory Access (ccNUMA) computer system. The Exemplar system supports the partitioning of the computing resources into subcomplexes which are collections of processors and memory from one or hypernodes in the system. The Subcomplex Manager (SCM) enables administrators to configure processor and memory resources into subcomplexes.

The following are provided in LSF to support the Exemplar:

- An external LIM (ELIM) which collects subcomplex load information. There are 6 load indices collected for each subcomplex including the subcomplex's private memory, global memory, number of CPUs, 5-second run queue, 30-second run queue, and 60-second run queue. This information can be used to control the scheduling of jobs onto a subcomplex.
- A queue-level job starter (see *Job Starter* on page 227) to start a job on a particular subcomplex. Each LSF Batch queue must be associated with one subcomplex.

LSF does not currently support dynamic load balancing between subcomplexes.

The following steps should be taken to setup an Exemplar system to run LSF.

Step 1 Build and install the Exemplar ELIM. Follow the instructions in the `README` file in the `examples` directory of the distribution.

Step 2 Add the definitions for the load indices for each subcomplex in the `lsf.shared` file.

Step 3 Add queue definitions for each subcomplex to the `lsb.queues` file.

Adding Load Indices Definitions

Edit the `LSF_CONFDIR/lsf.shared` configuration. Add the definitions for the load indices for each subcomplex. For example, if you have two subcomplexes, you need to configure 12 indices as follows:

```
Begin Resource
NAME      TYPE      INTERVAL INCREASING DESCRIPTION
sc1Pme    Numeric    60        N          (Subcomplex one private memory)
sc1Gme    Numeric    60        N          (Subcomplex one global memory)
sc1cpu    Numeric    60        N          (Subcomplex one number cpu)
sc1r5s    Numeric    60        Y          (Subcomplex one five sec runQ)
sc1r30    Numeric    60        Y          (Subcomplex one thirty sec runQ)
sc1rlm    Numeric    60        Y          (Subcomplex one one minute runQ)
sc2Pme    Numeric    60        N          (Subcomplex two private memory)
sc2Gme    Numeric    60        N          (Subcomplex two global memory)
sc2cpu    Numeric    60        N          (Subcomplex two number cpu)
sc2r5s    Numeric    60        Y          (Subcomplex two five sec runQ)
sc2r30    Numeric    60        Y          (Subcomplex two thirty sec runQ)
sc2rlm    Numeric    60        Y          (Subcomplex two one minute runQ)
End NewIndex
```

The index names should be of the form `scNxxx` where `N` is the subcomplex number. The name of the subcomplexes defined on the system can be obtained by running the following command

```
% sysinfo -ls
System      load average:      4.30  4.28  4.09
largeGlbMem load average:      3.20  2.18  2.07
```

The subcomplex number corresponds to its position in the list. In the above example, `System` is subcomplex 1 and `largeGlbMem` is subcomplex 2. The names of the indices can be modified if appropriate changes are made to the supplied ELIM.

It is possible to change the name of the indices to include the subcomplex name instead of using a number. This requires changes to the supplied Exemplar ELIM.

The built-in load indices reported by the LIM on the Exemplar are implemented as follows:

- `r15s`, `r1m`, and `r15m` are the total run queue lengths of all nodes in all subcomplexes
- `ut` and `pg` are the CPU utilization and paging rate averaged over all nodes in all subcomplexes
- `mem` is the amount of global memory available
- The `swp` index is not currently available on the Exemplar. The `swp` index should not be used as a threshold to control scheduling

Adding Queue Definitions

Edit the queue definitions in `LSB_CONFDIR/cluster/configdir/lsb.queues` to add queue definitions for each subcomplex. A Job Starter should be specified for each queue to control which subcomplex jobs from the queue will run on.

For example:

```
Begin Queue
QUEUE_NAME      = long
JOB_STARTER     = mpa -sc largeGlbMem
.
.
DESCRIPTION     = Long jobs on the largGlbMem subcomplex
End Queue
Begin Queue
QUEUE_NAME      = short
JOB_STARTER     = mpa -sc System
.
.
DESCRIPTION     = Short jobs on the System subcomplex
End Queue
```

The `JOB_STARTER` parameter uses the `mpa(1)` command to start the job script file onto the subcomplex specified with the `-sc` option. LSF sets the `LSB_JOBFILENAME` environment variable, which specifies a shell script containing the user's commands.

You can use the load indices for each subcomplex to control the scheduling or suspension of jobs on that subcomplex. For example:

```
Begin Queue
QUEUE_NAME      = idle
JOB_STARTER     = mpa -sc System
sclrlm          = 2.0/6.0
.
End Queue
```

would only start jobs on the `System` subcomplex if the 1-minute run queue was below 2.0 and suspend jobs if it goes above 6.0. Note that the load index specified in the scheduling constraints should correspond to the subcomplex specified in the `JOB_STARTER` parameter.

It is possible to make use of the queue level pre-/post-execution commands to move CPUs between subcomplexes on a per-job basis. For example, if an exclusive subcomplex has been set up, CPUs can be moved from the system subcomplex before job execution by the pre-exec command and moved back to the system subcomplex after job execution by the post-execution command.

```

Begin Queue
QUEUE_NAME      = exclusive
JOB_STARTER     = mpa -sc Exclusive
.
PRE_EXEC        = /usr/spp/moveCpuToEx
POST_EXEC       = /usr/spp/moveCpuToSys
End Queue

```

The Exemplar supports kernel level checkpointing using the `chkpnt(1)` and `restart(1)` commands. To enable checkpointing on Exemplar systems, copy `erestart` and `echkpnt` to the `LSF_SERVDIR`.

Users are not required to take any special actions for submitting jobs on a Exemplar system. If an Exemplar system is integrated into a larger cluster of machines, it is possible to set up queues that can dispatch to all machines. You need to specify a Job Starter script, which runs the job file through the `mpa(1)` for the Exemplar, and just executes the job file on non-Exemplar systems. Also scheduling constraints should be specified using the queue-level `RES_REQ` parameter to distinguish between Exemplar and non-Exemplar systems. For example:

```
RES_REQ= (type==Exemplar && sclrlm < 2.0) || (type != Exemplar && rlm < 2.0)
```

Configuring NQS Interoperation

NQS (Network Queuing System) is a UNIX batch queuing facility that allows users to queue batch jobs to individual UNIX hosts from remote systems. This section describes how to configure and use LSF to submit and control batch jobs in NQS queues.

If you are not going to configure LSF to interoperate with NQS, you do not need to read this section.

While it is desirable to run LSF on all hosts for transparent resource sharing, this is not always possible. Some of the computing resources may be under separate administrative control, or LSF may not currently be available for some of the hosts.

An example of this is sites that use Cray supercomputers. The supercomputer is often not under the control of the workstation system administrators. Users on the

workstation cluster still want to run jobs on the Cray supercomputer. LSF allows users to submit and control jobs on the Cray system using the same interface as they use for jobs on the local cluster.

LSF queues can be configured to forward jobs to remote NQS queues. Users can submit jobs, send signals to jobs, check the status of jobs, and delete jobs that are forwarded to the remote NQS. Although running on an NQS server outside the LSF cluster, jobs are still managed by LSF Batch in almost the same way as jobs running inside the LSF cluster.

Registering LSF with NQS

This section describes how to configure LSF and NQS so that jobs submitted to LSF can be run on NQS servers. You should already be familiar with the administration of the NQS system.

Hosts

NQS uses a machine identification number (MID) to identify each NQS host in the network. The MID must be unique and must be the same in the NQS database of each host in the network. LSF uses the NQS protocol to talk with NQS daemons for routing, monitoring, signalling and deleting LSF Batch jobs that run on NQS hosts. Therefore, you must assign a MID to each of the LSF hosts that might become the master host.

To do this, perform the following steps:

- Step 1** Log in to the NQS host as the NQS System Administrator or System Operator.
- Step 2** Run the `nmapmgr` command to create MIDs for each LSF host that can possibly become the master host. List all MIDs available. See the NQS `nmapmgr(1)` manual page for a description of this command.

