Platform LSF Configuration Reference

Platform™

Contents

Part III: Environment Variables ... 597

Part IV: Troubleshooting ... 641

Features

Feature: Between-host user account mapping

The between-host user account mapping feature enables job submission and execution within a cluster that has different user accounts assigned to different hosts. Using this feature, you can map a local user account to a different user account on a remote host.

Contents

- About between-host user account mapping
- Scope
- Configuration to enable between-host user account mapping
- Between-host user account mapping behavior
- Configuration to modify between-host user account mapping behavior
- Between-host user account mapping commands

About between-host user account mapping

For clusters with different user accounts assigned to different hosts., between-host user account mapping allows you to submit a job from a local host and run the job as a different user on a remote host. There are two types of between-host user account mapping:

- Local user account mapping—for UNIX or Windows hosts, a user can map the local user account to a different user on a remote host
- Windows workgroup account mapping—allows LSF administrators to map all Windows workgroup users to a single Windows system account, eliminating the need to create multiple users and passwords in LSF. Users can submit and run jobs using their local user names and passwords, and LSF runs the jobs using the mapped system account name and password. With Windows workgroup account mapping, all users have the same permissions because all users map to the same Windows system account.
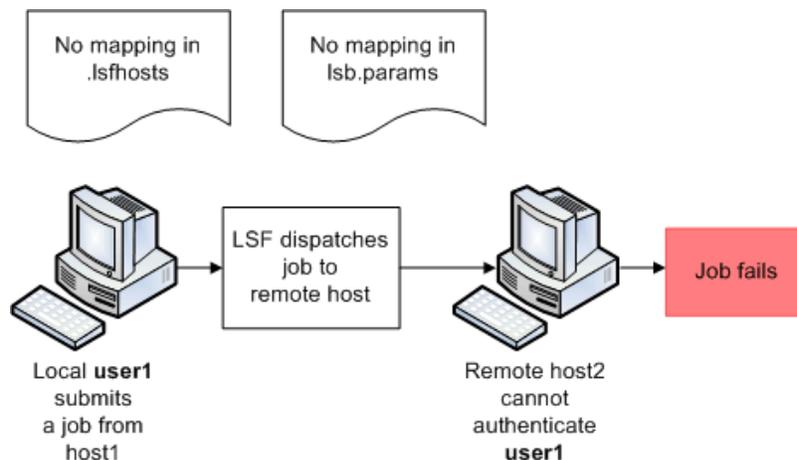
Figure 1: Default behavior (feature not enabled)
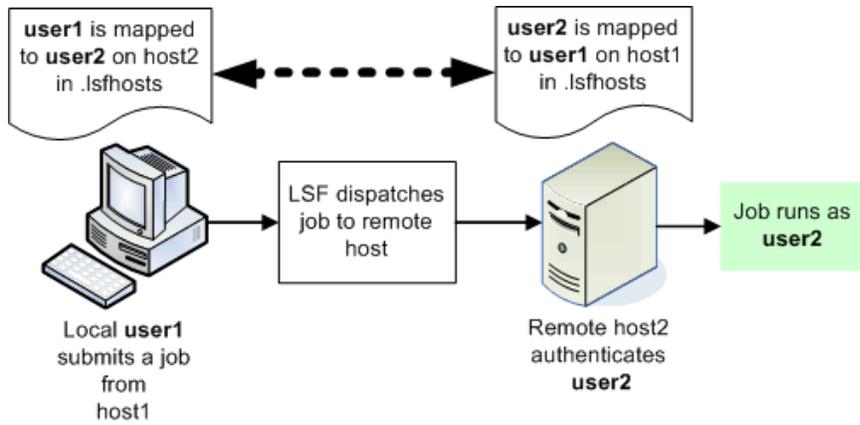
Figure 2: With local user account mapping enabled
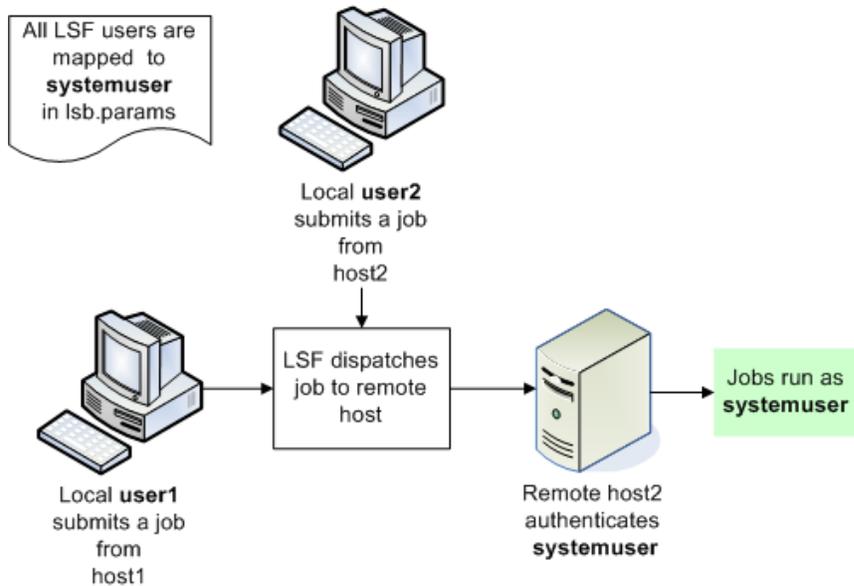
Figure 3: With Windows workgroup account mapping enabled

Scope

Applicability	Details
Operating system	
	• UNIX hosts • Windows hosts • A mix of UNIX and Windows hosts within a single clusters
Not required for	• A cluster with a uniform user name space • A mixed UNIX/Windows cluster in which user accounts have the same user name on both operating systems

Applicability	Details
Dependencies	• UNIX and Windows user accounts must be valid on all hosts in the cluster and must have the correct permissions to successfully run jobs. • For clusters that include both UNIX and Windows hosts, you must also enable the UNIX/Windows user account mapping feature.
Limitations	• For a MultiCluster environment that has different user accounts assigned to different hosts, you must also enable the cross-cluster user account mapping feature. Do not configure between-host user account mapping if you want to use system-level mapping in a MultiCluster environment; LSF ignores system-level mapping if mapping local user mapping is also defined in .lsfhosts. • For Windows workgroup account mapping in a Windows workgroup environment, all jobs run using the permissions associated with the specified system account.

Configuration to enable between-host user account mapping

Between-host user account mapping can be configured in one of the following ways:

- Users can map their local accounts at the user level in the file .lsfhosts. This file must reside in the user's home directory with owner read-write permissions for UNIX and owner read-write-execute permissions for Windows. It must not be readable and writable by any other user other than the owner. Save the .lsfhosts file without a file extension. Both the remote and local hosts must have corresponding mappings in their respective .lsfhosts files.
- LSF administrators can set up Windows workgroup account mapping at the system level in lsb.params.

Local user account mapping configuration

Local user account mapping is enabled by adding lines to the file .lsfhosts. Both the remote and local hosts must have corresponding mappings in their respective .lsfhosts files.

Configuration file	Syntax	Behavior
.lsfhosts	*host_name user_name* send	• Jobs sent from the local account run as *user_name* on *host_name*
	host_name user_name recv	• The local account can run jobs received from *user_name* submitted on *host_name*
	host_name user_name	• The local account can send jobs to and receive jobs from *user_name* on *host_name*
	+ +	• The local account can send jobs to and receive jobs from any user on any LSF host

Windows workgroup account mapping

Windows workgroup account mapping is enabled by defining the parameter SYSTEM_MAPPING_ACCOUNT in the file lsb.params.

Configuration file	Parameter and syntax	Default behavior
`lsb.params`	SYSTEM_MAPPING_ACCOUNT= *account*	• Enables Windows workgroup account mapping • Windows local user accounts run LSF jobs using the system account name and permissions

Between-host user account mapping behavior

Local user account mapping example

The following example describes how local user account mapping works when configured in the file `.lsfhosts` in the user's home directory. Only mappings configured in `.lsfhosts` on both the local and remote hosts work.

In the following example, the cluster contains hostA, hostB, and hostC. The account user1 is valid on all hosts except hostC, which requires a user account name of user99.

To allow …	On …	In the home directory of …	.lsfhosts must contain the line …
The account user1 to run jobs on all hosts within the cluster:			
• user1 to send jobs to user99 on hostC	hostA	user1	`hostC user99 send`
	hostB	user1	`hostC user99 send`
• user99 to receive jobs from user1 on either hostA or hostB	hostC	user99	`hostA user1 recv` `hostB user1 recv`

Windows workgroup account mapping example

The following example describes how Windows workgroup account mapping works when configured in the file `lsb.params`. In this example, the cluster has a Windows workgroup environment, and only the user account jobuser is valid on all hosts.

To allow …	In lsb.params, configure …	Behavior
All hosts within the cluster to run jobs on any other host within the cluster:		
• Map all local users to user account jobuser	`SYSTEM_MAPPING_ACCOUNT=jobuser`	When any local user submits an LSF job, the job runs under the account jobuser, using the permissions associated with the jobuser account.

Configuration to modify between-host user account mapping behavior

Not applicable: There are no parameters that modify the behavior of this feature.

Between-host user account mapping commands

Commands for submission

Command	Description
bsub	• Submits the job with the user name and password of the user who entered the command. The job runs on the execution host with the submission user name and password, unless you have configured between-host user account mapping. • With between-host user account mapping enabled, jobs that execute on a remote host run using the account name configured at the system level for Windows workgroups, or at the user level for local user account mapping.

Commands to monitor

Command	Description
bjobs -l	• Displays detailed information about jobs, including the user name of the user who submitted the job and the user name with which the job executed.
bhist -l	• Displays detailed historical information about jobs, including the user name of the user who submitted the job and the user name with which the job executed.

Commands to control

Not applicable.

Commands to display configuration

Command	Description
bparams	• Displays the value of SYSTEM_MAPPING_ACCOUNT defined in lsb.params.
badmin showconf	• Displays all configured parameters and their values set in lsf.conf or ego.conf that affect mbatchd and sbatchd. Use a text editor to view other parameters in the lsf.conf or ego.conf configuration files. • In a MultiCluster environment, badmin showconf only displays the parameters of daemons on the local cluster.

Use a text editor to view the file .lsfhosts.

Feature: Cross-cluster user account mapping

The cross-cluster user account mapping feature enables cross-cluster job submission and execution for a MultiCluster environment that has different user accounts assigned to different hosts. Using this feature, you can map user accounts in a local cluster to user accounts in one or more remote clusters.

Contents

- About cross-cluster user account mapping
- Scope
- Configuration to enable cross-cluster user account mapping
- Cross-cluster user account mapping behavior
- Configuration to modify cross-cluster user account mapping behavior
- Cross-cluster user account mapping commands

About cross-cluster user account mapping

For MultiCluster environments that have different user accounts assigned to different hosts, cross-cluster user account mapping allows you to submit a job from a local host and run the job as a different user on a remote host.
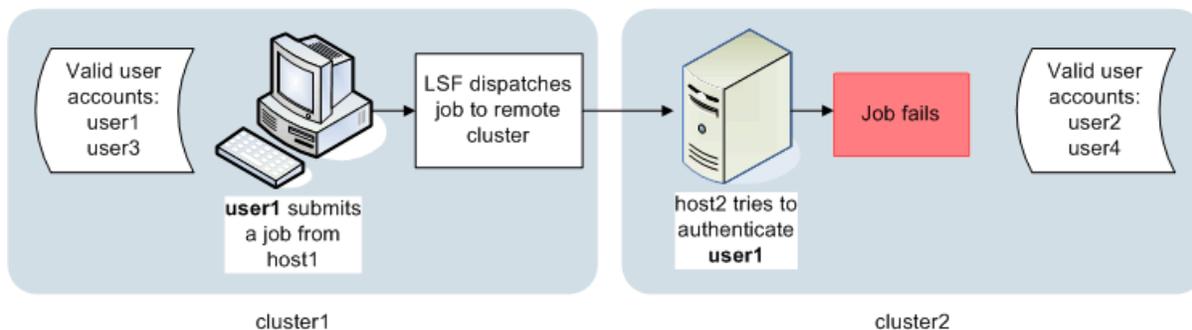
Figure 4: Default behavior (feature not enabled)
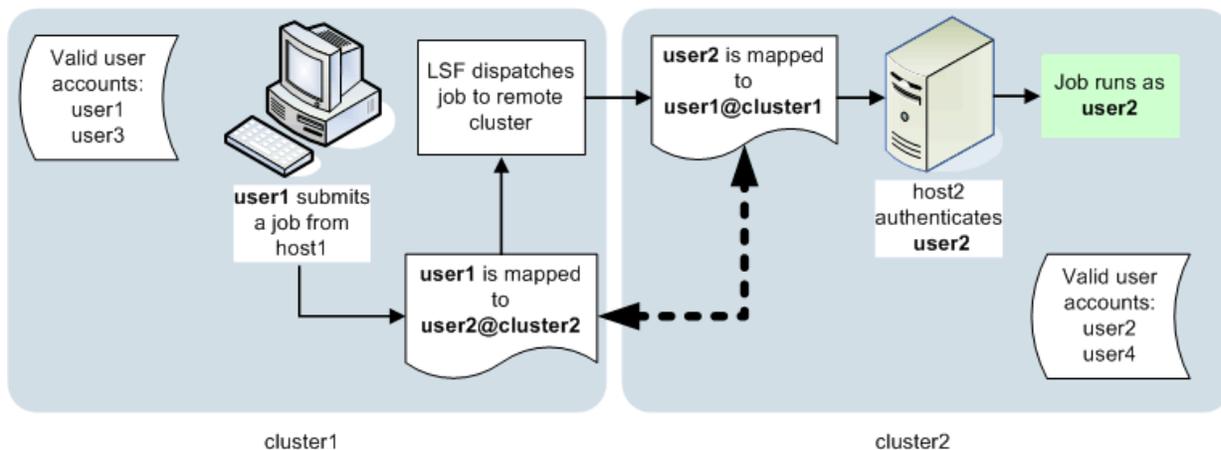
Figure 5: With cross-cluster user account mapping enabled

Scope

Applicability	Details
Operating system	
	• UNIX hosts
	• Windows hosts
	• A mix of UNIX and Windows hosts within one or more clusters
Not required for	• Multiple clusters with a uniform user name space
Dependencies	• UNIX and Windows user accounts must be valid on all hosts in the cluster and must have the correct permissions to successfully run jobs.
	• If users at your site have different user names on UNIX and Windows hosts within a single cluster, you must configure between-host user account mapping at the user level in .lsfhosts.
Limitations	• You cannot configure this feature at both the system-level and the user-level; LSF ignores system-level mapping if user-level mapping is also defined in .lsfhosts.
	• If one or more clusters include both UNIX and Windows hosts, you must also configure UNIX/Windows user account mapping.
	• If one or more clusters have different user accounts assigned to different hosts, you must also configure between-host user account mapping for those clusters, and then configure cross-cluster user account mapping at the system level only.

Configuration to enable cross-cluster user account mapping

- LSF administrators can map user accounts at the system level in the UserMap section of lsb.users. Both the remote and local clusters must have corresponding mappings in their respective lsb.users files.
- Users can map their local accounts at the user level in .lsfhosts. This file must reside in the user's home directory with owner read-write permissions for UNIX and owner read-write-execute permissions for Windows. Save the .lsfhosts file without a file extension. Both the remote and local hosts must have corresponding mappings in their respective .lsfhosts files.

Restriction:

Define *either* system-level or user-level mapping, but not both.
LSF ignores system-level mapping if user-level mapping is also defined in .lsfhosts.

Configuration file	Level	Syntax	Behavior
`lsb.users`	System	Required fields: LOCAL REMOTE DIRECTION	• Maps a user name on a local host to a different user name on a remote host • Jobs that execute on a remote host run using a mapped user name rather than the job submission user name
`.lsfhosts`	User	*host_name user_name* send	• Jobs sent from the local account run as *user_name* on *host_name*
		host_name user_name recv	• The local account can run jobs received from *user_name* submitted on *host_name*
		host_name user_name	• The local account can send jobs to and receive jobs from *user_name* on *host_name*
		cluster_name user_name	• The local account can send jobs to and receive jobs from *user_name* on any host in the cluster *cluster_name*
		+ +	• The local account can send jobs to and receive jobs from any user on any LSF host

Cross-cluster user account mapping behavior

System-level configuration example

The following example illustrates LSF behavior when the LSF administrator sets up cross-cluster user account mapping at the system level. This example shows the UserMap section of the file `lsb.users` on both the local and remote clusters.

On cluster1:

```
Begin UserMap
LOCAL     REMOTE                          DIRECTION
user1     user2@cluster2                  export
user3     user6@cluster2                  export
End UserMap
```

On cluster2:

```
Begin UserMap
LOCAL     REMOTE                          DIRECTION
user2     user1@cluster1                  import
user6     user3@cluster1                  import
End UserMap
```

The mappings between users on different clusters are as follows:
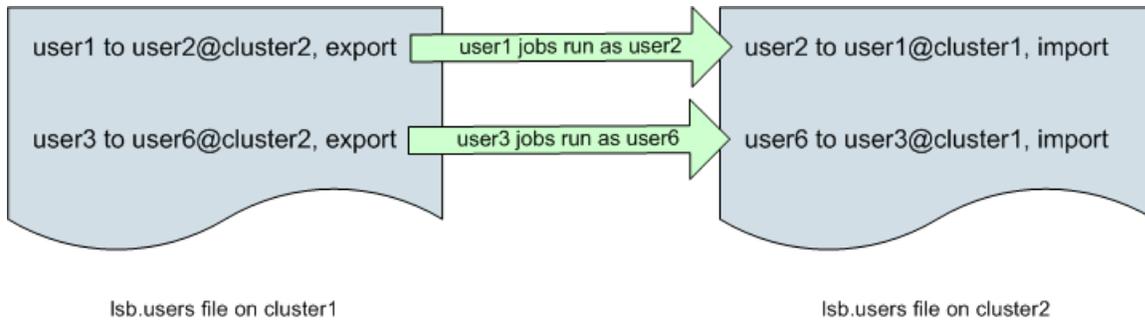
Figure 6: System-level mappings for both clusters

Only mappings configured in lsb.users on both clusters work. In this example, the common user account mappings are:

- user1@cluster1 to user2@cluster2
- user3@cluster1 to user6@cluster2

User-level configuration examples

The following examples describe how user account mapping works when configured at the user level in the file .lsfhosts in the user's home directory. Only mappings configured in .lsfhosts on hosts in both clusters work.