Users

NQS uses a mechanism similar to `ruserok(3)` to determine whether access is permitted. When a remote request from LSF is received, NQS looks in the `/etc/hosts.equiv` file. If the submitting host is found, requests are allowed as long as the user name is the same on both hosts. If the submitting host is not listed in the `/etc/hosts.equiv` file, NQS looks for a `.rhosts` file in the destination user's home

directory. This file must contain the names of both the submitting host and the submitting user. Finally, if access still is not granted, NQS checks for a file called `/etc/hosts.nqs`. This file is similar to the `.rhosts` file, but it can provide mapping of remote user names to local user names. Cray NQS also looks for a `.nqshosts` file in the destination user's home directory. The `.nqshosts` file has the same format as the `.rhosts` file.

NQS treats the LSF cluster just as if it were a remote NQS server, except that jobs never flow to the LSF cluster from NQS hosts.

For LSF users to get permission to run jobs on NQS servers, you must make sure the above setup is done properly. Refer to your local NQS documentation for details on setting up the NQS side.

`lsb.nqsmaps`

The `lsb.nqsmaps` file in the `LSB_CONFDIR/cluster/configdir` directory is for configuring inter-operation between LSF and NQS.

Hosts

LSF must use the MIDs of NQS hosts when talking with NQS servers. The `Hosts` section of the `LSB_CONFDIR/cluster/configdir/lsb.nqsmaps` file contains the MIDs and operating system types of your NQS hosts.

```
Begin Hosts
HOST_NAME      MID      OS_TYPE
cray001        1        UNICOS      #NQS host, must specify OS_TYPE
sun0101        2        SOLARIS     #NQS host
sgi006         3        IRIX        #NQS host
hostA          4        -           #LSF host; OS_TYPE is ignored
hostD          5        -           #LSF host
hostB          6        -           #LSF host
End Hosts
```

Note that the `OS_TYPE` column is required for NQS hosts only. For hosts in the LSF cluster, `OS_TYPE` is ignored; the type is specified by the `TYPE` field in the `lsf.cluster.cluster` file. The '-' entry is a placeholder.

User Name Mapping

LSF assumes that users have the same account names and user IDs on all LSF hosts. If the user accounts on the NQS hosts are not the same as on the LSF hosts, the LSF administrator must specify the NQS user names that correspond to LSF users.

The `Users` section of the `lsb.nqsmaps` file contains entries for LSF users and the corresponding account names on NQS hosts. The following example shows two users who have different accounts on the NQS server hosts.

```
Begin Users
FROM_NAME      TO_NAME
user7          (user7l@cray001 user7@sgi006)
user4          (suser4@cray001)
End Users
```

`FROM_NAME` is the user's login name in the LSF cluster, and `TO_NAME` is a list of the user's login names on the remote NQS hosts. If a user is not specified in the `lsb.nqsmaps` file, jobs are sent to the NQS hosts with the same user name.

Configuring Queues for NQS jobs

You must configure one or more LSF Batch queues to forward jobs to remote NQS hosts. A forward queue is an LSF Batch queue with the parameter `NQS_QUEUES` defined. *'Adding a Batch Queue'* on page 90 describes how to add a queue to an LSF cluster. The following queue forwards jobs to the NQS queue named `pipe` on host `cray001`:

```
Begin Queue
QUEUE_NAME    = nqsUse
PRIORITY      = 30
NICE          = 15
QJOB_LIMIT    = 5
UJOB_LIMIT    = ( )
CPULIMIT      = 15
NQS_QUEUES    = pipe@cray001
DESCRIPTION   = Jobs submitted to this queue are forwarded to NQS
               _QUEUES
USERS         = all
End Queue
```

You can specify more than one NQS queue for the `NQS_QUEUES` parameter. LSF Batch tries to send the job to each queue in the order they are listed, until one of the queues accepts the job.

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.

Handling Cray NQS Incompatibilities

Cray NQS is incompatible with some of the public domain versions of NQS. Different versions of NQS on Cray may be incompatible with each other. If your NQS server host is a Cray, some additional steps may be needed in order for LSF to understand the NQS protocol correctly.

If the NQS version on a Cray is NQS 80.42 or NQS 71.3, then no extra setup is needed. For other versions of NQS on a Cray, you need to define `NQS_REQUESTS_FLAGS` and `NQS_QUEUES_FLAGS` in the `lsb.params` file.

```
NQS_REQUESTS_FLAGS = integer
```

If the version is NQS 1.1 on a Cray, the value of this flag is 251918848.

For other versions of NQS on a Cray, do the following to get the value for this flag. Run the `NQS` command:

```
% qstat -h CrayHost -a
```

on a workstation, where `CrayHost` is the host name of the Cray machine. Watch the messages logged by Cray NQS (you need access to the NQS log file on the Cray host):

```
03/02 12:31:59 I pre_server(): Packet type=<NPK_QSTAT(203)>.
03/02 12:31:59 I pre_server(): Packet contents are as follows:
03/02 12:31:59 I pre_server(): Npk_str[1] = <>.
03/02 12:31:59 I pre_server(): Npk_str[2] = <platform>.
03/02 12:31:59 I pre_server(): Npk_int[1] = <1392767360>.
03/02 12:31:59 I pre_server(): Npk_int[2] = <2147483647>.
03/02 12:31:59 I show_qstat_flags(): Flags=SHO_R_ALLUID SHO_R_SHORT
```

```
SHO_RS_RUN SHO_RS_STAGE SHO_RS_QUEUED SHO_RS_WAIT SHO_RS_HOLD SHO_RS_ARRIVE  
SHO_Q_BATCH SHO_Q_PIPE SHO_R_FULL SHO_R_HDR
```

The value of `Npk_int[1]` in the above output is the value you need for the parameter `NQS_QUEUES_FLAGS`.

```
NQS_QUEUES_FLAGS = integer
```

To get the value for this flag, run the NQS command:

```
% qstat -h CrayHost -p -b -l
```

on a workstation, where `CrayHost` is the host name of the Cray machine. Watch the messages logged by Cray NQS (you need to have access to the Cray NQS log file):

```
03/02 12:32:57 I pre_server(): Packet type=<NPK_QSTAT(203)>.  
03/02 12:32:57 I pre_server(): Packet contents are as follows:  
03/02 12:32:57 I pre_server(): Npk_str[1] = <>.  
03/02 12:32:57 I pre_server(): Npk_str[2] = <platform>.  
03/02 12:32:57 I pre_server(): Npk_int[1] = <593494199>.  
03/02 12:32:57 I pre_server(): Npk_int[2] = <2147483647>.  
03/02 12:32:57 I show_qstat_flags(): Flags=SHO_H_ACCESS SHO_H_DEST  
SHO_H_LIM SHO_H_RUNL SHO_H_SERV SHO_R_ALLUID SHO_Q_HDR SHO_Q_LIMITS  
SHO_Q_BATCH SHO_Q_PIPE SHO_Q_FULL
```

The value of `Npk_int[1]` in the above output is the value you need for the parameter `NQS_QUEUES_FLAGS`.

If you are unable to get the required information after running the above NQS commands, make sure that your Cray NQS is configured properly to log these parameters. To do this, run:

```
% qmgr
```

and enter `show all` to get all information. The parameters related to the logging of the information you need are:

```
Debug level = 3  
MESSAGE_Header = Short
```

MESSAGE_Types:

Accounting	OFF	Checkpoint	OFF	Command_flow	OFF
CONfig	OFF	DB_Misc	OFF	DB_Reads	OFF
DB_Writes	OFF	Flow	OFF	NETWORK_Misc	ON
NETWORK_Reads	ON	NETWORK_Writes	ON	OPer	OFF
Output	OFF	PACKET_Contents	ON	PACKET_Flow	ON
PROTOCOL_Contents	ON	PROTOCOL_Flow	ON	REcovery	OFF
REQuest	OFF	ROuting	OFF	Scheduling	OFF
USER1	OFF	USER2	OFF	USER3	OFF
USER4	OFF	USER5	OFF		

Support for Atria ClearCase

Many sites use Atria's ClearCase environment for revision source control and development. A user uses the `cleartool` command to startup a ClearCase view. After the view is created, the user is presented with a file system containing the user's sources and binaries. The file system is not accessible outside the view. `cleartool` has an option to start up a view and run a command under the view.

LSF's job starter can be used to set up a view then run the command (see *'Job Starter'* on page 227 for further information). For example, if you create a script "clearstarter" similar to the following:

```
#!/bin/sh
if [ x_${CLEARCASE_ROOT} = x_ ]; then
    cd $LS_SUBCWD
    $*
else
    /usr/atria/bin/cleartool setview \
        -exec "cd $LS_SUBCWD;$*" \
        `basename $CLEARCASE_ROOT`
fi
```

And specify it as a job starter, the user's job will run by LSF using the command line:

```
clearstarter myjob
```

which sets up a view the same as the user's on submission host, changes directory to the same as on submission host, then runs the job. The remote job runs in the same view as on local host.

For interactive jobs, the user sets the environment variable `LSF_JOB_STARTER` to the ClearCase job starter. The RES on the remote host then will run the user's job via the job starter. After the job starter is set, `lsmake` can run transparently in ClearCase view.

There are three steps to run an interactive job through the RES in a ClearCase view:

Step 1 Write a ClearCase job starter script (see example above).

Step 2 Set the `LSF_JOB_STARTER` environment variable. This can be done by each user or as part of the login process. For example:

```
% setenv LSF_JOB_STARTER clearstarter
```

Step 3 Run the job. For example:

```
% lsmake -j 4 -V -f foo.mak
```

To run a batch job in ClearCase view, the `csub` command should be used instead of `bsub`. With `csub`, no job starter needs to be used (look in the `$LSF_INDEP/misc/examples/clearcase` directory for the files `clearstarter` and `csub`. `LSF_INDEP` is defined in the `lsf.conf` file).

`csub` checks whether the environment variable `CLEARCASE_ROOT` is set. If it is set, which means the job is submitted from a view, it wraps the user's job as following:

```
cleartool setview -  
exec "cd $LS_SUBCWD; job" `basename $CLEARCASE_ROOT`
```

and passes all options to `bsub`, except `-i`, `-o` and `-e`. These three options will be translated to shell I/O redirection. For example, suppose `CLEARCASE_ROOT=/view/myview` and the user enters:

```
% csub -q myqueue -o myout -i myin myjob
```

`csub` will translate this into:

```
bsub -q myqueue cleartool setview -  
exec "cd $LS_SUBCWD; myjob < myin > myout \  
  
2>&1" myview
```

An alternative way is to configure a queue-level job starter (define `JOB_STARTER` in the file `lsb.queues`; see ‘*Queue-Level Job Starters*’ on page 129 for details), then use `bsub` to submit the job.

Using LSF Without Shared File Systems

Some networks do not share files between hosts. LSF can still be used on these networks, with reduced fault tolerance.

You must choose one host to act as the LSF master host. The LSF configuration files and working directories must be installed on this host, and the master host must be listed first in the `lsf.cluster.cluster` file.

To install on a cluster without shared file systems, follow the complete installation procedure on every host to install all the binaries, manual pages, and configuration files. After you have installed LSF on every host, you must update the configuration files on all hosts so that they contain the complete cluster configuration. The configuration files must be the same on all hosts.

Some fault tolerance can be introduced by choosing more than one host as possible master hosts, and using NFS to mount the LSF Batch working directory on only these hosts. All the possible master hosts must be listed first in the `lsf.cluster.clustername` file. As long as one of these hosts is available, LSF Batch continues to operate.

D. LSF on Windows NT

This appendix describes how to run LSF on Windows NT. It is assumed that you are already familiar with LSF concepts, and have installed LSF on Windows NT following the instructions in the *LSF Installation Guide*.

Requirements

- You must use a domain account (as opposed to a local account) when interacting with .
- Users must enter their passwords into an encrypted database maintained by LSF and any changes to Windows NT passwords must be reflected in the password database used by LSF.

Recommended

- The Windows NT Resource Kit contains many useful utilities (for example, `pview`) for monitoring processes.
- A `telnet` daemon to enable remote login sessions or some other form of remote access software to allow for easier management.

Differences Between LSF for UNIX and NT

- The shell used to invoke commands is `cmd.exe` instead of `/bin/sh` as on UNIX. For example, the queue-level `pre` and `post-exec` commands are invoked as:

D LSF on Windows NT

```
cmd.exe /C pre-exec command
```

- The NULL device on Windows NT is NUL rather than /dev/null as on UNIX. LSF translates /dev/null to NUL for Windows NT.
- The /etc directory on UNIX corresponds to the %SYSTEMROOT% directory on NT.
- On UNIX, LSF always uses /tmp as the temporary directory. On Windows NT, the temporary directory used by LSF can be configured by setting LSF_TMPDIR as a system environment variable. If that variable is not found, LSF goes to the next item in the following list, until a directory is defined:

```
LSF_TMPDIR environment variable
LSF_TMPDIR variable in the lsf.conf file
TMP environment variable (C:\temp by default)
TEMP environment variable (C:\temp by default)
%SYSTEMROOT%
```

- There is no native support in Windows NT for UNIX-style signals. Therefore sending an arbitrary signal to a job via the -s option of bkill has no meaning on Windows NT. LSF, however, supports the job control functionality by providing the equivalent of SIGSTOP, SIGCONT, and SIGTERM to suspend, resume, and terminate a job. These can be accessed through the commands bstop, bresume, and bkill.
- The UNIX umask parameter is ignored on Windows NT.
- When inputting commands to bsub, remember that the syntax of the commands must be specified in the form understood by Windows NT batch files. For example to specify multiple commands in a single line, use '&&' as the command separator instead of ';' as in UNIX. For example, use:

```
bsub 'cmd1 && cmd2'
```

instead of:

```
bsub 'cmd1; cmd2'
```

Also when specifying commands from standard input, use CTRL-Z to indicate EOF. On UNIX, CTRL-D is used. For example:

```
c:\temp> bsub -q simulation
bsub> myjob arg1 arg2
bsub> ^Z
```

- The `tmp` index returned by `lim` measures the space on the drive specified by the `TEMP` system environment variable.

File Permissions

The directories `work`, `logs`, `bin`, `lib`, etc, and `conf`, are all subdirectories of your LSF directory.

For all LSF files, Platform Computing recommends that you give full control permission to the Domain Admins user group. Other permissions should be set as shown:

work, logs

```
LSF primary administrator: ..... full control (All) (All)
domain administrator:..... full control (All) (All)
everyone:..... special access (R) (R)
```

bin, lib, etc

```
LSF primary administrator: ..... full control (All) (All)
domain administrator:..... full control (All) (All)
everyone:..... special access (RX) (RX)
```

conf

```
LSF primary administrator: ..... full control (All) (All)
domain administrator:..... full control (All) (All)
```

Mail

When LSF needs to send email to users, it invokes the program defined by `LSB_MAILPROG` in the `lsf.conf` file (in the `etc` subdirectory). If `LSB_MAILPROG` is not defined, no email is sent.

To use email, you need to use LSF's `lsmail.exe` program, which can send email to a UNIX host by using the Windows NT `rsh` utility (`%WINDIR%\system32\rsh`) to invoke `sendmail(1)` on the UNIX host. In order for this to work, the UNIX machine must be set up to allow the NT `rsh` client to run on it.

To support this method of sending email, `lsmail.exe` should be copied to a file corresponding to the name of the UNIX host. For example,

```
copy lsmail.exe unixhost.exe
```

Here `unixhost` is a UNIX machine which supports `sendmail(1)`. The `LSB_MAILPROG` should correspond to the `unixhost.exe` file. For example:

```
LSB_MAILPROG=//serverA/tools/lsf/bin/unixhost.exe
```

See '`LSB_MAILPROG`' on page 162 for details on how `LSB_MAILPROG` is invoked.

The `cmd.exe` Program

The command shell (`cmd.exe`) under Windows NT 4.0 cannot be started from a directory which is specified as a UNC name. For example, if you type the following on the command line, `cmd.exe` will end up starting in the directory specified by `%WINDIR%`, the system root directory of the current machine.

```
start /d\\serverA\share\username cmd.exe
```

As a result, jobs submitted from a shared directory will not start in the correct directory on the execution host.