To allow …	On …	In the home directory of …	.lsfhosts must contain the line …
The accounts user1 and user2 to run jobs on all hosts in both clusters:			
• user1 to send jobs to and receive jobs from user2 on cluster2	All hosts in cluster1	user1	`cluster2 user2`
• user2 to send jobs to and receive jobs from user1 on cluster1	All hosts in cluster2	user2	`cluster1 user1`
The account user1 to run jobs on cluster2 using the lsfguest account:			
• user1 to send jobs as lsfguest to all hosts in cluster2	All hosts in cluster1	user1	`cluster2 lsfguest send`
• lsfguest to receive jobs from user1 on cluster1	All hosts in cluster2	lsfguest	`cluster1 user1 recv`

Configuration to modify cross-cluster user account mapping behavior

Not applicable: There are no parameters that modify the behavior of this feature.

Cross-cluster user account mapping commands

Commands for submission

Command	Description
bsub	• Submits the job with the user name and password of the user who entered the command. The job runs on the execution host with the submission user name and password, unless you have configured cross-cluster user account mapping. • With cross-cluster user account mapping enabled, jobs that execute on a remote host run using the account name configured at the system or user level.

Commands to monitor

Command	Description
bjobs -l	• Displays detailed information about jobs, including the user name of the user who submitted the job and the user name with which the job executed.
bhist -l	• Displays detailed historical information about jobs, including the user name of the user who submitted the job and the user name with which the job executed.

Commands to control

Not applicable. There are no commands to control the behavior of this feature.

Commands to display configuration

Not applicable. Use a text editor to view .lsfhosts or to view the UserMap section of lsb.users.

Feature: SSH

Secure Shell or SSH is a network protocol that provides confidentiality and integrity of data using a secure channel between two networked devices. You can enable and use SSH to secure communication between hosts and during job submission.

About SSH

SSH uses public-key cryptography to authenticate the remote computer and allow the remote computer to authenticate the user, if necessary.

SSH is typically used to log into a remote machine and execute commands, but it also supports tunneling, forwarding arbitrary TCP ports and X11 connections. SSH uses a client-server protocol.

SSH uses private/public key pairs to log into another host. Users no longer have to supply a password every time they log on to a remote host.

SSH is used when running any of the following:

- Remote log on to a lightly loaded host (lslogin)
- An interactive job (bsub -IS | -ISp | ISs)
- An interactive X-window job (bsub -IX)
- An externally submitted job that is interactive or X-window (esub)

Scope

Applicability	Details
Dependencies	• OpenSSH is fully supported. • You must have SSH correctly installed on all machines requiring it. • You must generate public/private key pairs and add the content of the public key to the authorized_keys file on remote hosts. For more information, refer to your SSH documentation. • For X-window jobs, you must set the DISPLAY environment variable to *X-serverHost*:0.0, where *X-serverHost* is the name of the X-window server. Ensure the X-server can access itself. Run, for example, xhost +localhost.
Operating system	Any that also support OpenSSH.
Limitations	Private user keys must have no password set.

Configuration to enable SSH

Configuration file	Level	Syntax	Behavior
lsf.conf	System	LSF_LSLOGIN_SSH=Y | y	A user with SSH configured can log on to a remote host without providing a password. All communication between local and remote hosts is encrypted.

Configuration to modify SSH

Not applicable.

SSH commands

Commands to monitor

Command	Behavior
netstat -an	Displays all active TCP connections and the TCP and UDP ports on which the computer is listening.

Troubleshoot SSH

Use the SSH command on the job execution host to connect it securely with the job submission host.

If the host fails to connect, you can perform the following steps to troubleshoot.

1. Check the SSH version on both hosts.

   If the hosts have different SSH versions, a message displays identifying a protocol version mismatch.

2. Check that public and private key pairs are correctly configured.

   More information on configuring key pairs is here: *http://sial.org/howto/openssh/ publickey-auth/*.

3. Check the domain name.

   $ ssh –f –L 6000:localhost:6000 *domain_name.example.com* date

   $ ssh –f –L 6000:localhost:6000 *domain_name* date

   If these commands return errors, troubleshoot the domain name with the error information returned.

The excecution host should connect without password and pass phrases.

```
$ ssh sahpia03
$ ssh sahpia03.example.com
```

Feature: External authentication

The external authentication feature provides a framework that enables you to integrate LSF with any third-party authentication product—such as Kerberos or DCE Security Services—to authenticate users, hosts, and daemons. This feature provides a secure transfer of data within the authentication data stream between LSF clients and servers. Using external authentication, you can customize LSF to meet the security requirements of your site.

Contents

- About external authentication (eauth)
- Scope
- Configuration to enable external authentication
- External authentication behavior
- Configuration to modify external authentication
- External authentication commands

About external authentication (eauth)

The external authentication feature uses an executable file called eauth. You can write an eauth executable that authenticates users, hosts, and daemons using a site-specific authentication method such as Kerberos or DCE Security Services client authentication. You can also specify an external encryption key (recommended) and the user account under which eauth runs.

Important:

LSF uses an internal encryption key by default. To increase security, configure an external encryption key by defining the parameter LSF_EAUTH_KEY in

lsf.sudoers.

During LSF installation, a default eauth executable is installed in the directory specified by the parameter LSF_SERVERDIR in lsf.conf. The default executable provides an example of how the eauth protocol works. You should write your own eauth executable to meet the security requirements of your cluster.

Figure 7: Default behavior (eauth executable provided with LSF)

The eauth executable uses corresponding processes eauth -c *host_name* (client) and eauth -s (server) to provide a secure data exchange between LSF daemons on client and server hosts. The variable *host_name* refers to the host on which eauth -s runs; that is, the host called by the command. For bsub, for example, the *host_name* is NULL, which means the authentication data works for any host in the cluster.
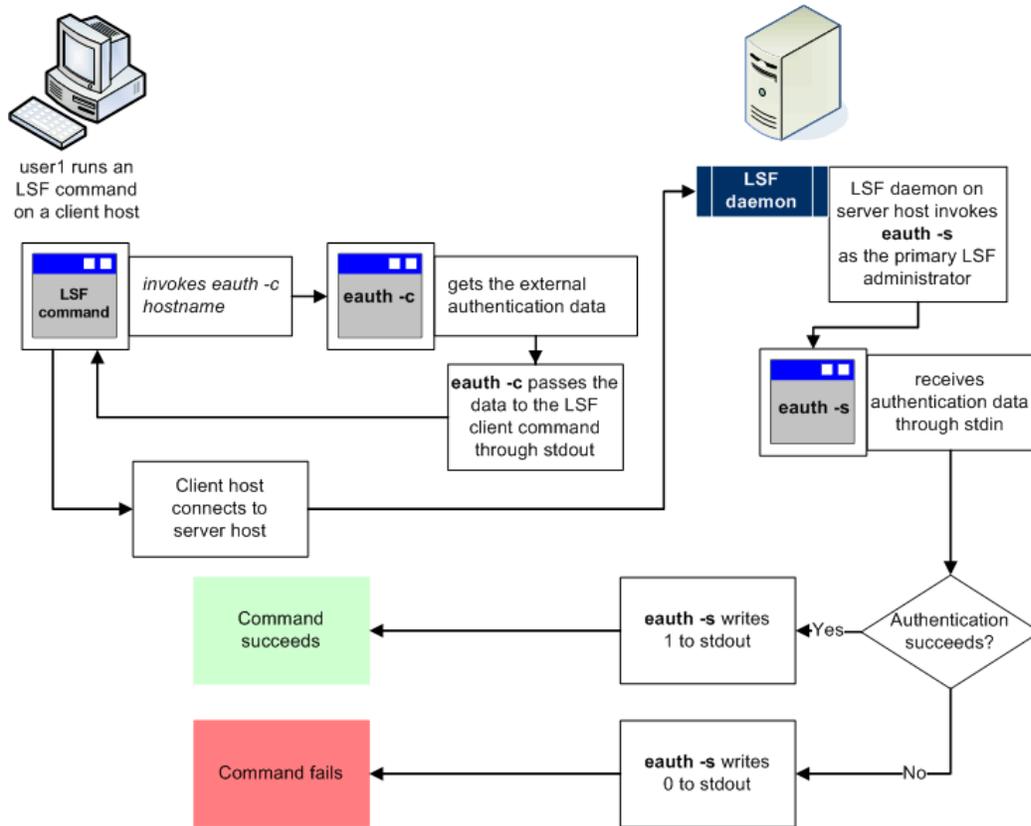
Figure 8: How eauth works

One eauth -s process can handle multiple authentication requests. If eauth -s terminates, the LSF daemon invokes another instance of eauth -s to handle new authentication requests.

The standard input stream to eauth -s is a text string with the following format:

```
uid gid user_name client_addr client_port user_auth_data_len eauth_client eauth_server
aux_data_file aux_data_status user_auth_data
```

where

The variable …	Represents the …
uid	User ID of the client user
gid	Group ID of the client user
user_name	User name of the client user
client_addr	IP address of the client host
client_port	Port number from which the client request originates
user_auth_data_len	Length of the external authentication data passed from the client host
eauth_client	Daemon or user that invokes eauth -c
eauth_server	Daemon that invokes eauth -s

The variable …	Represents the …
aux_data_file	Location of the temporary file that stores encrypted authentication data
aux_data_status	File in which `eauth -s` stores authentication status. When used with Kerberos authentication, `eauth -s` writes the source of authentication to this file if authentication fails. For example, if mbatchd to mbatchd authentication fails, `eauth -s` writes "mbatchd" to the file defined by *aux_data_status*. If user to mbatchd authentication fails, `eauth -s` writes "user" to the *aux_data_status* file.
user_auth_data	External authentication data passed from the client host

The variables required for the `eauth` executable depend on how you implement external authentication at your site. For `eauth` parsing, unused variables are marked by "'".

User credentials

When an LSF user submits a job or issues a command, the LSF daemon that receives the request verifies the identity of the user by checking the user credentials. External authentication provides the greatest security of all LSF authentication methods because the user credentials are obtained from an external source, such as a database, and then encrypted prior to transmission. For Windows hosts, external authentication is the only truly secure type of LSF authentication.

Host credentials

LSF first authenticates users and then checks host credentials. LSF accepts requests sent from all hosts configured as part of the LSF cluster, including floating clients and any hosts that are dynamically added to the cluster. LSF rejects requests sent from a non-LSF host. If your cluster requires additional host authentication, you can write an `eauth` executable that verifies both user and host credentials.

Daemon credentials

Daemon authentication provides a secure channel for passing credentials between hosts, mediated by the master host. The master host mediates authentication by means of the `eauth` executable, which ensures secure passing of credentials between submission hosts and execution hosts, even though the submission host does not know which execution host will be selected to run a job.

Daemon authentication applies to the following communications between LSF daemons:

* mbatchd requests to sbatchd
* sbatchd updates to mbatchd
* PAM interactions with res
* mbatchd to mbatchd (in a MultiCluster environment)

Kerberos authentication

Kerberos authentication is an extension of external daemon authentication, providing authentication of LSF users and daemons during client-server interactions. The `eauth` executable provided with the Platform integration package uses Kerberos Version 5 APIs for interactions between mbatchd and sbatchd, and between pam and res. When you use Kerberos authentication for a cluster or MultiCluster, authentication data is encrypted along the entire path from job submission through to job completion.

You can also use Kerberos authentication for delegation of rights (forwarding credentials) when a job requires a Kerberos ticket during job execution. LSF ensures that a ticket-granting ticket (TGT) can be forwarded securely to the execution host. LSF also automatically renews Kerberos credentials by means of daemon wrapper scripts.

Scope

Applicability	Details
Operating system	- UNIX - Windows (except for Kerberos authentication)
Allows for	- Authentication of LSF users, hosts, and daemons - Authentication of any number of LSF users
Not required for	Authorization of users based on account permissions
Dependencies	- UNIX and Windows user accounts must be valid on all hosts in the cluster, or the correct type of account mapping must be enabled: - For a mixed UNIX/Windows cluster, UNIX/Windows user account mapping must be enabled - For a cluster with a non-uniform user name space, between-host account mapping must be enabled - For a MultiCluster environment with a non-uniform user name space, cross-cluster user account mapping must be enabled - User accounts must have the correct permissions to successfully run jobs. - The owner of lsf.sudoers on Windows must be Administrators.

Configuration to enable external authentication

During LSF installation:

- The parameter LSF_AUTH in lsf.conf is set to eauth, which enables external authentication
- A default eauth executable is installed in the directory specified by the parameter LSF_SERVERDIR in lsf.conf

The default executable provides an example of how the eauth protocol works. You should write your own eauth executable to meet the security requirements of your cluster.

Configuration file	Parameter and syntax	Default behavior
`lsf.conf`	LSF_AUTH=eauth	• Enables external authentication
	LSF_AUTH_DAEMONS=y | Y	• Enables daemon authentication when external authentication is enabled
		Note: By default, daemon authentication is not enabled. If you enable daemon authentication and want to turn it off later, you must comment out or delete the parameter LSF_AUTH_DAEMONS.

External authentication behavior

The following example illustrates how a customized eauth executable can provide external authentication of users, hosts, and daemons. In this example, the eauth executable has been customized so that corresponding instances of eauth -c and eauth -s obtain user, host, and daemon credentials from a file that serves as the external security system. The eauth executable can also be customized to obtain credentials from an operating system or from an authentication protocol such as Kerberos.
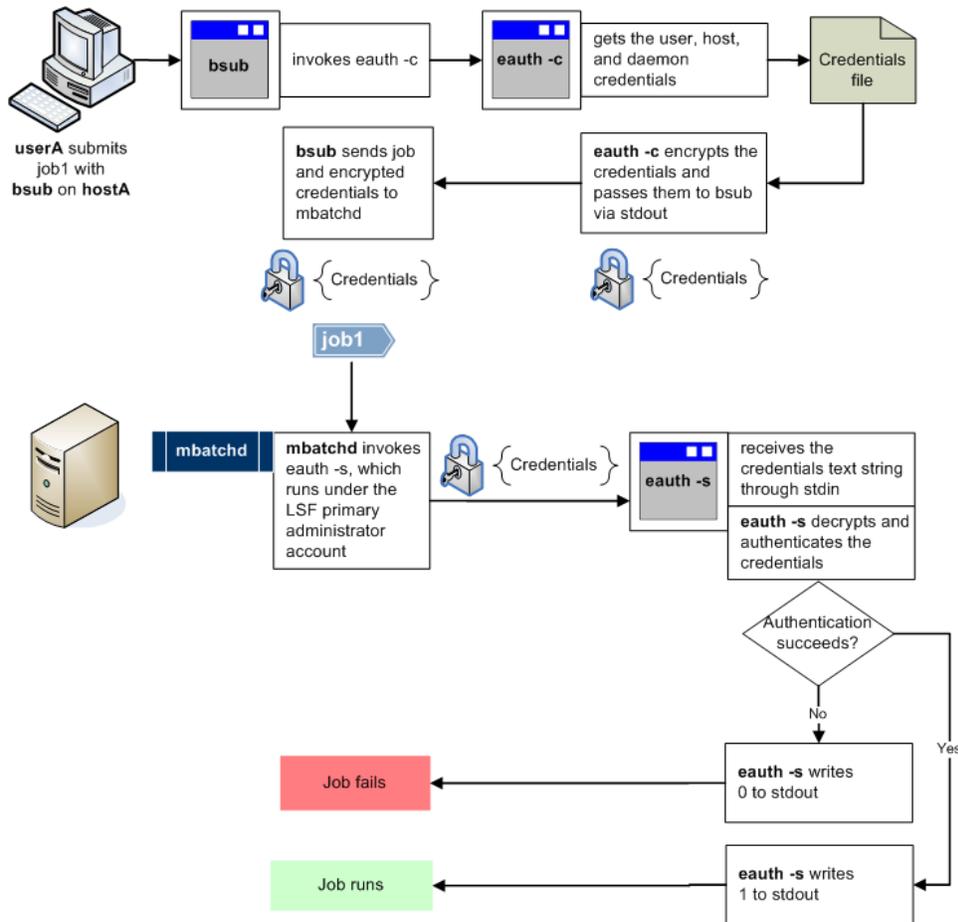
Figure 9: Example of external authentication

Authentication failure

When external authentication is enabled, the message

User permission denied

indicates that the eauth executable failed to authenticate the user's credentials.

Security

External authentication—and any other LSF authentication method—depends on the security of the root account on all hosts within the cluster. Limit access to the root account to prevent unauthorized use of your cluster.

Configuration to modify external authentication

You can modify external authentication behavior by writing your own eauth executable. There are also configuration parameters that modify various aspects of external authentication behavior by:

- Increasing security through the use of an external encryption key (recommended)
- Specifying a trusted user account under which the eauth executable runs (UNIX and Linux only)

You can also choose Kerberos authentication to provide a secure data exchange during LSF user and daemon authentication and to forward credentials to a remote host for use during job execution.

Configuration to modify security

File	Parameter and syntax	Descriptions
lsf.sudoers	LSF_EAUTH_KEY=*key*	• The eauth executable uses the external encryption key that you define to encrypt and decrypt the credentials. • The key must contain at least six characters and must use only printable characters. • For UNIX, you must edit the lsf.sudoers file on all hosts within the cluster and specify the same encryption key. You must also configure eauth as setuid to root so that eauth can read the lsf.sudoers file and obtain the value of LSF_EAUTH_KEY. • For Windows, you must edit the shared lsf.sudoers file.

Configuration to specify the eauth user account

On UNIX hosts, the eauth executable runs under the account of the primary LSF administrator. You can modify this behavior by specifying a different trusted user account. For Windows hosts, you do not need to modify the default behavior because eauth runs under the service account, which is always a trusted, secure account.

File	Parameter and syntax	Description
lsf.sudoers	LSF_EAUTH_USER=*user_name*	• UNIX only • The eauth executable runs under the account of the specified user rather than the account of the LSF primary administrator • You must edit the lsf.sudoers file on all hosts within the cluster and specify the same user name

Configuration to enable Kerberos authentication

To install and configure Kerberos authentication, refer to the information included with your Kerberos integration package provided by Platform Computing Inc..

Restriction:

Kerberos authentication is supported only for UNIX and Linux hosts, and only on the following operating systems:

- AIX 4
- Alpha 4.*x*
- IRIX 6.5
- Linux 2.*x*

- Solaris 2.*x*

Configuration file	Parameter and syntax	Behavior
lsf.conf	LSF_AUTH=eauth	• Enables external authentication
	LSF_AUTH_DAEMONS=y | Y	• Enables daemon authentication when external authentication is enabled
	LSF_DAEMON_WRAP=y | Y	• Required for Kerberos authentication • mbatchd, sbatchd, and RES run the executable LSF_SERVERDIR/daemons.wrap
lsf.sudoers	LSF_EAUTH_USER=root	• for Kerberos authentication, the eauth executable must run under the root account • You must edit the lsf.sudoers file on all hosts within the cluster and specify the same user name
	LSF_LOAD_PLUGINS=y | Y	• Required for Kerberos authentication when plug-ins are used instead of the daemon wrapper script • LSF loads plug-ins from the directory LSB_LIBDIR
	LSF_EEXEC_USER=root	• Required for Kerberos authentication. The parameter LSF_DAEMON_WRAP must also be set to y or Y. • The eexec executable provided with the Kerberos integration runs under the root account

External authentication commands

Commands for submission

Command	Description
All LSF commands	• If the parameter LSF_AUTH=eauth in the file lsf.conf, LSF daemons authenticate users and hosts—as configured in the eauth executable—before executing an LSF command • If external authentication is enabled and the parameter LSF_AUTH_DAEMONS=Y in the file lsf.conf, LSF daemons authenticate each other as configured in the eauth executable

Commands to monitor

Not applicable: There are no commands to monitor the behavior of this feature.

Commands to control

Not applicable: There are no commands to control the behavior of this feature.

Commands to display configuration

Command	Description
`badmin showconf`	• Displays all configured parameters and their values set in `lsf.conf` or `ego.conf` that affect `mbatchd` and `sbatchd`. Use a text editor to view other parameters in the `lsf.conf` or `ego.conf` configuration files. • In a MultiCluster environment, `badmin showconf` only displays the parameters of daemons on the local cluster.

Use a text editor to view the `lsf.sudoers` configuration file.

Feature: LSF daemon startup control

The LSF daemon startup control feature allows you to specify a list of user accounts other than root that can start LSF daemons on UNIX hosts. This feature also enables UNIX and Windows users to bypass the additional login required to start res and sbatchd when the EGO Service Controller (EGOSC) is configured to control LSF daemons; bypassing the EGO administrator login enables the use of scripts to automate system startup.

Contents

- About LSF daemon startup control
- Scope
- Configuration to enable LSF daemon startup control
- LSF daemon startup control behavior
- Configuration to modify LSF daemon startup control
- LSF daemon startup control commands

About LSF daemon startup control

Startup by users other than root (UNIX only)

On UNIX hosts, by default only root can manually start LSF daemons. To manually start LSF daemons, a user runs the commands lsadmin and badmin, which have been installed as setuid root. The LSF daemon startup control feature allows you to specify a list of user accounts that are allowed to run the commands lsadmin and badmin to start LSF daemons. The list is defined in the file lsf.sudoers.

On Windows hosts, the Platform services admin group identifies the user accounts that can start and shut down LSF daemons.
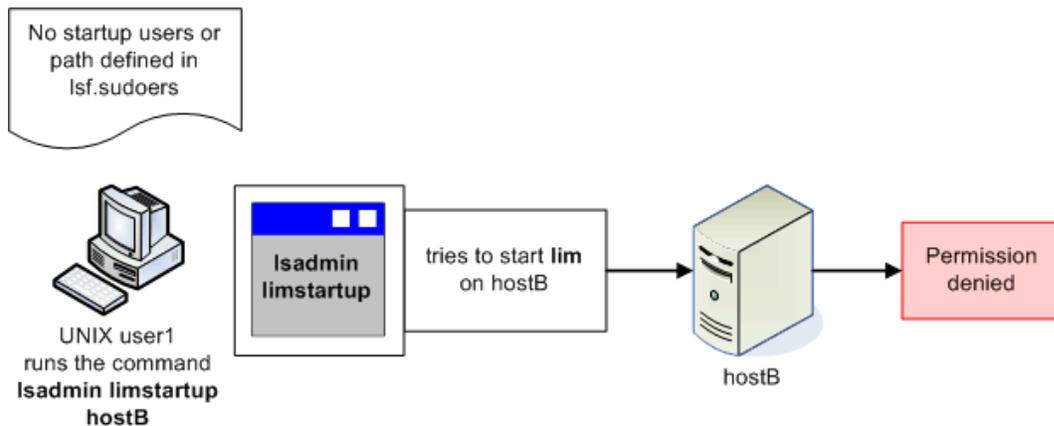
Figure 10: Default behavior (feature not enabled)
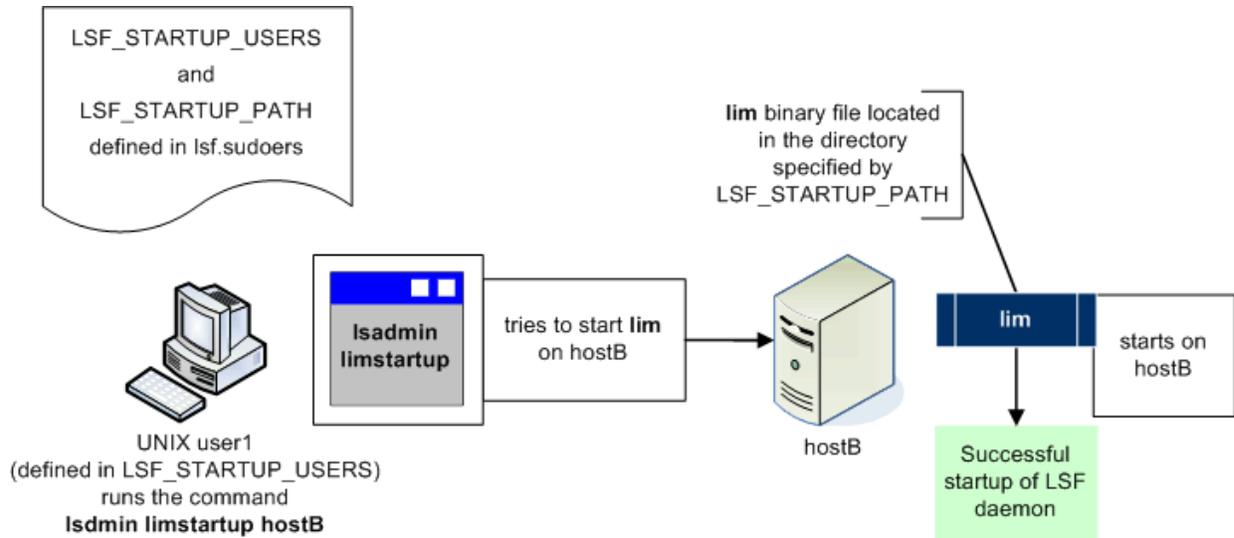
Figure 11: With LSF daemon startup control enabled

EGO administrator login bypass

If the EGO Service Controller (EGOSC) is configured to control LSF daemons, EGO will automatically restart the res and sbatchd daemons unless a user has manually shut them down. When manually starting a res or sbatchd daemon that EGO has not yet started, the user who invokes lsadmin or badmin is prompted to enter EGO administrator credentials. You can configure LSF to bypass this step by specifying the EGO administrator credentials in the file lsf.sudoers.

In the following illustrations, an authorized user is either a UNIX user listed in the LSF_STARTUP_USERS parameter or a Windows user with membership in the Platform services admin group.
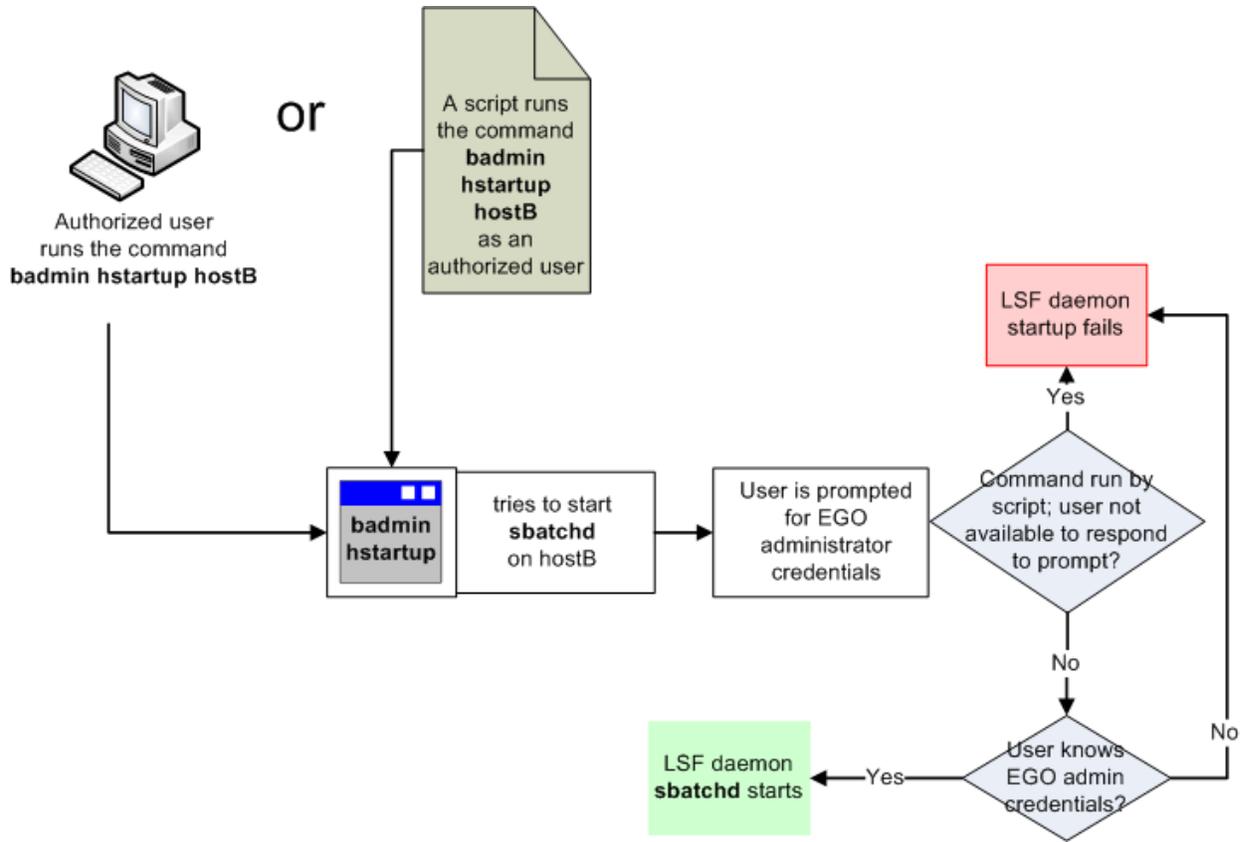
Figure 12: EGO administrator login bypass not enabled
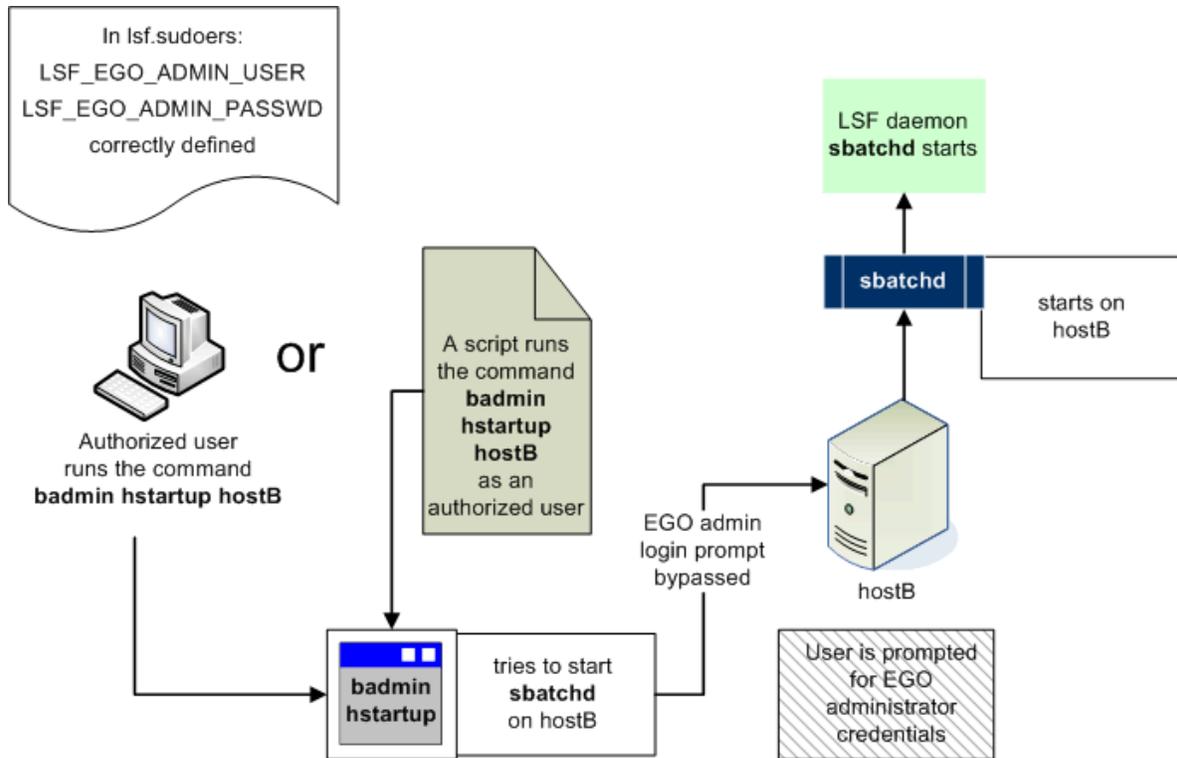
Figure 13: With EGO administrator login bypass enabled

Scope

Applicability	Details
Operating system	• UNIX hosts only within a UNIX-only or mixed UNIX/Windows cluster: Startup of LSF daemons by users other than root. • UNIX and Windows: EGO administrator login bypass.

Dependencies	• For startup of LSF daemons by users other than root:
	    • You must define both a list of users *and* the absolute path of the directory that contains the LSF daemon binary files.
	    • The commands lsadmin and badmin must be installed as setuid root.
	• For EGO administrator login bypass, the default Admin EGO cluster administrator account must be defined.

Limitations	• Startup of LSF daemons by users other than root applies only to the following lsadmin and badmin subcommands:
	    • badmin hstartup
	    • lsadmin limstartup
	    • lsadmin resstartup

Configuration to enable LSF daemon startup control

Startup by users other than root (UNIX-only)

The LSF daemon startup control feature is enabled for UNIX hosts by defining the LSF_STARTUP_USERS and LSF_STARTUP_PATH parameters in the lsf.sudoers file. Permissions for lsf.sudoers must be set to 600. For Windows hosts, this feature is already enabled at installation when the Platform services admin group is defined.

Configuration file	Parameter and syntax	Default behavior
lsf.sudoers	LSF_STARTUP_USERS=all_admins	• Enables LSF daemon startup by users other than root when LSF_STARTUP_PATH is also defined. • Allows all UNIX users defined as LSF administrators in the file lsf.cluster.*cluster_name* to start LSF daemons as root by running the lsadmin and badmin commands. • Not recommended due to the security risk of a non-root LSF administrator adding to the list of administrators in the lsf.cluster.*cluster_name* file. • Not required for Windows hosts because all users with membership in the Platform services admin group can start LSF daemons.
	LSF_STARTUP_USERS="*user_name1 user_name2 …*" LSF_STARTUP_USERS=*user_name*	• Enables LSF daemon startup by users other than root when LSF_STARTUP_PATH is also defined. • Allows the specified user accounts to start LSF daemons as root by running the lsadmin and badmin commands. • Specify only cluster administrator accounts; if you add a non-administrative user, the user can start —but not shut down—LSF daemons. • Separate multiple user names with a space. • For a single user, do not use quotation marks.

Configuration file	Parameter and syntax	Default behavior
	LSF_STARTUP_PATH=*path*	• Enables LSF daemon startup by users other than root when LSF_STARTUP_USERS is also defined. • Specifies the directory that contains the LSF daemon binary files. • LSF daemons are usually installed in the path specified by the LSF_SERVERDIR parameter defined in the cshrc.lsf, profile.lsf, or lsf.conf files. **Important:** For security reasons, you should move the LSF daemon binary files to a directory other than LSF_SERVERDIR or LSF_BINDIR. The user accounts specified by LSF_STARTUP_USERS can start any binary in the LSF_STARTUP_PATH.

EGO administrator login bypass

For both UNIX and Windows hosts, you can bypass the EGO administrator login for res and sbatchd by defining the parameters LSF_EGO_ADMIN_USER and LSF_EGO_ADMIN_PASSWORD in the lsf.sudoers file.

Configuration file	Parameter and syntax	Default behavior
lsf.sudoers	LSF_EGO_ADMIN_USER=Admin	• Enables a user or script to bypass the EGO administrator login prompt when LSF_EGO_ADMIN_PASSWD is also defined. • Applies only to startup of res or sbatchd. • Specify the Admin EGO cluster administrator account.
	LSF_EGO_ADMIN_PASSWD=*password*	• Enables a user or script to bypass the EGO administrator login prompt when LSF_EGO_ADMIN_USER is also defined. • Applies only to startup of res or sbatchd. • Specify the password for the Admin EGO cluster administrator account.