The command shell from Windows NT 3.51, however, does support this feature. Microsoft has confirmed that this is a bug in NT 4.0, and included a fix in service pack 3 (refer to the article Q156276 in the Microsoft Knowledge Database for information).

In order for LSF to work correctly on Windows NT 4.0 machines, you can use one of three methods.

- Update your Windows NT 4.0 installation with service pack 3. LSF modifies the appropriate registry keys mentioned in article Q156276 to allow the UNC path to work.
- Replace your existing Windows NT 4.0 `cmd.exe` with the `cmd.exe` from service pack 3. The `cmd.exe` file typically resides in the `%WINDIR%\system32` directory. LSF modifies the appropriate registry keys mentioned in article Q156276 to allow the UNC path to work.
- Copy the Windows NT 3.51 `cmd.exe` into the `%WINDIR%\system32` directory under another name, e.g. `cmd351.exe`, and set the `LSF_CMD_SHELL` variable in the `lsf.conf` file to tell LSF to use this shell instead of `cmd.exe`.

For example, put the following line into the `lsf.conf` file:

```
LSF_CMD_SHELL=cmd351.exe
```

Heterogeneous NT/UNIX Environments

User Accounts

To run jobs in a mixed cluster, LSF users should have a user account with the same user name on UNIX and Windows NT. It is particularly important that the LSF primary administrator user account always have the same user name on both platforms.

Configuration Files

Note

If you used the Windows NT version of LSF Setup to create the UNIX/NT mixed cluster, as described in the LSF Installation Guide, the following settings have already been configured.

The LSF configuration files must be accessible from both NT and UNIX hosts. You can set up a shared file system between the UNIX and NT machines via NFS client on NT or an SMB server on UNIX, or, alternatively, you can replicate the configuration files. No matter how you arrange your configuration files, you must make sure that the port numbers (LSF_LIM_PORT, LSF_RES_PORT, LSF_SBD_PORT and LSF_MBD_PORT) defined in the `lsf.conf` file are the same on both UNIX and NT.

For example, if you use an SMB server on the UNIX side, you would simply set the three variables—LSF_CONFDIR, LSB_CONFDIR, and LSF_SHAREDIRE—in the `lsf.conf` file to point to the corresponding directories used by the UNIX hosts. The LSF_CONFDIR and LSB_CONFDIR directories must be accessible to all users (read permission). However, only the LSF primary administrator should have full control of these directories (read and write permissions).

Environment Variables

By default, LSF transfers environment variables from the submission host to the execution host. However, some environment variables are not applicable to another operating system.

When submitting a job from a Windows NT machine to a UNIX machine, the `-L` option of the `bsub` command can be used to reinitialize the environment variables. If submitting a job from a UNIX machine to a NT machine, you can set the environment variables explicitly in your job script. Alternatively, the Job Starter feature can be used to reset the environment variables before starting the job. LSF automatically resets the PATH on the execution host if the submission host is of a different type.

If the submission host is Windows NT and the execution host is UNIX, then the PATH variable is set to `/bin:/usr/bin:/sbin:/usr/sbin` and LSF_BINDIR (if defined in the `lsf.conf` file) is appended to it. If the submission host is UNIX and the execution host is Windows NT, the PATH variable is set to the system PATH variable with LSF_BINDIR appended to it.

Cross-Platform Daemon Startup

The `lssrvctrl.exe` binary only works when invoked from a Windows NT host. You will not be able to start LSF daemons on a Windows NT machine from a UNIX host.

Signal Conversion

LSF supports signal conversion between UNIX and Windows NT for remote interactive execution through RES (when you are using `lssrun` and `bsub -I`).

On Windows NT, the CTRL+C and CTRL+BREAK key combinations are treated as signals for console applications (these signals are also called console control events). LSF supports these two NT console signals for remote interactive execution, i.e. on the execution host LSF regenerates these signals for users' tasks. In a mixed NT/UNIX environment, LSF has the following default conversion between the NT console signals and the UNIX signals:

Table 3. UNIX/NT Signal Conversion

Windows NT	UNIX
CTRL+C	SIGINT
CTRL+BREAK	SIGQUIT

For example, if you issue `lssrun` or `bsub -I` commands from an NT console, but the task is running on a UNIX host, pressing the CTRL+C keys will generate a UNIX SIGINT signal to your task on the UNIX host. The reverse is also true.

Custom Signal Conversion

For `lssrun` (but not `bsub -I`), LSF allows users to define their own signal conversion using the following two environment variables.

- `LSF_NT2UNIX_CLTRC`
- `LSF_NT2UNIX_CLTRB`

For example, suppose a user sets the following:

D LSF on Windows NT

```
LSF_NT2UNIX_CLTRC=SIGXXXX
```

```
LSF_NT2UNIX_CLTRB=SIGYYYY
```

Here, SIGXXXX/SIGYYYY are UNIX signal names such as SIGQUIT, SIGTTIN, etc. The conversions will then be: CTRL+C = SIGXXXX and CTRL+BREAK = SIGYYYY.

If both LSF_NT2UNIX_CLTRC and LSF_NT2UNIX_CLTRB are set to the same value, (LSF_NT2UNIX_CLTRC=SIGXXXX and LSF_NT2UNIX_CLTRB=SIGXXXX), then on the Windows NT execution host, CTRL+C will be generated.

For `bsub -I`, there is no conversion other than the default conversion.

Starting Services and Daemons

The LSF service and daemons on each LSF server host will start automatically when the machine is restarted.

If you cannot restart each host at this time, log on as an LSF cluster administrator (a member of the LSF Global Administrators group) and start the LSF service and daemons manually.

Note

You should not use the primary LSF administrator's account (normally `lsfadmin`) to start or stop LSF service and daemons.

To start the LSF service and daemons, use any one of the following methods:

- Use the Windows NT Server Manager to start “LSF Service” on all LSF server hosts.
- Click “Services” on the Windows NT Control Panel and start “LSF Service”. You will have to repeat this step on each LSF server host.
- Where LSF Batch has been installed, go to the “LSF Suite for Workload Management/LSF Batch” program folder, and use the LSF administrative tool

“LSF Batch Administration”. (You can use this tool to perform all your administrative tasks for LSF Base and LSF Batch products.)

- Start a new command console, and type:

```
lssrvctrl start -m all lssrvman
```

Usage information for `lssrvctrl` is available by typing `lssrvctrl` with no options.

Using LSF

Each user who wants to use LSF needs to supply the password of his/her domain user account. Use the `lspasswd.exe` command, and follow the instructions. For example:

```
lspasswd [-u user_name]
```

If you do not specify the `-u` option as above, the user is assumed to be the current user.

In addition, all users need to have the “Logon as a batch job” privilege on every LSF server host. For this purpose, you can simply put all LSF users into the ‘LSF user group’ created for or assigned by you during the installation. The LSF user group has the “Logon as a batch job” privilege on all LSF server hosts.

Miscellaneous

- The machines running LSF are expected to have fixed IP addresses. If you use DHCP to assign IP addresses dynamically, LSF can still work, provided the reassigned IP address of an LSF host does not change.
- When using LSF’s remote execution functions through the Remote Execution Server, there is no support for `pty`-type options for `lrun` and `bsub -I`, i.e. the `-P` and `-S` options for `lrun` and `-Ip` and `-Is` options for `bsub` are not supported.

D LSF on Windows NT

- If you log on as the LSF primary administrator from the console while the LSF service is running from a file server over the network, and then log off again, the LSF service and daemons may die on that host. Logging off appears to cause Windows NT to close all network connections for the LSF primary administrator user account, including those used by an LSF service or LSF daemons.
- When writing an external command that is invoked by LSF (for example, `elim`, `esub`, or `eexec`), the command must be a binary executable, that is, `elim.exe` or `esub.exe`. It cannot be a batch file such as `elim.bat`.
- `LSF_USE_HOSTEQUIV` parameter in `lsf.conf` is ignored on Windows NT.
- Nice values specified at the queue-level through the `NICE` parameter are mapped to NT process priority classes as follows:

`nice>=0` corresponds to an NT priority class of `IDLE`
`nice<0` corresponds to an NT priority class of `NORMAL`

LSF does not support `HIGH` or `REAL-TIME` priority classes.