LSF daemon startup control behavior

This example illustrates how LSF daemon startup control works when configured for UNIX hosts in a cluster with the following characteristics:

- The cluster contains both UNIX and Windows hosts
- The UNIX account user1 is mapped to the Windows account BUSINESS\user1 by enabling the UNIX/Windows user account mapping feature
- The account BUSINESS\user1 is a member of the Platform services admin group
- In the file lsf.sudoers:

```
LSF_STARTUP_USERS="user1 user2 user3"
LSF_STARTUP_PATH=LSF_TOP/7.0/linux2.4-glibc2.3-x86/etc
LSF_EGO_ADMIN_USER=Admin
LSF_EGO_ADMIN_PASSWD=Admin
```

Note:

You should change the Admin user password immediately after installation using the command egosh user modify.
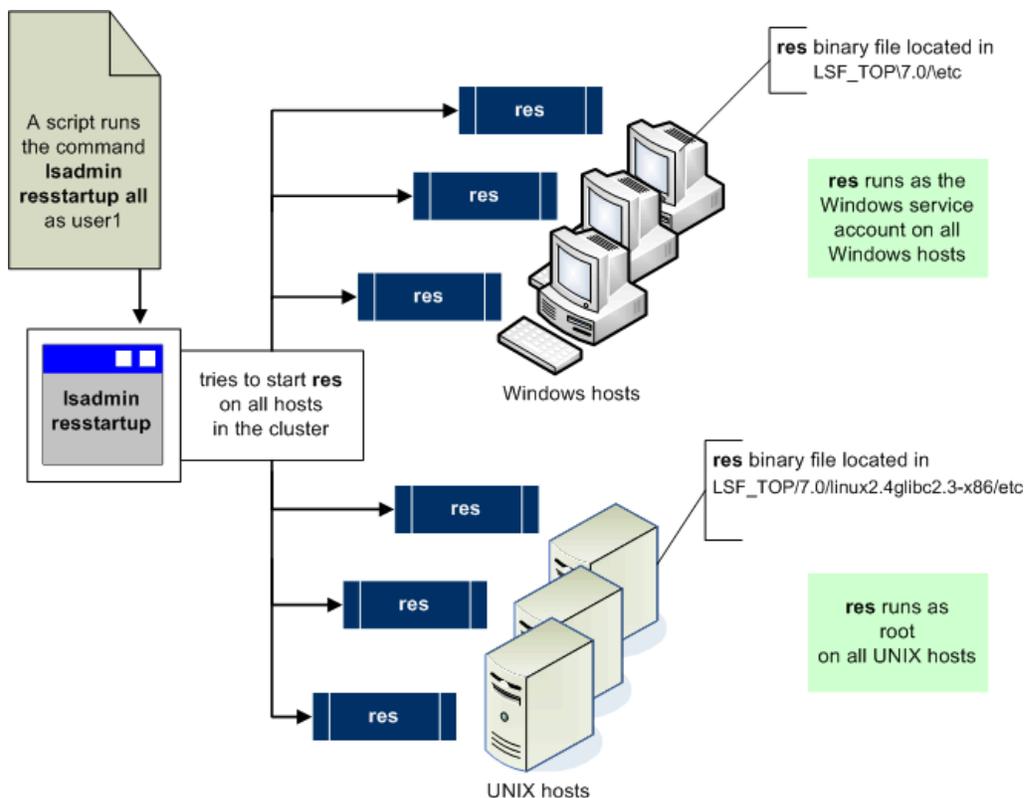
Figure 14: Example of LSF daemon startup control

Configuration to modify LSF daemon startup control

Not applicable: There are no parameters that modify the behavior of this feature.

LSF daemon startup control commands

Commands for submission

Command	Description
N/A	• This feature does not directly relate to job submission.

Commands to monitor

Command	Description
bhosts	• Displays the host status of all hosts, specific hosts, or specific host groups.
lsload	• Displays host status and load information.

Commands to control

Command	Description
`badmin hstartup`	• Starts the sbatchd daemon on specific hosts or all hosts. Only root and users listed in the LSF_STARTUP_USERS parameter can successfully run this command.
`lsadmin limstartup`	• Starts the lim daemon on specific hosts or all hosts in the cluster. Only root and users listed in the LSF_STARTUP_USERS parameter can successfully run this command.
`lsadmin resstartup`	• Starts the res daemon on specific hosts or all hosts in the cluster. Only root and users listed in the LSF_STARTUP_USERS parameter can successfully run this command.

Commands to display configuration

Command	Description
`badmin showconf`	• Displays all configured parameters and their values set in `lsf.conf` or `ego.conf` that affect `mbatchd` and `sbatchd`. Use a text editor to view other parameters in the `lsf.conf` or `ego.conf` configuration files. • In a MultiCluster environment, `badmin showconf` only displays the parameters of daemons on the local cluster.

Use a text editor to view the `lsf.sudoers` configuration file.

Feature: Pre-execution and post-execution processing

The pre- and post-execution processing feature provides a way to run commands on the execution host prior to and after completion of LSF jobs. Use pre-execution commands to set up an execution host with the required directories, files, software licenses, environment, and user permissions. Use post-execution commands to define post-job processing such as cleaning up job files or transferring job output.

Contents

- About pre- and post-execution processing
- Scope
- Configuration to enable pre- and post-execution processing
- Pre- and post-execution processing behavior
- Configuration to modify pre- and post-execution processing
- Pre- and post-execution processing commands

About pre- and post-execution processing

You can use the pre- and post-execution processing feature to run commands before a batch job starts or after it finishes. Typical uses of this feature include the following:

- Reserving resources such as tape drives and other devices not directly configurable in LSF
- Making job-starting decisions in addition to those directly supported by LSF
- Creating and deleting scratch directories for a job
- Customizing scheduling based on the exit code of a pre-execution command
- Checking availability of software licenses
- Assigning jobs to run on specific processors on SMP machines
- Transferring data files needed for job execution
- Modifying system configuration files before and after job execution
- Using a post-execution command to clean up a state left by the pre-execution command or the job

Pre-execution and post-execution commands can be defined at the queue, application, and job levels.

The command path can contain up to 4094 characters for UNIX and Linux, or up to 255 characters for Windows, including the directory, file name, and expanded values for %J (*job_ID*) and %I (*index_ID*).

When JOB_INCLUDE_POSTPROC is defined in an application profile, a job is considered in RUN state while the job is in post exec stage (which is DONE state for regular jobs). When the job is also resizable, job grow requests are ignored. However job shrink requests can be processed. For either case, LSF does not invoke the job resized notification command.

Default behavior (feature not enabled)
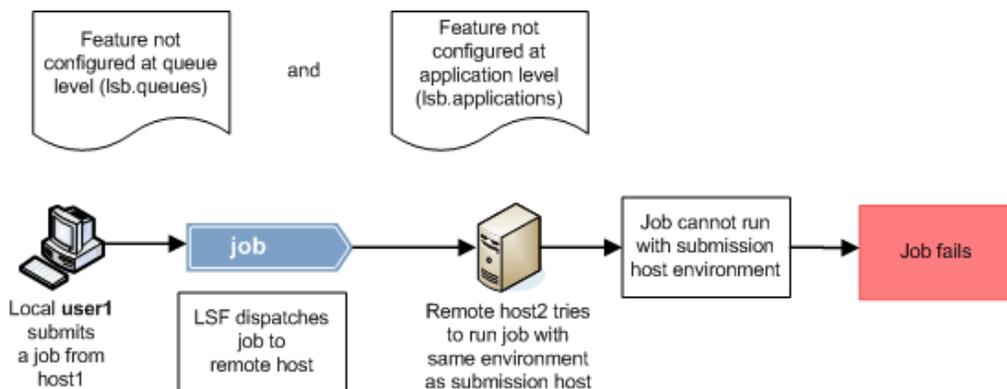
With pre- and post-execution processing enabled at the queue or application level

The following example illustrates how pre- and post-execution processing works for setting the environment prior to job execution and for transferring resulting files after the job runs.
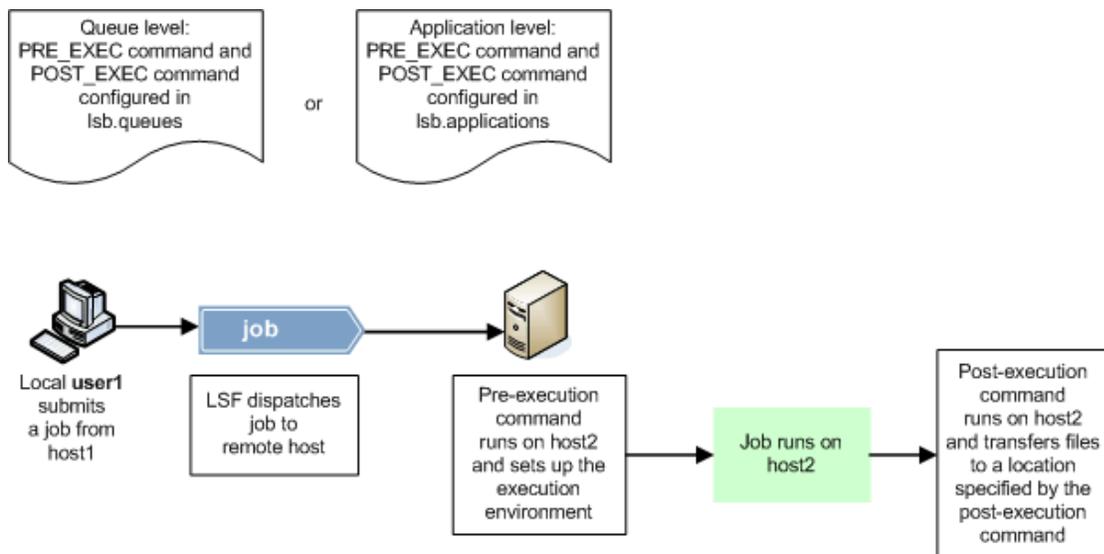
Any executable command line can serve as a pre-execution or post-execution command. By default, the commands run under the same user account, environment, home directory, and working directory as the job. For parallel jobs, the commands run on the first execution host.

Scope

Applicability	Details
Operating system	• UNIX • Windows • A mix of UNIX and Windows hosts

Applicability	Details
Dependencies	• UNIX and Windows user accounts must be valid on all hosts in the cluster and must have the correct permissions to successfully run jobs. • On a Windows Server 2003, x64 Edition platform, users must have read and execute privileges for cmd.exe.
Limitations	• Applies to batch jobs only (jobs submitted using the bsub command)

Configuration to enable pre- and post-execution processing

The pre- and post-execution processing feature is enabled by defining at least one of the parameters PRE_EXEC or POST_EXEC at the application or queue level, or by using the -E option of the bsub command to specify a pre-execution command. In some situations, specifying a queue-level or application-level pre-execution command can have advantages over requiring users to use bsub -E. For example, license checking can be set up at the queue or application level so that users do not have to enter a pre-execution command every time they submit a job.

Configuration file	Parameter and syntax	Behavior
lsb.queues	PRE_EXEC=*command*	• Enables pre-execution processing at the queue level. • The pre-execution command runs on the execution host before the job starts. • If the PRE_EXEC command exits with a non-zero exit code, LSF requeues the job to the front of the queue. • The PRE_EXEC command uses the same environment variable values as the job.
	POST_EXEC= *command*	• Enables post-execution processing at the queue level. • The POST_EXEC command uses the same environment variable values as the job. • The post-execution command for the queue remains associated with the job. The original post-execution command runs even if the job is requeued or if the post-execution command for the queue is changed after job submission. • Before the post-execution command runs, LSB_JOBEXIT_STAT is set to the exit status of the job. The success or failure of the post-execution command has no effect on LSB_JOBEXIT_STAT. • The post-execution command runs after the job finishes, even if the job fails. • Specify the environment variable $USER_POSTEXEC to allow UNIX users to define their own post-execution commands.

Configuration file	Parameter and syntax	Behavior
lsb.applications	PRE_EXEC=*command*	• Enables pre-execution processing at the application level. • The pre-execution command runs on the execution host before the job starts. • If the PRE_EXEC command exits with a non-zero exit code, LSF requeues the job to the front of the queue. • The PRE_EXEC command uses the same environment variable values as the job.
	POST_EXEC= *command*	• Enables post-execution processing at the application level. • The POST_EXEC command uses the same environment variable values as the job. • The post-execution command for the application profile remains associated with the job. The original post-execution command runs even if the job is moved to a different application profile or is requeued, or if the post-execution command for the original application profile is changed after job submission. • Before the post-execution command runs, LSB_JOBEXIT_STAT is set to the exit status of the job. The success or failure of the post-execution command has no effect on LSB_JOBEXIT_STAT. • The post-execution command runs after the job finishes, even if the job fails. • Specify the environment variable $USER_POSTEXEC to allow UNIX users to define their own post-execution commands.

Pre- and post-execution processing behavior

Pre- and post-execution processing applies to both UNIX and Windows hosts.

Host type	Environment
UNIX	• The pre- and post-execution commands run in the /tmp directory under /bin/sh -c, which allows the use of shell features in the commands. The following example shows valid configuration lines: PRE_EXEC= /usr/share/lsf/misc/testq_pre >> /tmp/pre.out POST_EXEC= /usr/share/lsf/misc/testq_post	grep -v "Testing..." • LSF sets the PATH environment variable to PATH='/bin /usr/bin /sbin /usr/sbin' • The stdin, stdout, and stderr are set to /dev/null
Windows	• The pre- and post-execution commands run under cmd.exe /c • The standard input, standard output, and standard error are set to NULL • The PATH is determined by the setup of the LSF Service

Note:

> If the pre-execution or post-execution command is not in your usual execution path, you must specify the full path name of the command.

Order of command execution

Pre-execution commands run in the following order:

1. The queue-level command
2. The application-level or job-level command. If you specify a command at both the application and job levels, the job-level command overrides the application-level command; the application-level command is ignored.

If a pre-execution command is specified at the …	Then the commands execute in the order of …
Queue, application, and job levels	1. Queue level 2. Job level
Queue and application levels	1. Queue level 2. Application level
Queue and job levels	1. Queue level 2. Job level
Application and job levels	1. Job level

Post-execution commands run in the following order:

1. The application-level command
2. The queue-level command
3. The job-level command

If both application-level (POST_EXEC in lsb.applications) and job-level post-execution commands are specified, job level post-execution overrides application-level post-execution commands.

If a post-execution command is specified at the …	Then the commands execute in the order of …
Queue, application, and job levels	1. Job level 2. Queue level
Queue and application levels	1. Application level 2. Queue level
Queue and job levels	1. Job level 2. Queue level

Pre-execution command behavior

A pre-execution command returns information to LSF by means of the exit status. LSF holds the job in the queue until the specified pre-execution command returns an exit code of zero (0). If the pre-execution command exits with a non-zero value, the job pends until LSF tries

again to dispatch it. While the job remains in the PEND state, LSF dispatches other jobs to the execution host.

If the pre-execution command exits with a value of 99, the job exits without pending. This allows you to cancel the job if the pre-execution command fails.

You must ensure that the pre-execution command runs without side effects; that is, you should define a pre-execution command that does not interfere with the job itself. For example, if you use the pre-execution command to reserve a resource, you cannot also reserve the same resource as part of the job submission.

LSF users can specify a pre-execution command at job submission. LSF first finds a suitable host on which to run the job and then runs the pre-execution command on that host. If the pre-execution command runs successfully and returns an exit code of zero, LSF runs the job.

Post-execution command behavior

A post-execution command runs after the job finishes, regardless of the exit state of the job. Once a post-execution command is associated with a job, that command runs even if the job fails. You cannot configure the post-execution command to run only under certain conditions.

The resource usage of post-execution processing is not included in the job resource usage calculation, and post-execution command exit codes are not reported to LSF.

If POST_EXEC=$USER_POSTEXEC in either lsb.applications or lsb.queues, UNIX users can define their own post-execution commands:

```
setenv USER_POSTEXEC /path_name
```

where the path name for the post-execution command is an absolute path.

If POST_EXEC=$USER_POSTEXEC and …	Then …
The user defines the USER_POSTEXEC environment variable	• LSF runs the post-execution command defined by the environment variable USER_POSTEXEC • After the user-defined command runs, LSF reports successful completion of post-execution processing • If the user-defined command fails, LSF reports a failure of post-execution processing
The user does not define the USER_POSTEXEC environment variable	• LSF reports successful post-execution processing without actually running a post-execution command

Important:

Do not allow users to specify a post-execution command when the pre- and post-execution commands are set to run under the root account.

Configuration to modify pre- and post-execution processing

Configuration parameters modify various aspects of pre- and post-execution processing behavior by:

- Preventing a new job from starting until post-execution processing has finished
- Controlling the length of time post-execution processing can run
- Specifying a user account under which the pre- and post-execution commands run

· Controlling how many times pre-execution retries

Configuration to modify when new jobs can start

When a job finishes, sbatchd reports a job finish status of DONE or EXIT to mbatchd. This causes LSF to release resources associated with the job, allowing new jobs to start on the execution host before post-execution processing from a previous job has finished.

In some cases, you might want to prevent the overlap of a new job with post-execution processing. Preventing a new job from starting prior to completion of post-execution processing can be configured at the application level or at the job level.

At the job level, the bsub -w option allows you to specify job dependencies; the keywords post_done and post_err cause LSF to wait for completion of post-execution processing before starting another job.

At the application level:

File	Parameter and syntax	Description
lsb.applications lsb.params	JOB_INCLUDE_POSTPROC=Y	· Enables completion of post-execution processing before LSF reports a job finish status of DONE or EXIT · Prevents a new job from starting on a host until post-execution processing is finished on that host

· sbatchd sends both job finish status (DONE or EXIT) and post-execution processing status (POST_DONE or POST_ERR) to mbatchd at the same time
· The job remains in the RUN state and holds its job slot until post-execution processing has finished
· Job requeue happens (if required) after completion of post-execution processing, not when the job itself finishes
· For job history and job accounting, the job CPU and run times include the post-execution processing CPU and run times
· The job control commands bstop, bkill, and bresume have no effect during post-execution processing
· If a host becomes unavailable during post-execution processing for a rerunnable job, mbatchd sees the job as still in the RUN state and reruns the job
· LSF does not preempt jobs during post-execution processing

Configuration to modify the post-execution processing time

Controlling the length of time post-execution processing can run is configured at the application level.

File	Parameter and syntax	Description

`lsb.applications`
 	
`lsb.params`
	JOB_POSTPROC_TIMEOUT=
	minutes
		• Specifies the length of time, in minutes, that post-execution processing can run.
		• The specified value must be greater than zero.
		• If post-execution processing takes longer than the specified value, sbatchd reports post-execution failure—a status of POST_ERR—and kills the process group of the job's post-execution processes. This kills the parent process only.
		• If JOB_INCLUDE_POSTPROC=Y and sbatchd kills the post-execution process group, post-execution processing CPU time is set to zero, and the job's CPU time does not include post-execution CPU time.

Configuration to modify the pre- and post-execution processing user account

Specifying a user account under which the pre- and post-execution commands run is configured at the system level. By default, both the pre- and post-execution commands run under the account of the user who submits the job.

File	Parameter and syntax	Description
`lsf.sudoers`	LSB_PRE_POST_EXEC_USER= *user_name*	• Specifies the user account under which pre- and post-execution commands run (UNIX only) • This parameter applies only to pre- and post-execution commands configured at the application and queue levels; pre-execution commands defined at the job level with `bsub -E` run under the account of the user who submits the job • If the pre-execution or post-execution commands perform privileged operations that require root permissions on UNIX hosts, specify a value of root • You must edit the `lsf.sudoers` file on all UNIX hosts within the cluster and specify the same user account

Configuration to control how many times pre-execution retries

By default, if job pre-execution fails, LSF retries the job automatically. The job remains in the queue and pre-execution is retried 5 times by default, to minimize any impact to performance and throughput.

Limiting the number of times LSF retries job pre-execution is configured cluster-wide (`lsb.params`), at the queue level (`lsb.queues`), and at the application level (`lsb.applications`). pre-execution retry in `lsb.applications` overrides `lsb.queues`, and `lsb.queues` overrides `lsb.params` configuration.

Feature: Pre-execution and post-execution processing

Configuration file	Parameter and syntax	Behavior
lsb.params	LOCAL_MAX_PREEXEC_RETRY=*integer*	• Controls the maximum number of times to attempt the pre-execution command of a job on the local cluster. • Specify an integer greater than 0 By default, the number of retries is unlimited.
	MAX_PREEXEC_RETRY=*integer*	• Controls the maximum number of times to attempt the pre-execution command of a job on the remote cluster. • Specify an integer greater than 0 By default, the number of retries is 5.
	REMOTE_MAX_PREEXEC_RETRY=*integer*	• Controls the maximum number of times to attempt the pre-execution command of a job on the remote cluster. Equivalent to MAX_PREEXEC_RETRY • Specify an integer greater than 0 By default, the number of retries is 5.
lsb.queues	LOCAL_MAX_PREEXEC_RETRY=*integer*	• Controls the maximum number of times to attempt the pre-execution command of a job on the local cluster. • Specify an integer greater than 0 By default, the number of retries is unlimited.
	MAX_PREEXEC_RETRY=*integer*	• Controls the maximum number of times to attempt the pre-execution command of a job on the remote cluster. • Specify an integer greater than 0 By default, the number of retries is 5.
	REMOTE_MAX_PREEXEC_RETRY=*integer*	• Controls the maximum number of times to attempt the pre-execution command of a job on the remote cluster. Equivalent to MAX_PREEXEC_RETRY • Specify an integer greater than 0 By default, the number of retries is 5.
lsb.applications	LOCAL_MAX_PREEXEC_RETRY=*integer*	• Controls the maximum number of times to attempt the pre-execution command of a job on the local cluster. • Specify an integer greater than 0 By default, the number of retries is unlimited.

Configuration file	Parameter and syntax	Behavior
	MAX_PREEXEC_RETRY=*integer*	• Controls the maximum number of times to attempt the pre-execution command of a job on the remote cluster. • Specify an integer greater than 0 By default, the number of retries is 5.
	REMOTE_MAX_PREEXEC_RETRY=*integer*	• Controls the maximum number of times to attempt the pre-execution command of a job on the remote cluster. Equivalent to MAX_PREEXEC_RETRY • Specify an integer greater than 0 By default, the number of retries is 5.

When pre-execution retry is configured, if a job pre-execution fails and exits with non-zero value, the number of pre-exec retries is set to 1. When the pre-exec retry limit is reached, the job is suspended with PSUSP status.

The number of times that pre-execution is retried includes queue-level, application-level, and job-level pre-execution command specifications. When pre-execution retry is configured, a job will be suspended when the sum of its queue-level pre-exec retry times + application-level pre-exec retry times is greater than the value of the pre-execution retry parameter or if the sum of its queue-level pre-exec retry times + job-level pre-exec retry times is greater than the value of the pre-execution retry parameter.

The pre-execution retry limit is recovered when LSF is restarted and reconfigured. LSF replays the pre-execution retry limit in the PRE_EXEC_START or JOB_STATUS events in lsb.events.

Pre- and post-execution processing commands

Commands for submission

The bsub -E option specifies a pre-execution command. Post-execution commands cannot be specified using bsub; post-execution processing can only be defined at the queue and application levels.

The bsub -w option allows you to specify job dependencies that cause LSF to wait for completion of post-execution processing before starting another job.

Command	Description
bsub -E *command*	• Defines the pre-execution command at the job level.
bsub -w 'post_done(job_id | "job_name")'	• Specifies the job dependency condition required to prevent a new job from starting on a host until post-execution processing on that host has finished without errors.
bsub -w 'post_err(job_id | "job_name")'	• Specifies the job dependency condition required to prevent a new job from starting on a host until post-execution processing on that host has exited with errors.

Commands to monitor

Command	Description
`bhist` `bhist -l`	• Displays the host status of all hosts, specific hosts, or specific host groups, including the POST_DONE and POST_ERR states, if the user submitted the job with the -w option of `bsub`. • The CPU and run times shown do not include resource usage for post-execution processing unless the parameter JOB_INCLUDE_POSTPROC is defined in `lsb.applications` or `lsb.params`. • Displays the job exit code and reason if the pre-exec retry limit is exceeded.
`bjobs -l`	• Displays information about pending, running, and suspended jobs. During post-execution processing, the job status will be RUN if the parameter JOB_INCLUDE_POSTPROC is defined in `lsb.applications or lsb.params`. • The resource usage shown does not include resource usage for post-execution processing. • Displays the job exit code and reason if the pre-exec retry limit is exceeded.
`bacct`	• Displays accounting statistics for finished jobs. • The CPU and run times shown do not include resource usage for post-execution processing, unless the parameter JOB_INCLUDE_POSTPROC is defined in `lsb.applications or lsb.params`.

Commands to control

Command	Description
`bmod` -E *command*	• Changes the pre-execution command at the job level.
`bmod` -w 'post_done(job_id | "job_name")'	• Specifies the job dependency condition required to prevent a new job from starting on a host until post-execution processing on that host has finished without errors.
`bmod` -w 'post_err(job_id | "job_name")'	• Specifies the job dependency condition required to prevent a new job from starting on a host until post-execution processing on that host has exited with errors.

Commands to display configuration

Command	Description
`bapp` -l	• Displays information about application profiles configured in `lsb.applications`, including the values defined for PRE_EXEC, POST_EXEC, JOB_INCLUDE_POSTPROC, JOB_POSTPROC_TIMEOUT, LOCAL_MAX_PREEXEC_RETRY, MAX_PREEXEC_RETRY, and REMOTE_MAX_PREEXEC_RETRY.
`bparams`	• Displays the value of parameters defined in `lsb.params`, including the values defined for LOCAL_MAX_PREEXEC_RETRY, MAX_PREEXEC_RETRY, and REMOTE_MAX_PREEXEC_RETRY.

Command	Description
bqueues -l	• Displays information about queues configured in lsb.queues, including the values defined for PRE_EXEC and POST_EXEC, LOCAL_MAX_PREEXEC_RETRY, MAX_PREEXEC_RETRY, and REMOTE_MAX_PREEXEC_RETRY.

Use a text editor to view the lsf.sudoers configuration file.

Feature: Preemptive scheduling

The preemptive scheduling feature allows a pending high-priority job to preempt a running job of lower priority. The lower-priority job is suspended and is resumed as soon as possible. Use preemptive scheduling if you have long-running, low-priority jobs causing high-priority jobs to wait an unacceptably long time.

Contents

- About preemptive scheduling
- Scope
- Configuration to enable preemptive scheduling
- Preemptive scheduling behavior
- Configuration to modify preemptive scheduling behavior
- Preemptive scheduling commands

About preemptive scheduling

Preemptive scheduling takes effect when two jobs compete for the same job slots. If a high-priority job is pending, LSF can suspend a lower-priority job that is running, and then start the high-priority job instead. For this to happen, the high-priority job must be pending in a *preemptive queue* (a queue that can preempt other queues), or the low-priority job must belong to a *preemptable queue* (a queue that can be preempted by other queues).

If multiple slots are required, LSF can preempt multiple jobs until sufficient slots are available. For example, one or more jobs can be preempted for a job that needs multiple job slots.

A preempted job is resumed as soon as more job slots become available; it does not necessarily have to wait for the preempting job to finish.

Preemptive queue	Jobs in a preemptive queue can preempt jobs in any queue of lower priority, even if the lower-priority queues are not specified as preemptable. Preemptive queues are more aggressive at scheduling jobs because a slot that is not available to a low-priority queue may be available by preemption to a high-priority queue.
Preemptable queue	Jobs in a preemptable queue can be preempted by jobs from any queue of a higher priority, even if the higher-priority queues are not specified as preemptive. When multiple preemptable jobs exist (low-priority jobs holding the required slots), and preemption occurs, LSF preempts a job from the least-loaded host.

Resizable jobs

Resize allocation requests are not able take advantage of the queue-based preemption mechanism to preempt other jobs. However, regular pending jobs are still able to preempt running resizable jobs, even while they have a resize request pending. When a resizable job is preempted and goes to the SSUSP state, its resize request remains pending and LSF stops scheduling it until it returns back to RUN state.

- New pending allocation requests cannot make use of preemption policy to get slots from other running or suspended jobs.
- Once a resize decision has been made, LSF updates its job counters to be reflected in future preemption calculations. For instance, resizing a running preemptable job from 2 slots to 4 slots, makes 4 preemptable slots for high priority pending jobs.
- If a job is suspended, LSF stops allocating resources to a pending resize request.
- When a preemption decision is made, if job has pending resize request and scheduler already has made an allocation decision for this request, LSF cancels the allocation decision.
- If a preemption decision is made while a job resize notification command is running, LSF prevents the suspend signal from reaching the job.

Scope

Preemptive scheduling does not apply to jobs that have been forced to run or backfill and exclusive jobs.

Limitations	Description
Exclusive compute units	Jobs requesting exclusive use of compute units in the resource requirements cannot preempt other jobs.
	Jobs using compute units exclusively cannot be preempted.

Default behavior (preemptive scheduling not enabled)
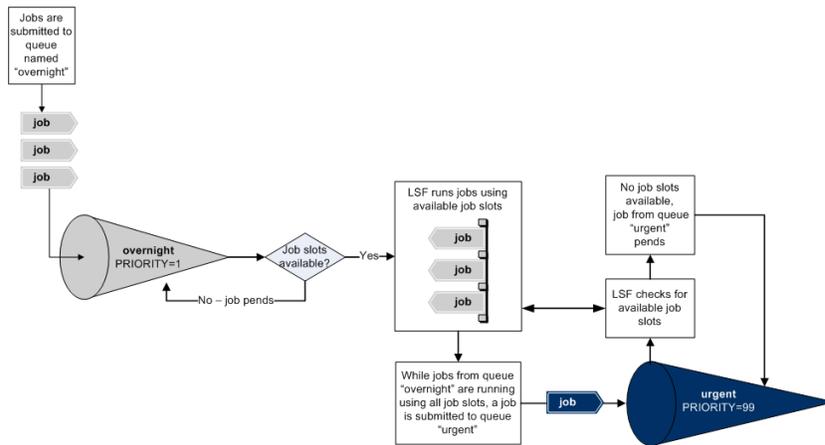
With preemptive scheduling enabled (preemptive queue)
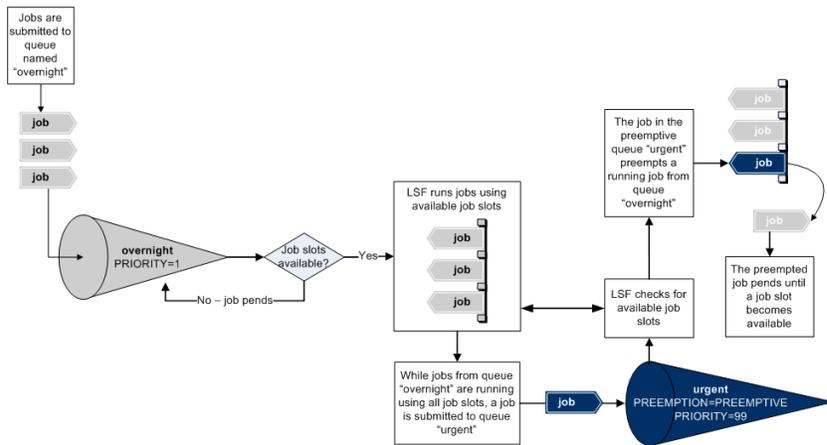
With preemptive scheduling enabled (preemptable queue)
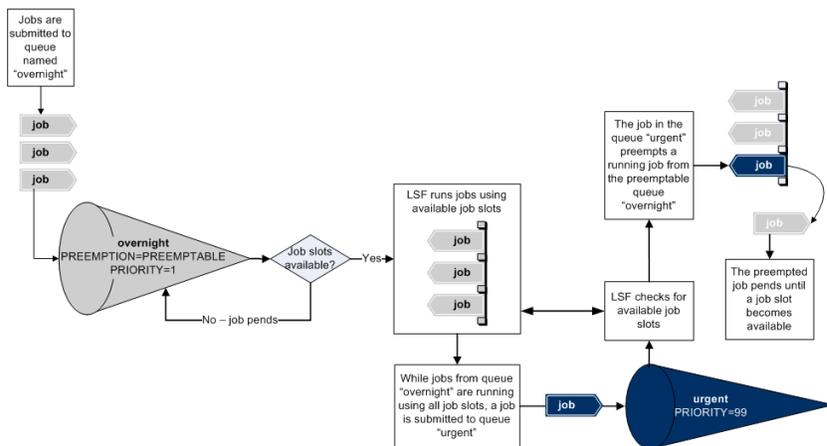
Configuration to enable preemptive scheduling

The preemptive scheduling feature is enabled by defining at least one queue as preemptive or preemptable, using the PREEMPTION parameter in the lsb.queues file. Preemption does not actually occur until at least one queue is assigned a higher relative priority than another queue, using the PRIORITY parameter, which is also set in the lsb.queues file.

Both PREEMPTION and PRIORITY are used to determine which queues can preempt other queues, either by establishing relative priority of queues or by specifically defining preemptive properties for a queue.

Configuration file	Parameter and syntax	Default behavior
l sb. queues	PREEMPTION=PREEMPTIVE	• Enables preemptive scheduling • Jobs in this queue can preempt jobs in any queue of lower priority, even if the lower-priority queue is not specified as preemptable
	PREEMPTION=PREEMPTABLE	• Enables preemptive scheduling • Jobs in this queue can be preempted by jobs from any queue of higher priority, even if the higher-priority queue is not specified as preemptive
	PRIORITY=*integer*	• Sets the priority for this queue relative to all other queues • The larger the number, the higher the priority—a queue with PRIORITY=99 has a higher priority than a queue with PRIORITY=1

Preemptive scheduling behavior

Preemptive scheduling is based primarily on parameters specified at the queue level: some queues are eligible for preemption, others are not. Once a hierarchy of queues has been established, other factors determine which jobs from a queue should be preempted.

There are three ways to establish which queues should be preempted:

- Based on queue priority—the PREEMPTION parameter defines a queue as preemptive or preemptable and preemption is based on queue priority, where jobs from higher-priority queues can preempt jobs from lower-priority queues
- Based on a preferred order—the PREEMPTION parameter defines queues that can preempt other queues, in a preferred order
- Explicitly, by specific queues—the PREEMPTION parameter defines queues that can be preempted, and by which queues

When …	The behavior is …
Preemption is not enabled—no queue is defined as preemptable, and no queue is defined as preemptive	• High-priority jobs do not preempt jobs that are already running
A queue is defined as preemptable, but no specific queues are listed that can preempt it	• Jobs from this queue can be preempted by jobs from any queue with a higher value for priority
A queue is defined as preemptable, and one or more queues are specified that can preempt it	• Jobs from this queue can be preempted only by jobs from the specified queues
A queue is defined as preemptive, but no specific queues are listed that it can preempt	• Jobs from this queue preempt jobs from all queues with a lower value for priority • Jobs are preempted from the least-loaded host

When …	The behavior is …
A queue is defined as preemptive, and one or more specific queues are listed that it can preempt, but no queue preference is specified	• Jobs from this queue preempt jobs from any queue in the specified list • Jobs are preempted on the least-loaded host first
A queue is defined as preemptive, and one or more queues have a preference number specified, indicating a preferred order of preemption	• Queues with a preference number are preferred for preemption over queues without a preference number • Queues with a higher preference number are preferred for preemption over queues with a lower preference number • For queues that have the same preference number, the queue with lowest priority is preferred for preemption over queues with higher priority • For queues without a preference number, the queue with lower priority is preferred for preemption over the queue with higher priority
A queue is defined as preemptive, or a queue is defined as preemptable, and preemption of jobs with the shortest run time is configured	• A queue from which to preempt a job is determined based on other parameters as shown above • The job that has been running for the shortest period of time is preempted
A queue is defined as preemptive, or a queue is defined as preemptable, and preemption of jobs that will finish within a certain time period is prevented	• A queue from which to preempt a job is determined based on other parameters as shown above • A job that has a run limit or a run time specified and that will not finish within the specified time period is preempted
A queue is defined as preemptive, or a queue is defined as preemptable, and preemption of jobs with the specified run time is prevented	• A queue from which to preempt a job is determined based on other parameters as shown above • The job that has been running for less than the specified period of time is preempted

Case study: Three queues with varying priority

Consider the case where three queues are defined as follows:

Queue A has the highest relative priority, with a value of 99
Queue B is both preemptive and preemptable, and has a relative priority of 10
Queue C has the lowest relative priority, with the default value of 1

The queues can preempt as follows:

• A can preempt B because B is preemptable and B has a lower priority than A
• B can preempt C because B is preemptive and C has a lower priority than B
• A cannot preempt C, even though A has a higher priority than C, because A is not preemptive, nor is C preemptable

Calculation of job slots in use

The number of job slots in use determines whether preemptive jobs can start. The method in which the number of job slots in use is calculated can be configured to ensure that a preemptive job can start. When a job is preempted, it is suspended. If the suspended job still counts towards the total number of jobs allowed in the system, based on the limits imposed in the

lsb.resources file, suspending the job may not be enough to allow the preemptive job to run.