- The `io` index shows 0, unless the disk performance counters are turned on. To turn on disk performance counters, use the `DISKPERF` command.

Note

Turning on the performance counters incurs extra overhead in disk I/O.

- 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 OS at the process level.
- The UNIX man pages, converted to HTML format, are stored in the `html` subdirectory of your LSF directory.

E. The LSF SNMP Agent

This appendix describes how to use the LSF SNMP (Simple Network Management Protocol) agent. It is assumed that you are already familiar with SNMP concepts.

About the Agent

To integrate with existing network and system management frameworks, LSF supports SNMP, an IETF (Internet Engineering Task Force) standard protocol used to monitor and manage devices and software on the network. Platform has also defined a Management Information Base (MIB) specific to LSF.

Any SNMP client, from command-line utilities to full network and system management frameworks, can monitor information provided by the LSF SNMP agent. It does this by reading or modifying the values of objects in the LSF MIB. The MIB defines these variables according to internet standards RFC 1155 and RFC 1212, so any fully compliant SNMP client can query the LSF MIB.

In addition, the LSF SNMP agent supports the Internet standard Management Information Base (MIB-II) for use in TCP/IP based internets (RFC 1213). This allows the network manager to run the LSF agent while still being able to retrieve basic MIB-II statistics.

Requirements

The agent can be installed on any LSF host running UNIX and it is compatible with all SNMP version 1 clients, including HP OpenView, CA Unicenter TNG, and Tivoli TME 10. For information about a particular network or system management framework, please refer to the documentation supplied by the vendor.

E The LSF SNMP Agent

Distribution

The agent is available from Platform's FTP site. Installation instructions are included. The following files are provided by Platform:

- the SNMP agent (`snmpd`)
- the LSF MIB (`LSF_CONFDIR/snmp/lsf-agent-mib.txt`)
- a configuration file (`LSF_CONFDIR/snmp/snmpd.conf`)
- a script to start the SNMP agent (`lsfsnmpd`)

Starting the Agent

To simplify the startup process, the LSF SNMP binary file (`snmpd`) is accompanied by a script (`lsfsnmpd`). This script is customized to start the agent in a specific LSF environment. For example, it contains the location of the configuration file used by the agent, (`LSF_CONFDIR/snmp/snmpd.conf`), and the log file that is created when the agent runs (`LSF_LOGDIR/snmpd.log`).

The LSF administrator can modify the parameters in the `lsfsnmpd` script, or run the `snmpd` binary file without using the script.

Options which can be set in this file include:

- p *port_number***
Listen on port *port_number* (default: port 161).
- l *log_file***
Log all output from the agent (including stdout/err) to *log_file*.
- c *conf_file***
Read *conf_file* as a configuration file.
- f**
Don't fork from the calling shell.

-L

Don't open a log file; use stdout/err instead.

Structure of the LSF MIB

The LSF MIB (`LSF_CONFDIR/snmp/lsf-agent-mib.txt`) consists of several tables of information, organized into three groups:

- LSF Base host information (`lsfHosts` group)
- LSF Base resource information (`lsfResources` group)
- LSF Batch information (`lsfBatch` group)

The `lsfHosts` MIB Group

`lsfStaticTable`

Consists of one row for each LSF host, indexed by host IP address. Each row contains static host information, corresponding to the `lshosts` command.

`lsfDynamicTable`

Consists of one row for each LSF server, indexed by host IP address. Each row contains dynamic host information, (corresponding to the `lsload` command) composed of the built-in load indices and the host status.

The `lsfResources` MIB Group

`lsfNumericTable`

Consists of several rows for each resource, one for each LSF host using that resource, indexed by resource number (generated by the agent) and host IP address. Each row contains the name of a numeric shared resource or external index, a location (a host using the resource), and the resource value.

For shared resources, the resource value is the same for all the hosts that share the resource instance.

The IsfBatch MIB Group

IsbHostsTable

Consists of one row for each LSF batch server, indexed by host IP address. Each row contains the host limits as well as the host counters.

IsbQueuesTable

Consists of one row for each LSF batch queue, indexed by a number generated by the agent (corresponding to alphabetical order of queue names). Each row contains queue limits and queue counters.

IsbJobsTable

Consists of one row for each running batch job, indexed by job ID (for performance reasons, only running jobs are displayed). Each row contains information such as the queue, user and execution hosts, and job resource information.

Optional Configuration of the Agent

The configuration file (`LSF_CONFDIR/snmp/snmpd.conf`) has the format of one directive per line. Lines preceded by the '#' character are treated as comments, and not parsed.

Directives which can be set in this file are:

syslocation *string*

This sets the system location for the agent in the system table of the MIB-II tree to *string*.

syscontact *string*

This sets the system contact for the agent in the system table of the MIB-II tree to *string*.

trapsink *host*

This sets the host *host* to receive traps (e.g., the agent sends a Cold Start trap when it starts up). To enable multiple hosts to receive traps, add a new line for each additional host. The default value is null (no hosts receive traps).

trapcommunity string

This sets the community string in the trap PDU (Protocol Data Unit) to *string*.

authtrapenable number

This enables the sending of authentication failure traps when *number* is set to 1 (enable). The default value is 2 (disable).

community number string

This sets the community string in slot *number* to *string*.

The agent has 5 slots available to keep community strings, so the acceptable values for *number* are from 1 to 5. SNMP PDUs sent to the agent should contain one of the communities in the 5 slots. The default values for each slot are:

- 1) public
- 2) private
- 3) regional
- 4) proxy
- 5) core

If the agent receives a PDU without a known community, it will discard the request, and if `authtrapenable` is set to 1, it will generate an authentication failure trap.

Index

`$HOME/.rhosts` 15
`%USRCMD` 227
`/etc/hosts.equiv` 41

A

address (Platform) xv
AFS (Andrew File System) 6, 42
agent, SNMP 290
Atria ClearCase
 `CLEARCASE_ROOT` 276
 `cleartool` 275
 `csub` 276
authd 11
authentication 10
 DCE client using GSSAPI 12
 Kerberos 4 12
authentication, in a mixed UNIX/NT
 cluster 166
automount 6, 240

B

`bacct` 80
backfilling 32, 40
`badmin` 62, 80, 87, 90, 244
`badmin hclose` 86, 89
`badmin hhist` 86
`badmin hist` 84
`badmin hopen` 86
`badmin hrestart` 85
`badmin hshutdown` 85, 89
`badmin hstartup` 49
`badmin qact` 87
`badmin qinact` 87

`badmin reconfig` 85, 90
batch job state 20
batch jobs
 changing execution order 96
 checkpoint, restart and migration 37,
 224
 dispatching order 25
 exclusive 223
 file access 40, 42
 killing and signalling 97
 migration, *see* `bmig`
 moving to other queues 96
 pending and suspended 20
 post-execution conditions 37
 preemptable 223
 preemption 113
 pre-execution conditions 36
 resuming 35
 scheduling algorithm 111
 scheduling priority 208
 state diagram 20
 suspended by LSF Batch 111
 suspending 33
batch server host 88
`bbot` 25, 96
`bclusters` 153
benchmarks 68
`bhist` 80, 155
`bhosts` 23, 24, 33, 155
`bhosts -l` 111
bhosts, closed status 83
`bhpart` 114
binary file
 `snmpd` 290
`bjobs` 21, 22, 24, 30, 35, 154
`bjobs -lp` 111

Index

bkill.....35, 89, 97
bmig.....38
bqueues23, 24, 33, 86, 154
bresume36, 97
brun98
brun(1)83
bstop.....35, 97
bsub.....5, 13, 14, 17, 24, 36, 40
bsub -x32
bswitch90, 96
bswitch -q.....91
bswitch(1).....97
btop.....25, 96
btop(1)97
built-in load indices43
busy thresholds.....69