The PREEMPT_FOR parameter is used to change the calculation of job slot usage, ignoring suspended jobs in the calculation. This ensures that if a limit is met, the preempting job can actually run.

When …	The effect on the calculation of job slots used is …
Preemption is not enabled	• Job slot limits are enforced based on the number of job slots taken by both running and suspended jobs. • Job slot limits specified at the queue level are enforced for both running and suspended jobs.
Preemption is enabled	• The total number of jobs at both the host and individual user level is not limited by the number of suspended jobs—only running jobs are considered. • The number of running jobs never exceeds the job slot limits. If starting a preemptive job violates a job slot limit, a lower-priority job is suspended to run the preemptive job. If, however, a job slot limit is still violated (i.e. the suspended job still counts in the calculation of job slots in use), the preemptive job still cannot start. • Job slot limits specified at the queue level are always enforced for both running and suspended jobs. • When preemptive scheduling is enabled, suspended jobs never count against the total job slot limit for individual users.
Preemption is enabled, and PREEMPT_FOR=GROUP_JLP	• Only running jobs are counted when calculating the per-processor job slots in use for a user group, and comparing the result with the limit specified at the user level.
Preemption is enabled, and PREEMPT_FOR=GROUP_MAX	• Only running jobs are counted when calculating the job slots in use for this user group, and comparing the result with the limit specified at the user level.
Preemption is enabled, and PREEMPT_FOR=HOST_JLU	• Only running jobs are counted when calculating the total job slots in use for a user group, and comparing the result with the limit specified at the host level. Suspended jobs do not count against the limit for individual users.
Preemption is enabled, and PREEMPT_FOR=USER_JLP	• Only running jobs are counted when calculating the per-processor job slots in use for an individual user, and comparing the result with the limit specified at the user level.

Preemption of backfill jobs

With preemption of backfill jobs enabled (PREEMPT_JOBTYPE=BACKFILL in lsb.params), LSF maintains the priority of jobs with resource or slot reservations by preventing lower-priority jobs that preempt backfill jobs from "stealing" resources from jobs with reservations. Only jobs from queues with a higher priority than queues that define resource or slot reservations can preempt backfill jobs. For example,

If …	Is configured …	And a priority of …	The behavior is …
queueR	With a resource or slot reservation	80	Jobs in this queue reserve resources. If backfill scheduling is enabled, backfill jobs with a defined run limit can use the resources.
queueB	As a preemptable backfill queue	50	Jobs in queueB with a defined run limit use job slots reserved by jobs in queueR.
queueP	As a preemptive queue	75	Jobs in this queue do not necessarily have a run limit. LSF prevents jobs from this queue from preempting backfill jobs because queueP has a lower priority than queue R.

To guarantee a minimum run time for interruptible backfill jobs, LSF suspends them upon preemption. To change this behavior so that LSF terminates interruptible backfill jobs upon preemption, you must define the parameter TERMINATE_WHEN=PREEMPT in lsb.queues.

Configuration to modify preemptive scheduling behavior

There are configuration parameters that modify various aspects of preemptive scheduling behavior, by

- Modifying the selection of the queue to preempt jobs from
- Modifying the selection of the job to preempt
- Modifying preemption of backfill and exclusive jobs
- Modifying the way job slot limits are calculated
- Modifying the number of jobs to preempt for a parallel job
- Modifying the control action applied to preempted jobs
- Control how many times a job can be preempted

Configuration to modify selection of queue to preempt

File	Parameter	Syntax and description
lsb.queues	PREEMPTION	PREEMPTION=PREEMPTIVE[*low_queue+pref* …] • Jobs in this queue can preempt running jobs from the specified queues, starting with jobs in the queue with the highest value set for preference PREEMPTION=PREEMPTABLE[*hi_queue* …] • Jobs in this queue can be preempted by jobs from the specified queues
	PRIORITY=*integer*	• Sets the priority for this queue relative to all other queues • The higher the priority value, the more likely it is that jobs from this queue may preempt jobs from other queues, and the less likely it is for jobs from this queue to be preempted by jobs from other queues

Configuration to modify selection of job to preempt

Files	Parameter	Syntax and description
`lsb.params` `lsb.applications`	PREEMPT_FOR	PREEMPT_FOR=LEAST_RUN_TIME • Preempts the job that has been running for the shortest time
	NO_PREEMPT_RUN_TIME	NO_PREEMPT_RUN_TIME=% • Prevents preemption of jobs that have been running for the specified percentage of minutes, or longer • If NO_PREEMPT_RUN_TIME is specified as a percentage, the job cannot be preempted after running the percentage of the job duration. For example, if the job run limit is 60 minutes and NO_PREEMPT_RUN_TIME=50%, the job cannot be preempted after it running 30 minutes or longer. • If you specify percentage for NO_PREEMPT_RUN_TIME, requires a run time (`bsub -We` or RUNTIME in `lsb.applications`), or run limit to be specified for the job (`bsub -W`, or RUNLIMIT in `lsb.queues`, or RUNLIMIT in `lsb.applications`)
	NO_PREEMPT_FINISH_TIME	NO_PREEMPT_FINISH_TIME=% • Prevents preemption of jobs that will finish within the specified percentage of minutes • If NO_PREEMPT_FINISH_TIME is specified as a percentage, the job cannot be preempted if the job finishes within the percentage of the job duration. For example, if the job run limit is 60 minutes and NO_PREEMPT_FINISH_TIME=10%, the job cannot be preempted after it running 54 minutes or longer. • If you specify percentage for NO_PREEMPT_RUN_TIME, requires a run time (`bsub -We` or RUNTIME in `lsb.applications`), or run limit to be specified for the job (`bsub -W`, or RUNLIMIT in `lsb.queues`, or RUNLIMIT in `lsb.applications`)

Configuration to modify preemption of backfill and exclusive jobs

File	Parameter	Syntax and description
`lsb.params`	PREEMPT_JOBTYPE	**PREEMPT_JOBTYPE=BACKFILL** • Enables preemption of backfill jobs. • Requires the line PREEMPTION=PREEMPTABLE in the queue definition. • Only jobs from queues with a higher priority than queues that define resource or slot reservations can preempt jobs from backfill queues. **PREEMPT_JOBTYPE=EXCLUSIVE** • Enables preemption of and preemption by exclusive jobs. • Requires the line PREEMPTION=PREEMPTABLE or PREEMPTION=PREEMPTIVE in the queue definition. • Requires the definition of LSB_DISABLE_LIMLOCK_EXCL in lsf.conf. **PREEMPT_JOBTYPE=EXCLUSIVE BACKFILL** • Enables preemption of exclusive jobs, backfill jobs, or both.
`lsf.conf`	LSB_DISABLE_LIMLOCK_EXCL	**LSB_DISABLE_LIMLOCK_EXCL=y** • Enables preemption of exclusive jobs. • For a host running an exclusive job: ◦ `lsload` displays the host status ok. ◦ `bhosts` displays the host status closed. ◦ Users can run tasks on the host using `lsrun` or `lsgrun`. To prevent users from running tasks during execution of an exclusive job, the parameter LSF_DISABLE_LSRUN=y must be defined in `lsf.conf`. • Changing this parameter requires a restart of all sbatchds in the cluster (`badmin hrestart`). Do not change this parameter while exclusive jobs are running.

Configuration to modify how job slot usage is calculated

File	Parameter	Syntax and description
lsb.params	PREEMPT_FOR	**PREEMPT_FOR=GROUP_JLP** • Counts only running jobs when evaluating if a user group is approaching its per-processor job slot limit (SLOTS_PER_PROCESSOR, USERS, and PER_HOST=all in the lsb.resources file), ignoring suspended jobs
		PREEMPT_FOR=GROUP_MAX • Counts only running jobs when evaluating if a user group is approaching its total job slot limit (SLOTS, PER_USER=all, and HOSTS in the lsb.resources file), ignoring suspended jobs
		PREEMPT_FOR=HOST_JLU • Counts only running jobs when evaluating if a user or user group is approaching its per-host job slot limit (SLOTS, PER_USER=all, and HOSTS in the lsb.resources file), ignoring suspended jobs
		PREEMPT_FOR=USER_JLP • Counts only running jobs when evaluating if a user is approaching their per-processor job slot limit (SLOTS_PER_PROCESSOR, USERS, and PER_HOST=all in the lsb.resources file) • Ignores suspended jobs when calculating the per-processor job slot limit for individual users

Configuration to modify preemption of parallel jobs

File	Parameter	Syntax and description
lsb.params	PREEMPT_FOR	**PREEMPT_FOR=MINI_JOB** • Optimizes preemption of parallel jobs by preempting only enough low-priority parallel jobs to start the high-priority parallel job
		PREEMPT_FOR=OPTIMAL_MINI_JOB • Optimizes preemption of parallel jobs by preempting only low-priority parallel jobs using the least number of slots to allow the high-priority parallel job to start

Configuration to modify the control action applied to preempted jobs

File	Parameter	Syntax and description
lsb.queues	TERMINATE_WHEN	TERMINATE_WHEN=PREEMPT • Changes the default control action of SUSPEND to TERMINATE so that LSF terminates preempted jobs

Configuration to control how many times a job can be preempted

By default, if preemption is enabled, there is actually no guarantee that a job will ever actually complete. A lower priority job could be preempted again and again, and ultimately end up being killed due to a run limit.

Limiting the number of times a job can be preempted is configured cluster-wide (lsb.params), at the queue level (lsb.queues), and at the application level (lsb.applications). MAX_JOB_PREEMPT in lsb.applications overrides lsb.queues, and lsb.queues overrides lsb.params configuration.

Files	Parameter	Syntax and description
lsb.params lsb.queues lsb.applications	MAX_JOB_PREEMPT	MAX_JOB_PREEMPT=*integer* • Specifies the maximum number of times a job can be preempted. • Specify a value within the following ranges: 0 < MAX_JOB_PREEMPT < INFINIT_INT INFINIT_INT is defined in lsf.h • By default, the number of preemption times is unlimited.

When MAX_JOB_PREEMPT is set, and a job is preempted by higher priority job, the number of job preemption times is set to 1. When the number of preemption times exceeds MAX_JOB_PREEMPT, the job will run to completion and cannot be preempted again.

The job preemption limit times is recovered when LSF is restarted or reconfigured.

Preemptive scheduling commands

Commands for submission

Command	Description
bsub -q *queue_name*	• Submits the job to the specified queue, which may have a run limit associated with it
bsub -W *minutes*	• Submits the job with the specified run limit, in minutes
bsub -app *application_profile_name*	• Submits the job to the specified application profile, which may have a run limit associated with it

Commands to monitor

Command	Description
bjobs -s	• Displays suspended jobs, together with the reason the job was suspended

Commands to control

Command	Description
brun	• Forces a pending job to run immediately on specified hosts. For an exclusive job, when LSB_DISABLE_LIMLOCK_EXCL=y , LSF allows other jobs already running on the host to finish but does not dispatch any additional jobs to that host until the exclusive job finishes.

Commands to display configuration

Command	Description
bqueues	• Displays the priority (PRIO) and run limit (RUNLIMIT) for the queue, and whether the queue is configured to be preemptive, preemptable, or both
bhosts	• Displays the number of job slots per user for a host • Displays the number of job slots available
bparams	• Displays the value of parameters defined in lsb.params.
badmin showconf	• Displays all configured parameters and their values set in lsf.conf or ego.conf that affect mbatchd and sbatchd. Use a text editor to view other parameters in the lsf.conf or ego.conf configuration files. • In a MultiCluster environment, badmin showconf only displays the parameters of daemons on the local cluster.

Feature: UNIX/Windows user account mapping

The UNIX/Windows user account mapping feature enables cross-platform job submission and execution in a mixed UNIX/Windows environment. Using this feature, you can map Windows user accounts, which include a domain name, to UNIX user accounts, which do not include a domain name, for user accounts with the same user name on both operating systems.

Contents

- About UNIX/Windows user account mapping
- Scope
- Configuration to enable UNIX/Windows user account mapping
- UNIX/Windows user account mapping behavior
- Configuration to modify UNIX/Windows user account mapping behavior
- UNIX/Windows user account mapping commands

About UNIX/Windows user account mapping

In a mixed UNIX/Windows cluster, LSF treats Windows user names (with domain) and UNIX user names (no domain) as different users. The UNIX/Windows user account mapping feature makes job submission and execution transparent across operating systems by mapping Windows accounts to UNIX accounts. With this feature enabled, LSF sends the user account name in the format required by the operating system on the execution host.
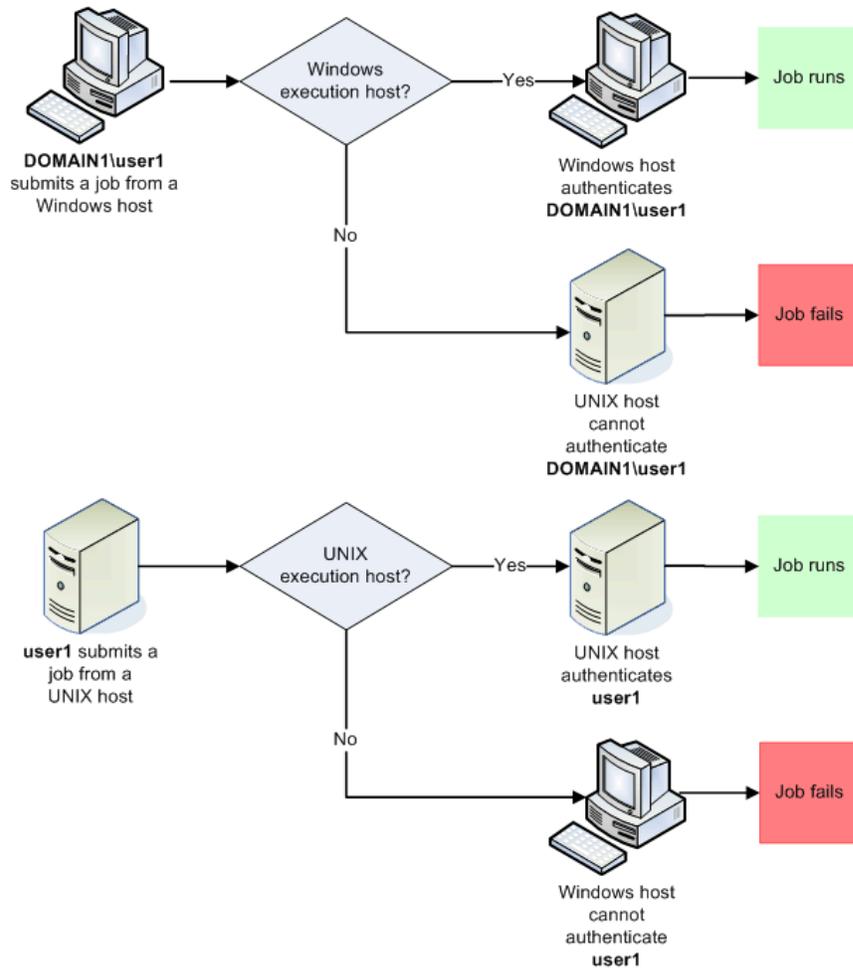
Figure 15: Default behavior (feature not enabled)
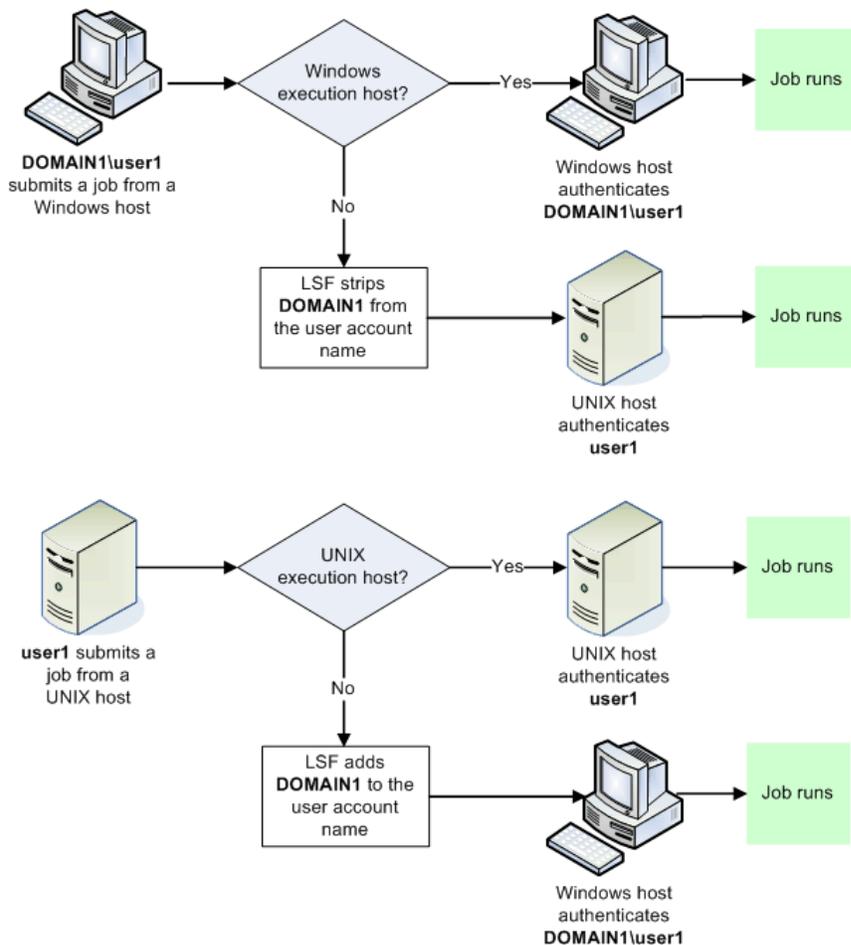
Figure 16: With UNIX/Windows user account mapping enabled

For mixed UNIX/Windows clusters, UNIX/Windows user account mapping allows you to do the following:

- Submit a job from a Windows host and run the job on a UNIX host
- Submit a job from a UNIX host and run the job on a Windows host
- Specify the domain\user combination used to run a job on a Windows host
- Schedule and track jobs submitted with either a Windows or UNIX account as though the jobs belong to a single user

LSF supports the use of both single and multiple Windows domains. In a multiple domain environment, you can choose one domain as the preferred execution domain for a particular job.

Existing Windows domain trust relationships apply in LSF. If the execution domain trusts the submission domain, the submission account is valid on the execution host.