C

CA Unicenter.....289
capacity planning.....71
cc7
checkpointing37, 204, 224
chkpnt269
CLEARCASE_ROOT environment
 variable276
cleartool.....275
client hosts.....171, 183
client_addr11
client_port12
closed status in bhosts83
cluster.....3
Cluster Administrator99
clusters
 adding hosts.....56
 creating58
 removing hosts57
command.....3
command-level job starter.....17
community (SNMP)293

computer.....3
configuration
 adding a batch queue88
 adding hosts56
 changing55
 creating new clusters.....58
 example queues136
 file formats52
 horizontal section.....52
 hosts file.....188
 lsb.hosts file202, 271
 lsb.nqsmaps file235
 lsb.queues file208
 lsb.users file198
 lsf.conf file.....161
 lsf.shared file173
 lsf.task file.....58
 removing a batch queue90
 removing hosts.....57
 resource requirements.....58
 specifying default values53
 tuning busy thresholds69
 tuning LSF Batch108
 vertical section53
configuration file
 SNMP.....292
configuration lsf.cluster.*cluster* file
 178
Configure Base tab102
Configure Batch tab103
Configuring Hierarchical Fairshare . 117
configuring LSF.....102
contacting Platform Computing xv
counted software licenses131
CPU factor174, 217
 tuning.....68
CPU Run Queue Length.....111
CPU run queue length68, 70
CPU time limit.....194, 233
CPU utilization111

csub.....276

D

daemons

authd.....11
error logs.....45, 168, 239
lmgrd.....73
pidentd.....11
shutting down.....49
syslogd.....168

Days.....8

DCE.....6, 42

Deadline Constraint Scheduling212

default.....114

default host specification.....232

default resource requirement.....183

default shell

/bin/sh on UNIX.....279
cmd.exe on NT.....279

definition

master host.....81

DEMO license.....72

DFS (Distributed File System).....6

directories

remote access.....40, 42
user accounts.....7

dispatch windows.....23

hosts.....203
queues.....210

dispatching batch jobs.....25

DNS (Domain Name Service).....188

documentation.....xv

domain names.....171

DONE batch job state.....21

duplicate event logging.....81

E

eauth.....11

eauth.....12, 14, 15, 164

echkpt.....38

eexec.....42

effective run queue length.....71, 111

egroup.....43

ELIM.....43, 179

END.....42

environment variables

CLEARCASE_ROOT.....276

HOSTRESORDER.....243

LS_EXEC_T.....42

LS_JOBPID.....42

LSB_CHKPNT_DIR.....126

LSB_JOBFILENAME.....268

LSB_JOBPGIDS.....229

LSB_JOBPGIDS.....229

LSB_SUSP_REASON.....229

LSF_ENVDIR.....167

LSF_JOB_STARTER.....17

PATH.....7, 128, 167

WINDIR.....128

erestart.....38

error logs.....45, 239

error message

"Cannot locate master LIM
now".....242

"chdir(...) failed: no such
file or directory" 243

"Communication time out" 241

"User permission denied".14,
242

User permission denied... 16

esub.....42

/etc/hosts file.....243, 188

/etc/hosts.equiv file... 15, 172, 243

/etc/lsf.conf file.....45

/etc/services file.....167

Index

/etc/syslog.conf file 46, 168
exclusive requeue 232
exclusive scheduling 32
execution host 4
EXIT batch job state 21
external authentication
 external key 165
external authentication (eauth) 11
External Host Groups 206
external key (eauth) 165

F

FAIRSHARE 114
Fairshare Policy 113
fairshare scheduling 31
fault tolerance 277
fax numbers (Platform) xv
FEATURE 75
file sharing 7
First-Come-First-Serve 118
FLEXlm 72, 167
 log file 46
 network ports 75
 updating licenses 75
 utilities 74
floating license 132
forward queue 140, 233, 272
front end queue example 139

G

gid 11
Globetrotter Software, *see* FLEXlm
guides xv

H

help xv

heterogeneous NT/UNIX environments
 283

Hierarchical Fairshare 117

horizontal configuration section 52

Host 29

Host 19, 26, 28

host 3

 execution host 4

 local 4

 master 4, 5, 81

 remote 4

 submission host 4

host dispatch windows 203

host ftp.lysator.liu.se 11

host group 88

host locked licenses 131

host partitions 31, 88, 206

host preference 221

host status

 lockW 184

host status closed 83

host thresholds 24

HOSTRESORDER environment variable .

 243

hosts

 adding to a cluster 56

 client 171, 183

 configuring 57, 58

 enabling and disabling 47

 host names 171

 job limits 209

 job slot limits 26

 naming 188

 removing from a cluster 57

hosts.equiv file 15

hosts.equiv(5) 16

hour 8

HP Exemplar Technical Servers 265

HP OpenView 289

HTML help files (Windows NT) 288

I	
IBM SP-2	263
ident	164
identification protocol	10, 164
idle hosts, sharing	111
idle queue example	136
IGNORE_DEADLINE	213
installation	
client hosts	171
executable files	165
manual pages	169
servers	171
it	34
load index	34, 111
idle time	110
J	
JL/P	28
JL/U	28
job	3
<i>see</i> batch jobs	
job ladders, <i>see</i> batch jobs, pre-execution conditions	
job limits	200, 209
job scheduling	212
job slot	26
job slot limits	26
job starter	16
command-level	17
queue-level	17, 129, 227
job state	
DONE	21
EXIT	21
PEND	20
PSUSP	22
RUN	20
SSUSP	22
USUSP	22
JOB_CONTROLS	39
JOB_STARTER	18, 130, 227
K	
killing jobs	97
L	
liblsf.a	167
license key	72
license management	74
license queue example	138
license.dat file	167
LIM	1
selection of master	182
LIM policy	51
lim.acct file	71
lim.log.hostname file	46, 240
limitations	
number of remote connections ..	10
lmcksum	75
lmdown	75
lmgrd	73, 75
lmhostid	75
lmremove	75
lmreread	75
lmstat	75
lmver	75
load index	
pg	34
load indices	9
overriding built-in	66
load thresholds	69
LSF Batch queues	216
paging rate	70
run queue length	70
set in <code>lsf.cluster.cluster</code> file	182

Index

- local host 4
- lockW host status 184
- log files 45
 - FLEXlm 46
 - lim.log *hostname* 46
 - maintenance 45
 - res.log *hostname* 46
 - SNMP 290
- LOG_DAEMON 46, 168
- lost and found queue 90
- ls_connect 42
- LS_EXEC_T environment variables 42
- LS_JOBPID environment variable 42
- lsadmin 11, 62, 82, 245
 - limstartup 49
 - reconfig 62
 - resstartup 49
- lsb.acct 80
- lsb.acct file 36
- lsb.events 5, 6, 80
- lsb.events file 5
- lsb.host 28
- lsb.hosts file 19, 89, 202, 271
 - CHKPNT 204
 - DISPATCH_WINDOW 203
 - Host section 202
 - HOST_NAME 202
 - HostGroup section 205
 - HostPartition section 206
 - JL/U 28, 203
 - load thresholds 203
 - MIG 203
 - MXJ 28, 202
 - USER_SHARES 199
- lsb.nqsmaps file 235, 271
 - Hosts section 235
 - HOST_NAME 236
 - MID 236
 - OS_TYPE 236
 - Users section 237
 - FROM_NAME 237
 - TO_NAME 237
- lsb.params file
 - CLEAN_PERIOD 195
 - DEFAULT_HOST_SPEC 194
 - DEFAULT_PROJECT 194
 - DEFAULT_QUEUE 193
 - HIST_HOURS 114, 195
 - JOB_ACCEPT_INTERVAL... 25, 195
 - JOB_TERMINATE_INTERVAL.. 228
 - MAX_JOB_ARRAY_SIZE 196
 - MAX_JOB_NUM 80, 195
 - MAX_PREEEXEC_RETRY 149
 - MAX_SBD_FAIL 195
 - MBD_SLEEP_TIME 25, 194
 - NQS_QUEUES_FLAGS 197, 274
 - NQS_REQUESTS_FLAGS... 197, 273
 - PG_SUSP_IT 34, 196
 - SBD_SLEEP_TIME 30, 33, 194
- lsb.queues file 89, 90, 140, 208, 233, 272
 - ADMINISTRATORS 210
 - CORELIMIT 218
 - CPULIMIT 217
 - DATALIMIT 218
 - DEFAULT_HOST_SPEC 232
 - DESCRIPTION 233
 - DISPATCH_WINDOW 210
 - EXCLUDE (*value*) 232
 - EXCLUSIVE 223
 - FILELIMIT 218
 - HJOB_LIMIT 29, 123, 209
 - HOSTS 220
 - JOB_CONTROLS 229
 - RESUME 229
 - SUSPEND 229
 - TERMINATE 230
 - JOB_STARTER 268, 277
 - MAX_RESERVE_TIME[n] 211
 - MEMLIMIT 218
 - MIG 224

NICE	209	lsb.hosts	89
NQS_QUEUES	233	LSB_CONFDIR/cluster/configdir/ lsb.queues	90, 91
PJOB_LIMIT	29, 123, 209	LSB_JOBFILENAME environment variable	268
POST_EXEC	225	LSB_JOBPGIDS environment variable . 229	
PRE_EXEC	225	LSB_JOBPIIDS environment variable 229	
PREEMPTABLE	223	LSB_LOCALDIR	81, 82
PRIORITY	208	LSB_SHAREDIRE	81, 82
PROESSLIMIT	219	LSB_SHAREDIRE/cluster/logdir .	79
PROCLIMIT	219	LSB_SUB	
QJOB_LIMIT	29, 209	BEGIN_TIME	94
QUEUE_NAME	208	CHKPNT_DIR	92
RCVJOBS_FROM	148	CHKPNT_PERIOD	92
REQUEUE_EXIT_VALUES	231	DEPEND_COND	93
RES_REQ	213, 269	ERR_FILE	92
RESUME_COND	35, 215	EXCEPTION	95
RUN_WINDOW	210	EXCLUSIVE	92
RUNLIMIT	218	HOST_SPEC	93
SLOT_RESERVE	211	HOSTS	94
SNDJOBS_TO	148	IN_FILE	92
STACKLIMIT	218	INTERACTIVE	94
STOP_COND	215	JOB_NAME	92
SWAPLIMIT	219	LOGIN_SHELL	93
TERMINATE_WHEN	230	MAIL_USER	93
UJOB_LIMIT	29, 209	MAX_NUM_PROCESSORS	94
USER_SHARES	199, 222	MODIFY	93
USERS	199, 220	MODIFY_ONCE	93
lsb.queues file		NOTIFY_BEGIN	92
UJOB_LIMIT	29	NOTIFY_END	92
lsb.users	117	NUM_PROCESSORS	94
lsb.users file	116, 198	OTHER_FILES	94
JL/P	28	OTHER_FILES_nn	94
MAX_JOBS	28	OUT_FILE	92
USER	199	PRE_EXEC	93
User section	200	PROJECT_NAME	93
lsb.users file	16, 28	PTY	94
LSB_CHKPNT_DIR environment variable	126	PTY_SHELL	94
LSB_CONFDIR/cluster/configdir . 88, 89		QUEUE	92
LSB_CONFDIR/cluster/configdir/			

Index

- RERUNNABLE 93
- RES_REQ 93
- RESTART 93
- RESTART_FORCE 93
- RLIMIT_CORE 95
- RLIMIT_CPU 95
- RLIMIT_DATA 95
- RLIMIT_FSIZE 95
- RLIMIT_RSS 95
- RLIMIT_RUN 95
- RLIMIT_STACK 95
- TERM_TIME 94
- TIME_EVENT 94
- USER_GROUP 92
- WINDOW_SIG 93
- LSB_SUB_ABORT_VALUE 92
- LSB_SUB_PARM_FILE 91
- LSB_SUB_PARM_FILE **Option Names** .
92
- LSB_SUSP_REASON **environment**
variable 229
- .lsbatch **directory** 7
- lsclusters 150
- LSF administrator 16, 47, 96, 181
- LSF Base API 1
- LSF Base Tool 1
- LSF Base tools 19
- LSF Batch
 - administration, *see* LSF
administrator
 - queue configurations 136
 - queue definitions 208
 - tuning 108
- LSF Batch 82
- LSF Batch configuration files
 - lsb.hosts 199
 - lsb.params 194, 233
 - lsb.queues 194, 199, 233
 - lsb.users 199
- LSF Enterprise Edition xiv
- LSF master host 4, 5, 277
- LSF MIB 291
- LSF Service 13
- LSF SNMP agent 290
- LSF Standard Edition xiv
- LSF Suite documentation xv
- LSF Suite products xiii
- lsf.cluster.cluster 110, 111
- lsf.cluster.cluster **file** 144
- lsf.cluster.*cluster* **file** ... 5, 178, 277
 - backwards compatibility**
 - ClusterManager, *see*
ClusterAdmins
 - section**
 - Manager, *see* ADMINISTRATORS
 - ClusterAdmins **section** 181
 - ADMINISTRATORS 181
 - Host **section** 69, 182, 185
 - HOSTNAME 183
 - model 183
 - nd (**number of disks**) 183
 - RESOURCES 184
 - server 183
 - type 183
 - Hosts **section** 145
 - Parameters **section** 178
 - ELIM_POLL_INTERVAL ... 180
 - ELIMARGS 179
 - EXINTERVAL 180
 - FEATURES 146
 - HOST_INACTIVITY_LIMIT 180
 - MASTER_INACTIVITY_LIMIT
180
 - PROBE_TIMEOUT 181
 - PRODUCTS 179
 - RETRY_LIMIT 181
 - RemoteClusters **section**
 - CACHE_INTERVAL 147
 - CLUSTERNAME 147
 - EQUIV 147

RECV_FROM.....	147, 152
lsf.conf	13, 14, 15, 38, 41, 81
lsf.conf file	
LSF_CROSS_UNIX_NT.....	165
LSF_ECHKPNTDIR	166
lsf.conf file	
LSB_CONFDIR.....	161
LSB_DEBUG	162
LSB_LOCALDIR.....	164
LSB_MAILPROG.....	162, 282
LSB_MAILTO	163
LSB_MBD_PORT.....	167
LSB_SBD_PORT.....	167
LSB_SHAREDIR.....	163
LSF_AFS_CELLNAME.....	164
LSF_AUTH	14, 164
LSF_BINDIR.....	165
LSF_CONFDIR.....	165
LSF_ENVDIR.....	166
LSF_INCLUDEDIR	166
LSF_INDEP	166, 276
LSF_LIBDIR.....	167
LSF_LICENSE_FILE.....	167
LSF_LIM_DEBUG	167
LSF_LIM_PORT.....	167
LSF_LOG_MASK.....	168
LSF_LOGDIR.....	45, 168, 239
LSF_MACHDEP.....	169
LSF_MANDIR.....	169
LSF_RES_ACCT.....	170
LSF_RES_ACCTDIR.....	170
LSF_RES_DEBUG	170
LSF_RES_PORT.....	167
LSF_RES_RLIMIT_UNLIM.....	173
LSF_ROOT_REX.....	148, 170
LSF_SERVER_HOSTS.....	171
LSF_SERVERDIR	171
LSF_STRIP_DOMAIN.....	171
LSF_USE_HOSTEQUIV.....	172
XLSF_APPDIR.....	172
XLSF_UIDDIR	172
lsf.shared file.....	144, 173
Clusters section	145, 173
HostModel section.....	68, 174
HostType section	174
Resources section.....	175
lsf.sudoers file	
LSF_EAUTH_KEY.....	191
lsf.sudoers file.....	13
LSB_PRE_POST_EXEC_USER..	190
LSF_EAUTH_USER.....	191
LSF_EEXEC_USER.....	191
LSF_STARTUP_PATH	190
LSF_STARTUP_USERS	190
lsf.task file	58
LSF_AUTH	15
LSF_BINDIR	41
LSF_CROSS_UNIX_NT.....	165
LSF_EAUTH_KEY	12, 165, 191
LSF_EAUTH_USER	11
LSF_ECHKPNTDIR	38, 166
LSF_EEXEC_USER	42
LSF_ENVDIR	13
LSF_ENVDIR environment variable.	167
LSF_JOB_STARTER	17
LSF_SERVERDIR	12, 14, 38, 42, 43
LSF_STARTUP_PATH.....	13
LSF_STARTUP_USERS.....	13
LSF_USE_HOSTEQUIV.....	15
lsfadmin	13, 14
lsfdaemons start.....	14
.lsfhosts	16
.lsfhosts file	156
lsfsnmpd script	290
lsfstartup.....	49
lsgrun.....	18
lshosts.....	24, 47, 151
lsid.....	4
lsinfo.....	9
LSLIB.....	10, 11