Scope

Applicability	Details
Operating system	• UNIX and Windows hosts within a single cluster
Not required for	• Windows-only clusters • UNIX-only clusters
Dependencies	• UNIX and Windows user accounts must be valid on all hosts in the cluster and must have the correct permissions to successfully run jobs.
Limitations	• This feature works with a uniform user name space. If users at your site have different user names on UNIX and Windows hosts, you must enable between-host user account mapping. • This feature does not affect Windows workgroup installations. If you want to map all Windows workgroup users to a single Windows system account, you must configure between-host user account mapping. • This feature applies only to job execution. If you issue an LSF command or define an LSF parameter and specify a Windows user, you must use the long form of the user name, including the domain name typed in uppercase letters.

Configuration to enable UNIX/Windows user account mapping

Enable the UNIX/Windows user account mapping feature by defining one or more LSF user domains using the LSF_USER_DOMAIN parameter in lsf.conf.

Important:

Configure LSF_USER_DOMAIN immediately after you install LSF—changing this parameter in an existing cluster requires that you verify and possibly reconfigure service accounts, user group memberships, and user passwords.

Configuration file	Parameter and syntax	Behavior
`lsf.conf`	LSF_USER_DOMAIN=*domain_name*	• Enables Windows domain account mapping in a single-domain environment • To run jobs on a UNIX host, LSF strips the specified domain name from the user name • To run jobs on a Windows host, LSF appends the domain name to the user name
	LSF_USER_DOMAIN=*domain_name*:*domain_name* …	• Enables Windows domain account mapping in a multi-domain environment • To run jobs on a UNIX host, LSF strips the specified domain names from the user name • To run jobs on a Windows host, LSF appends the first domain name to the user name. If the first domain\user combination does not have permissions to run the job, LSF tries the next domain in the LSF_USER_DOMAIN list.
	LSF_USER_DOMAIN= .	• Enables Windows domain account mapping • To run jobs on a UNIX host, LSF strips the local machine name from the user name • To run jobs on a Windows host, LSF appends the local machine name to the user name

UNIX/Windows user account mapping behavior

The following examples describe how UNIX/Windows user account mapping enables job submission and execution across a mixed UNIX/Windows cluster.

When …	In the file …	And the job is submitted by …	The job …
UNIX/Windows user account mapping is not enabled	—	• BUSINESS\user1 on a Windows host	• Runs on a Windows host as BUSINESS\user1 • Fails on a UNIX host: BUSINESS\user1 is not a valid UNIX user name
UNIX/Windows user account mapping is not enabled	—	• user1 on a UNIX host	• Fails on a Windows host: Windows requires a domain\user combination • Runs on a UNIX host as user1

When …	In the file …	And the job is submitted by …	The job …
LSF_USER_DOMAIN=BUSINESS	lsf.conf	• BUSINESS\user1 on a Windows host	• Runs on a Windows host as BUSINESS\user1 • Runs on a UNIX host as user1
LSF_USER_DOMAIN=BUSINESS	lsf.conf	• user1 on a UNIX host	• Runs on a Windows host as BUSINESS\user1 • Runs on a UNIX host as user1
LSF_USER_DOMAIN= SUPPORT:ENGINEERING	lsf.conf	• SUPPORT\user1 on a Windows host	• Runs on a Windows host as SUPPORT\user1 • Runs on a UNIX host as user1
LSF_USER_DOMAIN= SUPPORT:ENGINEERING	lsf.conf	• BUSINESS\user1 on a Windows host	• Runs on a Windows host as BUSINESS\user1 • Fails on a UNIX host: LSF cannot strip the domain name, and BUSINESS\user1 is not a valid UNIX user name
LSF_USER_DOMAIN= SUPPORT:ENGINEERING	lsf.conf	• user1 on a UNIX host	• Runs on a Windows host as SUPPORT\user1; if the job cannot run with those credentials, the job runs as ENGINEERING\user1 • Runs on a UNIX host as user1

Configuration to modify UNIX/Windows user account mapping behavior

You can select a preferred execution domain for a particular job. The execution domain must be included in the LSF_USER_DOMAIN list. When you specify an execution domain, LSF ignores the order of the domains listed in LSF_USER_DOMAIN and runs the job using the specified domain. The environment variable LSF_EXECUTE_DOMAIN, defined in the user environment or from the command line, defines the preferred execution domain. Once you submit a job with an execution domain defined, you cannot change the execution domain for that particular job.

Configuration file	Parameter and syntax	Behavior
.cshrc .profile	LSF_EXECUTE_DOMAIN=*domain_name*	• Specifies the domain that LSF uses to run jobs on a Windows host • If LSF_USER_DOMAIN contains a list of multiple domains, LSF tries the LSF_EXECUTE_DOMAIN first

The following example shows the changed behavior when you define the LSF_EXECUTE_DOMAIN.

When …	In the file …	And the job is submitted by …	The job …

LSF_USER_DOMAIN= SUPPORT:ENGINEERING
 	
and
 	
LSF_EXECUTE_DOMAIN= ENGINEERING
	`lsf.conf`
	
	`.`
	`profile .`
	`cshrc`
		• user1 on a UNIX host	• Runs on a Windows host as ENGINEERING\user1; if the job cannot run with those credentials, runs as SUPPORT \user1
			• Runs on a UNIX host as user1

These additional examples are based on the following conditions:

- In `lsf.conf`, LSF_USER_DOMAIN=SALES:ENGINEERING:BUSINESS
- The user has sufficient permissions to run the job in any of the LSF user domains

UNIX user1 enters …	And LSF_EXECUTE_DOMAIN is …	Then LSF runs the job as …
bsub -m "hostb" myjob	Not defined in the user environment file	SALES\user1
bsub -m "hostb" myjob	Defined as BUSINESS in the user environment file	BUSINESS\user1
setenv LSF_EXECUTE_DOMAIN BUSINESSbsub -m "hostb" myjob	Either defined or not defined in the user environment file	BUSINESS\user1 The command line overrides the user environment file.

UNIX/Windows user account mapping commands

Commands for submission

Command	Description
bsub	• Submits the job with the user name and password of the user who entered the command. The job runs on the execution host with the same user name and password, unless you have configured UNIX/Windows user account mapping. • With UNIX/Windows user account mapping enabled, jobs that execute on a remote host run with the user account name in the format required by the operating system on the execution host.

Commands to monitor

Command	Description
bjobs -w	• Displays detailed information about jobs. • Displays the long form of the Windows user name including the domain name.

Commands to control

Command	Description
lspasswd	• Registers a password for a Windows user account. Windows users must register a password for each domain\user account using this command.

Commands to display configuration

Command	Description
bugroup -w	• Displays information about user groups. • If UNIX/Windows user account mapping is enabled, the command bugroup displays user names without domains. • If UNIX/Windows user account mapping is not enabled, the command bugroup displays user names with domains.
busers	• Displays information about specific users and user groups. • If UNIX/Windows user account mapping is enabled, the command busers displays user names without domains. • If UNIX/Windows user account mapping is not enabled, the command busers displays user names with domains.
badmin showconf	• Displays all configured parameters and their values set in lsf.conf or ego.conf that affect mbatchd and sbatchd. Use a text editor to view other parameters in the lsf.conf or ego.conf configuration files. • In a MultiCluster environment, badmin showconf only displays the parameters of daemons on the local cluster.

Feature: External job submission and execution controls

The job submission and execution controls feature enables you to use external, site-specific executables to validate, modify, and reject jobs, transfer data, and modify the job execution environment. By writing external submission (esub) and external execution (eexec) binaries or scripts, you can, for example, prevent the overuse of resources, specify execution hosts, or set required environment variables based on the job submission options.

Contents

- About job submission and execution controls
- Scope
- Configuration to enable job submission and execution controls
- Job submission and execution controls behavior
- Configuration to modify job submission and execution controls
- Job submission and execution controls commands

About job submission and execution controls

The job submission and execution controls feature uses the executables esub and eexec to control job options and the job execution environment.

External submission (esub)

An esub is an executable that you write to meet the job requirements at your site. The following are some of the things that you can use an esub to do:

- Validate job options
- Change the job options specified by a user
- Change user environment variables on the submission host (at job submission only)
- Reject jobs (at job submission only)
- Pass data to stdin of eexec

When a user submits a job using bsub or modifies a job using bmod, LSF runs the esub executable(s) on the submission host before accepting the job. If the user submitted the job with options such as -R to specify required resources or -q to specify a queue, an esub can change the values of those options to conform to resource usage policies at your site.

> **Note:**
>
> When compound resource requirements are used at any level, an esub can create job-level resource requirements which overwrite most application-level and queue-level resource requirements. -R merge rules are explained in detail in *Administering Platform LSF*.

An esub can also change the user environment on the submission host prior to job submission so that when LSF copies the submission host environment to the execution host, the job runs on the execution host with the values specified by the esub. For example, an esub can add user environment variables to those already associated with the job.

Use of esub not enabled
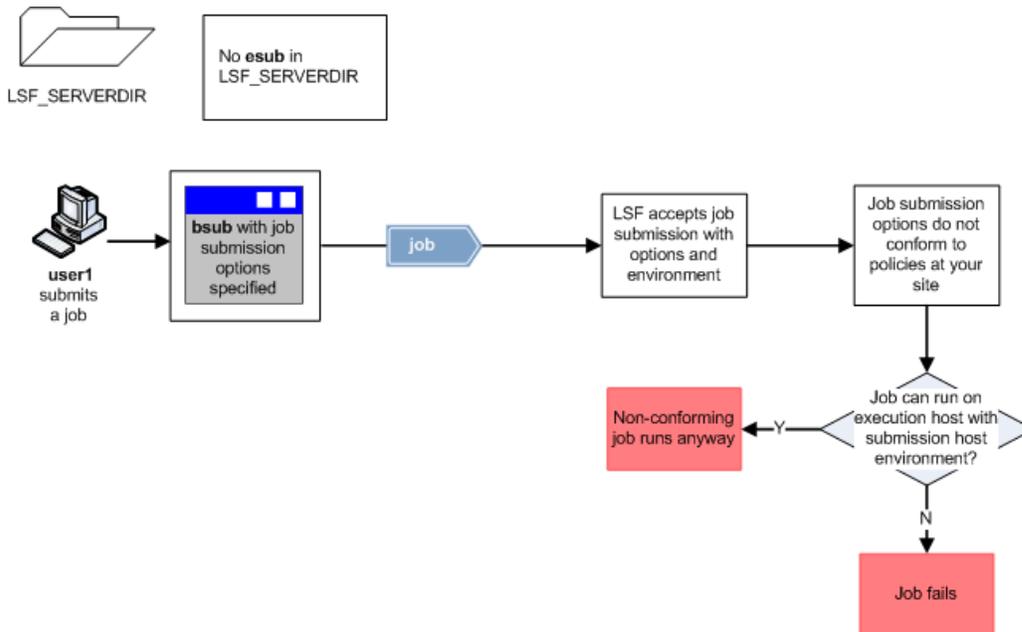
With esub enabled
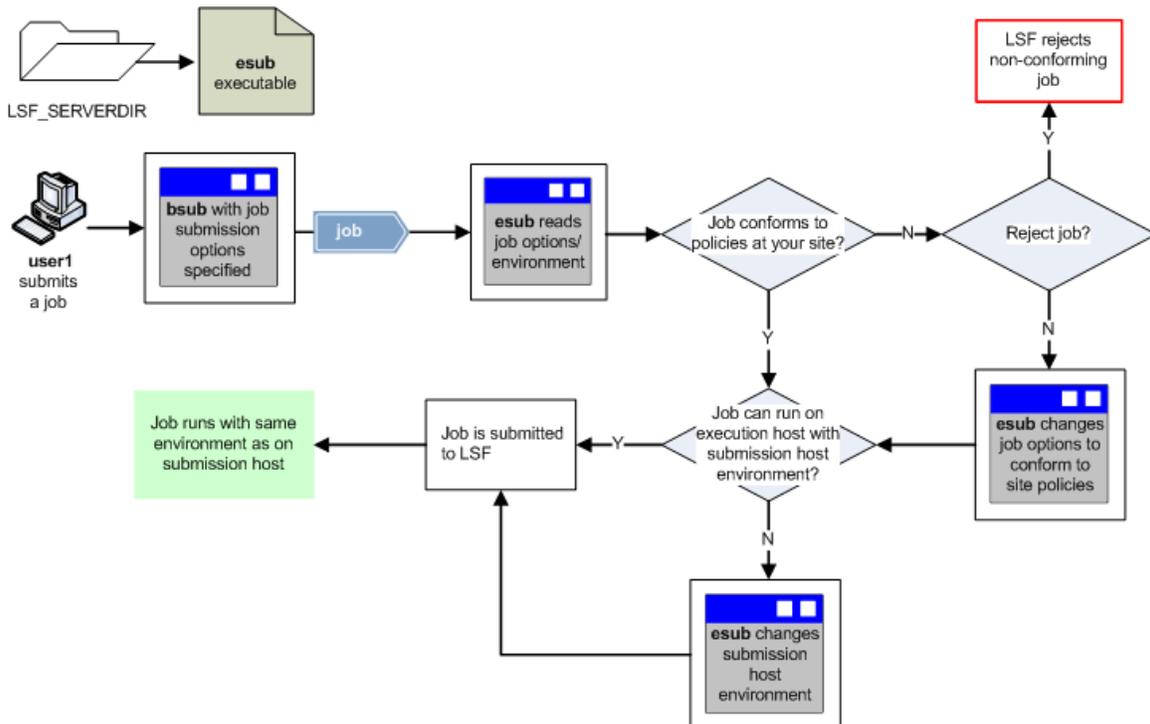
An esub executable is typically used to enforce site-specific job submission policies and command-line syntax by validating or pre-parsing the command line. The file indicated by the environment variable LSB_SUB_PARM_FILE stores the values submitted by the user. An esub reads the LSB_SUB_PARM_FILE and then accepts or changes the

option values or rejects the job. Because an esub runs before job submission, using an esub to reject incorrect job submissions improves overall system performance by reducing the load on the master batch daemon (mbatchd).

An esub can be used to:

- Reject any job that requests more than a specified number of CPUs
- Change the submission queue for specific user accounts to a higher priority queue
- Check whether the job specifies an application and, if so, submit the job to the correct application profile

Note:

If an esub executable fails, the job will still be submitted to LSF.

Multiple esub executables

LSF provides a master external submission executable (LSF_SERVERDIR/mesub) that supports the use of application-specific esub executables. Users can specify one or more esub executables using the -a option of bsub or bmod. When a user submits or modifies a job or when a user restarts a job that was submitted or modified with the -a option included, mesub runs the specified esub executables.

An LSF administrator can specify one or more mandatory esub executables by defining the parameter LSB_ESUB_METHOD in lsf.conf. If a mandatory esub is defined, mesub runs the mandatory esub for all jobs submitted to LSF in addition to any esub executables specified with the -a option.

The naming convention is esub.*application*. LSF always runs the executable named "esub" (without *.application*) if it exists in LSF_SERVERDIR.

Note:

All esub executables must be stored in the LSF_SERVERDIR directory defined in lsf.conf.

The mesub runs multiple esub executables in the following order:

1. The mandatory esub or esubs specified by LSB_ESUB_METHOD in lsf.conf
2. Any executable with the name "esub" in LSF_SERVERDIR
3. One or more esubs in the order specified by bsub -a

Example of multiple esub execution

An esub runs only once, even if it is specified by both the bsub -a option and the parameter LSB_ESUB_METHOD.
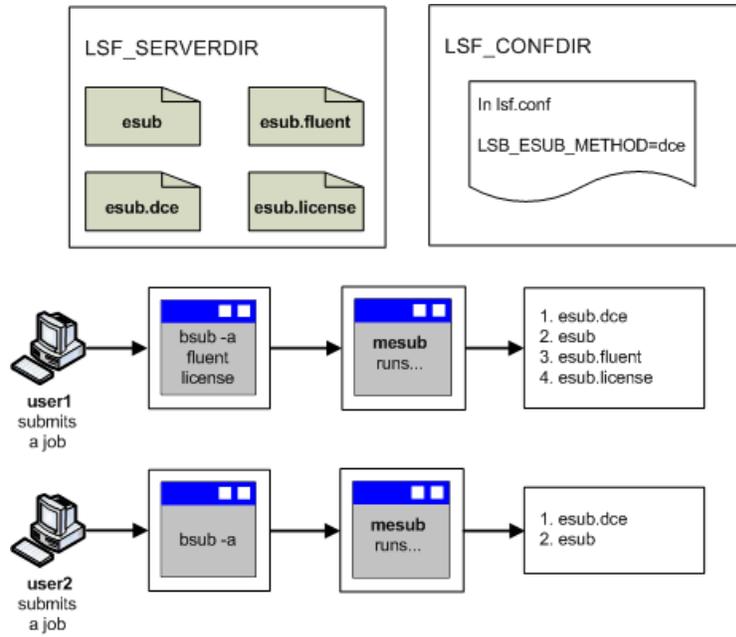
External execution (eexec)

An eexec is an executable that you write to control the job environment on the execution host.