Index

lsload 47, 69, 71, 112, 151
lslogin 110
lslogin 152
lsmon 47, 69, 71, 151
lsplace 151
lsrccp 40, 41
lsrun 4, 11, 14, 16, 17, 18, 42, 152
lstcsh 1
lstcsh 19

M

machine 3
mailing address (Platform) xv
Manage Base tab 100
Manage Batch tab 100
Management Information Base (MIB) 291
master host 4, 81
master LIM 182
MAX_JOB_ARRAY_SIZE parameter ..
 196
mbatchd 2, 5, 6, 14, 15, 30, 80
mbatchd.log . *hostname* file 240
MIB (Management Information Base) 291
MID (Machine Identification Number) ..
 270
migration 37, 224
 also see bmig
mixed UNIX/NT clusters 283
mpa 268
multiprocessor computer 3
multiprocessor hosts 29, 71, 202, 209, 217
MXJ 28

N

network failure 5
network management software 289
network monitoring 71

NFS (Network File System) 6, 240
night queue example 137
normalized run queue length 71
nosuid 14
NQS (Network Queuing System) ... 269
 forward queue example 140
 lsb.nqsmaps file 271
MID (Machine Identification
 Number) 270
NQS_QUEUES parameter ... 140, 233,
 272, 273
 server 270
 user name mapping 237, 272
NT commands
 cmd.exe 282
 DISKPERF 288
 lssrvcntrl.exe 285
NT environment
 signals 280
null device
 /dev/null on UNIX 280
 NUL on NT 280

O

online documentation xv
 HTML files 288
others 114
owners queue example 137

P

path
 /etc/hosts.equiv 15
 /etc/lsf.conf 14
 /etc/lsf.sudoers 11
 /net 40
 /usr/bin 7
 /usr/local/lsf/mnt 7

/usr/local/lsf/mt/bin/ ...	7
PATH environment variable ...	128, 167
PATH environment variables ...	7
PEND ...	20, 38
PEND state ...	22
periodic tasks ...	45
permanent license ...	72
permission ...	15
per-user job limit ...	200, 209
pg	
paging rate ...	109
pg load index ...	34
PG_SUSP_IT ...	110
phone numbers (Platform) ...	xv
pidentd ...	11
PIM (Process Information Manager) ...	30
PJOB_LIMIT ...	29
Platform Computing Corporation ...	xv
POE (Parallel Operating Environment) .	
263	
preemption ...	36
preemptive scheduling ...	32
privileged ports ...	10
procedures	
Activating and Inactivating Queues.	87
Adding a Batch Server Host ...	89
Opening and Closing of Batch Server	
Hosts ...	86
Opening and Closing Queues ...	87
Removing a Batch Server Host ...	89
Shutting Down LSF Batch Daemons	
85	
process ...	3
process migration, <i>see</i> <i>bmig</i>	
Processor reservation ...	32
processor set ...	257
PSUSP batch job state ...	22
pub/ident/server ...	11

Q

QJOB_LIMIT ...	29
queue dispatch windows ...	210
queue thresholds ...	24
QUEUE_NAME ...	90
queue-level job starter ...	17
queues	
activating and inactivating ...	87
adding a queue ...	88
checkpoint, restart and migration . .	
224	
configuring ...	208
description ...	233
eligible hosts and users ...	220
example configurations ...	136
exclusive ...	223
inter-queue priority ...	113
job slot limits ...	26
load thresholds ...	216
lost and found ...	90
moving jobs between ...	96
opening and closing ...	87
preemptable ...	223
removing a queue ...	90
resource limits ...	217
scheduling algorithm ...	111
scheduling priority ...	208
status ...	86

R

r15m ...	111
r15s ...	111
r1m ...	111
rcp ...	41
reconfiguring ...	55
remote command ...	4
remote execution, permission ...	15

Index

- remote host 4
- requeue 231, 232
- RES 1
- RES 42
- res.log *hostname* file 46, 240
- resource limits 217
- resource requirements 24
 - configuring 58
 - default 183
- resource reservation 39, 123, 214
- resources 58
- restart 269
- restarting 224
- RESUME 108
- RESUME_COND 35, 109
- RFC 1413 12
- RFC 1413 identification protocol . 10, 164
- RFC 1413 protocol 10
- RFC 93 protocol 10
- RFC 931 identification protocol . 10, 164
- .rhosts file 15, 41, 172, 243
- rlogin 10, 110
- rsh 10, 41
- RUN 26
- RUN batch job state 20
- run limit 212, 218
- run window 212
- run windows 23, 210
- RUNLIMIT 218
- ruserok 15, 172
- ruserok(3) 15, 16

- S**

- sbatch 89
- sbatchd 2, 5, 6, 30, 40, 85, 89
- sbatchd.log *hostname* file 240
- scheduling
 - exclusive 32
 - fairshare 31
 - host partition 199, 207, 222
 - loadSched 216
 - loadStop 216
 - migration of rerunnable jobs 38
 - PREEMPTABLE 223
 - preemptable 32
 - preemptive 32
 - RES_REQ 213
 - RESUME_COND 215
 - STOP_COND 215
- scheduling condition 24, 25
- scheduling jobs 212
- scheduling priority 208
- scheduling threshold 24, 216
- script
 - lfsnmpd 290
- security 12
 - /etc/hosts.equiv and .rhosts
172
 - remote access as root 170
 - user authentication 14, 164
- server status closed 83
- setuid permissions 10, 164, 242
- setuid root 13
- shared files 240
- shared resource 9
- sharing hosts 206
- short queue example 138
- SIGCONT 38, 97
- SIGKILL 38, 97
- signals 97
- SIGSTOP 38, 97
- SIGTERM 97
- SIGTSTP 97
- SIGXCPU 217
- Simple Network Management Protocol
(SNMP) 289
- SNMP (Simple Network Management
Protocol) 289
- snmpd binary file 290

snmpd.conf file	292	telephone numbers (Platform)	xv
snmpd.log file	290	temporary directory	
software licenses		/tmp on UNIX.....	280
counted	131	C:\temp on NT	280
DEMO	72	termination time	212
FLEXlm	72	thresholds	
floating	132	host	24
host locked	131	queue	24
permanent	72	scheduling and suspending	111
updating	75	time window	8
special user names		time-limited software license	72
others		/tmp directory.....	46, 168
default	114	/tmp_mnt directory.....	241
SSUSP.....	26	TNG Tivoli TME 10.....	289
SSUSP batch job state.....	22	traps (SNMP)	292
START.....	42	tuning	
Starting LSF on Windows NT.....	286	busy thresholds	69
static resources	9	LSF Batch.....	108
status			
of hosts	101		
of queues	101		
status closed in bhosts	83		
STOP_COND.....	112		
submission host	4		
support	xv		
SUSPEND	108, 109		
suspending condition	33		
suspending thresholds.....	33, 111, 216		
syscontact.....	292		
sysinfo	267		
syslocation.....	292		
syslog	45, 168, 239		
System Administrator	269, 270		
system management software	289		
T		U	
task	3	uid	11
task migration	224	UJOB_LIMIT	29
technical assistance.....	xv	UNIX/NT user groups.....	198
		User	26
		user authentication	10
		eauth	14
		ident	14
		user directories	7
		user groups.....	116
		user job slot limits.....	26
		user names	237, 272
		user_auth_data	12
		user_auth_data_len	12
		USER_SHARES.....	114, 117
		username	11
		Using LSF on Windows NT.....	287
		USRCMD	227
		USUSP	26, 35
		USUSP batch job state	22

Index

ut
 CPU utilization.....111
ut load index111

V

vendor daemon.....73
vertical configuration section53

W

WINDIR environment variable128

X

xbsub.....17
xlsadmin99
xlsmon47