Use of eexec not enabled
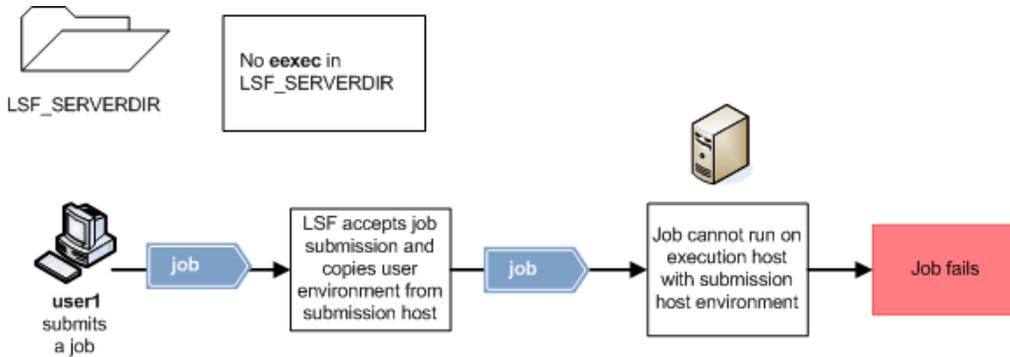
With eexec enabled
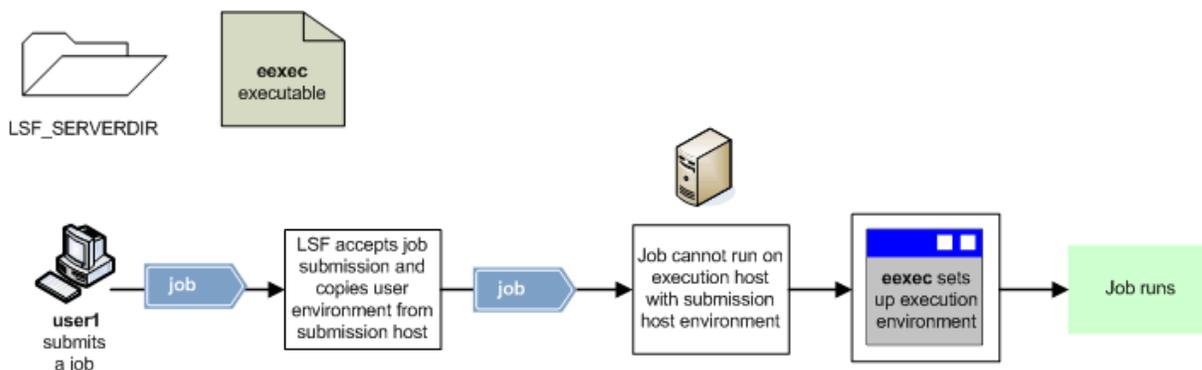
The following are some of the things that you can use an eexec to do:

- Set up the user environment variables on the execution host
- Monitor job state or resource usage
- Receive data from stdout of esub
- Run a shell script to create and populate environment variables needed by jobs
- Monitor the number of tasks running on a host and raise a flag when this number exceeds a pre-determined limit
- Pass DCE credentials and AFS tokens using a combination of esub and eexec executables; LSF functions as a pipe for passing data from the stdout of esub to the stdin of eexec

An eexec can change the user environment variable values transferred from the submission host so that the job runs on the execution host with a different environment.

For example, if you have a mixed UNIX and Windows cluster, the submission and execution hosts might use different operating systems. In this case, the submission host environment might not meet the job requirements when the job runs on the execution host. You can use an eexec to set the correct user environment between the two operating systems.

Typically, an eexec executable is a shell script that creates and populates the environment variables required by the job. An eexec can also monitor job execution and enforce site-specific resource usage policies.

The following are some of the things that you can use an eexec to do:

- Set up the user environment variables on the execution host
- Monitor job state or resource usage
- Receive data from stdout of esub
- Run a shell script to create and populate environment variables needed by jobs
- Monitor the number of tasks running on a host and raise a flag when this number exceeds a pre-determined limit
- Pass DCE credentials and AFS tokens using a combination of esub and eexec executables; LSF functions as a pipe for passing data from the stdout of esub to the stdin of eexec

If an eexec executable exists in the directory specified by LSF_SERVERDIR, LSF invokes that eexec for all jobs submitted to the cluster. By default, LSF runs eexec on the execution host before the job starts. The job process that invokes eexec waits for eexec to finish before continuing with job execution.

Unlike a pre-execution command defined at the job, queue, or application levels, an eexec:

- Runs at job start, finish, or checkpoint
- Allows the job to run without pending if eexec fails; eexec has no effect on the job state
- Runs for all jobs, regardless of queue or application profile

Scope

Applicability	Details
Operating system	• UNIX and Linux • Windows
Security	• Data passing between esub on the submission host and eexec on the execution host is not encrypted.
Job types	• Batch jobs submitted with bsub or modified by bmod. • Batch jobs restarted with brestart. • Interactive tasks submitted with lsrun and lsgrun (eexec only).

Dependencies	• UNIX and Windows user accounts must be valid on all hosts in the cluster, or the correct type of account mapping must be enabled.
	
	  • For a mixed UNIX/Windows cluster, UNIX/Windows user account mapping must be enabled.
	  • For a cluster with a non-uniform user name space, between-host account mapping must be enabled.
	  • For a MultiCluster environment with a non-uniform user name space, cross-cluster user account mapping must be enabled.
	• User accounts must have the correct permissions to successfully run jobs.
	• An eexec that requires root privileges to run on UNIX or Linux, must be configured to run as the root user.

Limitations	• Only an esub invoked by bsub can change the job environment on the submission host. An esub invoked by bmod or brestart cannot change the environment. • Any esub messages provided to the user must be directed to standard error, not to standard output. Standard output from any esub is automatically passed to eexec. • An eexec can handle only one standard output stream from an esub as standard input to eexec. You must make sure that your eexec handles standard output from correctly if any esub writes to standard output. • The esub/eexec combination cannot handle daemon authentication. To configure daemon authentication, you must enable external authentication, which uses the eauth executable.

Configuration to enable job submission and execution controls

This feature is enabled by the presence of at least one esub or one eexec executable in the directory specified by the parameter LSF_SERVERDIR in lsf.conf. LSF does not include a default esub or eexec; you should write your own executables to meet the job requirements of your site.

Executable file	UNIX naming convention	Windows naming convention
esub	LSF_SERVERDIR/ esub.*application*	LSF_SERVERDIR \esub.*application*.exe
		LSF_SERVERDIR \esub.*application*.bat
eexec	LSF_SERVERDIR/eexec	LSF_SERVERDIR\eexec.exe
		LSF_SERVERDIR\eexec.bat

The name of your esub should indicate the application with which it runs. For example: esub.fluent.

Restriction:

The name esub.user is reserved. Do not use the name esub.user for an application-specific esub.

Valid file names contain only alphanumeric characters, underscores (_), and hyphens (-).

Once the LSF_SERVERDIR contains one or more esub executables, users can specify the esub executables associated with each job they submit. If an eexec exists in LSF_SERVERDIR, LSF invokes that eexec for all jobs submitted to the cluster.

The following esub executables are provided as separate packages, available from Platform Computing Inc. upon request:

- esub.openmpi : OpenMPI job submission
- esub.pvm: PVM job submission
- esub.poe : POE job submission
- esub.ls_dyna : LS-Dyna job submission
- esub.fluent : FLUENT job submission
- esub.afs or esub.dce: AFS or DCE security
- esub.lammpi LAM/MPI job submission
- esub.mpich_gm : MPICH-GM job submission
- esub.intelmpi : Intel®MPI job submission
- esub.bproc: Beowulf Distributed Process Space (BProc) job submission
- esub.mpich2: MPICH2 job submission
- esub.mpichp4: MPICH-P4 job submission
- esub.mvapich: MVAPICH job submission
- esub.tv, esub.tvlammpi, esub.tvmpich_gm, esub.tvpoe: TotalView®debugging for various MPI applications.

Environment variables used by esub

When you write an esub, you can use the following environment variables provided by LSF for the esub execution environment:

LSB_SUB_PARM_FILE

Points to a temporary file that LSF uses to store the bsub options entered in the command line. An esub reads this file at job submission and either accepts the values,

changes the values, or rejects the job. Job submission options are stored as name-value pairs on separate lines with the format option_name=value.

For example, if a user submits the following job,

bsub -q normal -x -P myproject -R "rlm rusage[mem=100]" -n 90 myjob

The LSB_SUB_PARM_FILE contains the following lines:

```
LSB_SUB_QUEUE="normal"
LSB_SUB_EXLUSIVE=Y
LSB_SUB_RES_REQ="rlm usage[mem=100]"
LSB_SUB_PROJECT_NAME="myproject"
LSB_SUB_COMMAND_LINE="myjob"
LSB_SUB_NUM_PROCESSORS=90
LSB_SUB_MAX_NUM_PROCESSORS=90
```

An esub can change any or all of the job options by writing to the file specified by the environment variable LSB_SUB_MODIFY_FILE.

The temporary file pointed to by LSB_SUB_PARM_FILE stores the following information:

Option	bsub or bmod option	data type	Description
LSB_SUB_ADDITIONAL	-a	string	String that contains the application name or names of the esub executables requested by the user. **Restriction:** This is the only option that an esub cannot change or add at job submission.
LSB_SUB_BEGIN_TIME	-b	integer	Begin time, in seconds since 00:00:00 GMT, Jan. 1, 1970
LSB_SUB_CHKPNT_DIR	-k	string	Checkpoint directory The file path of the checkpoint directory can contain up to 4000 characters for UNIX and Linux, or up to 255 characters for Windows, including the directory and file name.
LSB_SUB_COMMAND_LINE	bsub job command argument	string	LSB_SUB_COMMANDNAME must be set in lsf.conf to enable esub to use this variable.

Option	bsub or bmod option	data type	Description
LSB_SUB_CHKPNT_PERIOD	-k	integer	Checkpoint period
LSB_SUB_DEPEND_COND	-w	string	Dependency condition
LSB_SUB_ERR_FILE	-e, -eo	string	Standard error file name
LSB_SUB_EXCLUSIVE	-x	boolean	Exclusive execution, specified by "Y"
LSB_SUB_EXTSCHED_PARAM	-ext	string	External scheduler options
LSB_SUB_HOLD	-H	boolean	Hold job
LSB_SUB_HOST_SPEC	-c or -w	string	Host specifier, limits the CPU time or RUN time.
LSB_SUB_HOSTS	-m	string	List of requested execution host names
LSB_SUB_IN_FILE	-i, -io	string	Standard input file name
LSB_SUB_INTERACTIVE	-I	boolean	Interactive job, specified by "Y"
LSB_SUB_LOGIN_SHELL	-L	string	Login shell
LSB_SUB_JOB_NAME	-J	string	Job name

Option	bsub or bmod option	data type	Description
LSB_SUB_JOB_WARNING_AC TION	-wa	string	Job warning action
LSB_SUB_JOB_ACTION_WAR NING_TIME	-wt	integer	Job warning time period
LSB_SUB_MAIL_USER	-u	string	Email address to which LSF sends job-related messages
LSB_SUB_MAX_NUM_PROCE SSORS	-n	integer	Maximum number of processors requested
LSB_SUB_MODIFY	bmod	boolean	Indicates that bmod invoked esub, specified by "Y".
LSB_SUB_MODIFY_ONCE	bmod	boolean	Indicates that the job options specified at job submission have already been modified by bmod, and that bmod is invoking esub again, specified by "Y".
LSB_SUB_NOTIFY_BEGIN	-B	boolean	LSF sends an email notification when the job begins, specified by "Y".
LSB_SUB_NOTIFY_END	-N	boolean	LSF sends an email notification when the job ends, specified by "Y".
LSB_SUB_NUM_PROCESSOR S	-n	integer	Minimum number of processors requested.
LSB_SUB_OTHER_FILES	bmod -f	integer	Indicates the number of files to be transferred. The value is SUB_RESET if bmod is being used to reset the number of files to be transferred. The file path of the directory can contain up to 4094 characters for UNIX and Linux, or up to 255 characters for Windows, including the director and file name.

Option	bsub or bmod option	data type	Description
LSB_SUB_OTHER_FILES_ *number*	bsub -f	integer	The *number* indicates the particular file transfer value in the specified file transfer expression. For example, for bsub -f "a > b" -f "c < d", the following would be defined: LSB_SUB_OTHER_FILES=2 LSB_SUB_OTHER_FILES_0="a > b" LSB_SUB_OTHER_FILES_1="c < d"
LSB_SUB_OUT_FILE	-o, -oo	string	Standard output file name.
LSB_SUB_PRE_EXEC	-E	string	Pre-execution command. The file path of the directory can contain up to 4094 characters for UNIX and Linux, or up to 255 characters for Windows, including the directory and file name.
LSB_SUB_PROJECT_NAME	-P	string	Project name.
LSB_SUB_PTY	-Ip	boolean	An interactive job with PTY support, specified by "Y"
LSB_SUB_PTY_SHELL	-Is	boolean	An interactive job with PTY shell support, specified by "Y"
LSB_SUB_QUEUE	-q	string	Submission queue name

Option	bsub or bmod option	data type	Description
LSB_SUB_RERUNNABLE	-r	boolean	"Y" specifies a rerunnable job "N" specifies a nonrerunnable job (specified with bsub -rn). The job is not rerunnable even it was submitted to a rerunable queue or application profile For bmod -rn, the value is SUB_RESET.
LSB_SUB_RES_REQ	-R	string	Resource requirement string—*does not* support multiple resource requirement strings
LSB_SUB_RESTART	`brestart`	boolean	"Y" indicates to esub that the job options are associated with a restarted job.
LSB_SUB_RESTART_FORCE	`brestart -f`	boolean	"Y" indicates to esub that the job options are associated with a forced restarted job.
LSB_SUB_RLIMIT_CORE	-C	integer	Core file size limit
LSB_SUB_RLIMIT_CPU	-c	integer	CPU limit
LSB_SUB_RLIMIT_DATA	-D	integer	Data size limit For AIX, if the XPG_SUS_ENV=ON environment variable is set in the user's environment before the process is executed and a process attempts to set the limit lower than current usage, the operation fails with errno set to EINVAL. If the XPG_SUS_ENV environment variable is not set, the operation fails with errno set to EFAULT.
LSB_SUB_RLIMIT_FSIZE	-F	integer	File size limit
LSB_SUB_RLIMIT_PROCESS	-p	integer	Process limit

Feature: External job submission and execution controls

Option	bsub or bmod option	data type	Description
LSB_SUB_RLIMIT_RSS	-M	integer	Resident size limit
LSB_SUB_RLIMIT_RUN	-W	integer	Wall-clock run limit
LSB_SUB_RLIMIT_STACK	-S	integer	Stack size limit
LSB_SUB_RLIMIT_THREAD	-T	integer	Thread limit
LSB_SUB_TERM_TIME	-t	integer	Termination time, in seconds, since 00:00:00 GMT, Jan. 1, 1970
LSB_SUB_TIME_EVENT	-wt	string	Time event expression
LSB_SUB_USER_GROUP	-G	string	User group name
LSB_SUB_WINDOW_SIG	-s	boolean	Window signal number
LSB_SUB2_JOB_GROUP	-g	string	Submits a job to a job group
LSB_SUB2_LICENSE_PROJECT	-Lp	string	LSF License Scheduler project name
LSB_SUB2_IN_FILE_SPOOL	-is	string	Spooled input file name
LSB_SUB2_JOB_CMD_SPOOL	-Zs	string	Spooled job command file name
LSB_SUB2_JOB_PRIORITY	-sp	integer	Job priority For bmod -spn, the value is SUB_RESET.

Option	bsub or bmod option	data type	Description
LSB_SUB2_SLA	-sla	string	SLA scheduling options
LSB_SUB2_USE_RSV	-U	string	Advance reservation ID
LSB_SUB3_ABSOLUTE_PRIORITY	bmod -aps bmod -apsn	string	For bmod -aps, the value equal to the APS string given. For bmod -apsn, the value is SUB_RESET.
LSB_SUB3_AUTO_RESIZABLE	-ar	boolean	Job autoresizable attribute. LSB_SUB3_AUTO_RESIZABLE=Y if bsub -ar or bmod -ar is specified. LSB_SUB3_AUTO_RESIABLE=SUB_RESET if bmod -arn is used.
LSB_SUB3_APP	-app	string	Application profile name For bmod -appn, the value is SUB_RESET.
LSB_SUB3_CWD	-cwd	string	Current working directory
LSB_SUB3_ INIT_CHKPNT_PERIOD	-k init	integer	Initial checkpoint period
LSB_SUB_INTERACTIVE LSB_SUB3_INTERACTIVE_SSH	bsub -IS	boolean	The session of the interactive job is encrypted with SSH.
LSB_SUB_INTERACTIVE LSB_SUB_PTY LSB_SUB3_INTERACTIVE_SSH	bsub –ISp	boolean	If LSB_SUB_INTERACTIVE is specified by "Y", LSB_SUB_PTY is specified by "Y", and LSB_SUB3_INTERACTIVE_SSH is specified by "Y", the session of interactive job with PTY support is encrypted by SSH.

Option	bsub or bmod option	data type	Description
LSB_SUB_INTERACTIVE LSB_SUB_PTY LSB_SUB_PTY_SHELL LSB_SUB3_INTERACTIVE_SSH	bsub –ISs	boolean	If LSB_SUB_INTERACTIVE is specified by "Y", LSB_SUB_PTY is specified by "Y", LSB_SUB_PTY_SHELL is specified by "Y", and LSB_SUB3_INTERACTIVE_SSH is specified by "Y", the session of interactive job with PTY shell support is encrypted by SSH.
LSB_SUB3_JOB_REQUEUE	-Q	string	String format parameter containing the job requeue exit values For bmod -Qn, the value is SUB_RESET.
LSB_SUB3_MIG	-mig -mign	integer	Migration threshold
LSB_SUB3_POST_EXEC	-Ep	string	Run the specified post-execution command on the execution host after the job finishes. The file path of the directory can contain up to 4094 characters for UNIX and Linux, or up to 255 characters for Windows, including the directory and file name.
LSB_SUB3_RESIZE_NOTIFY_CMD	-rnc	string	Job resize notification command. LSB_SUB3_RESIZE_NOTIFY_CMD=*<cmd>* if $bsub\ -rnc$ or $bmod\ -rnc$ is specified. LSB_SUB3_RESIZE_NOTIFY_CMD=SUB_RESET if $bmod\ -rnc$ is used.
LSB_SUB3_RUNTIME_ESTIMATION	-We	integer	Runtime estimate
LSB_SUB3_RUNTIME_ESTIMATION_ACC	-We+	integer	Runtime estimate that is the accumulated run time plus the runtime estimate
LSB_SUB3_RUNTIME_ESTIMATION_PERC	-Wep	integer	Runtime estimate in percentage of completion

Option	bsub or bmod option	data type	Description
LSB_SUB3_USER_SHELL_LIMITS	-ul	boolean	Pass user shell limits to execution host
LSB_SUB_INTERACTIVE LSB_SUB3_XJOB_SSH	bsub -IX	boolean	If both are set to "Y", the session between the X-client and X-server as well as the session between the execution host and submission host are encrypted with SSH.

LSB_SUB_MODIFY_FILE

Points to the file that esub uses to modify the bsub job option values stored in the LSB_SUB_PARM_FILE. You can change the job options by having your esub write the new values to the LSB_SUB_MODIFY_FILE in any order, using the same format shown for the LSB_SUB_PARM_FILE. The value SUB_RESET, integers, and boolean values do not require quotes. String parameters must be entered with quotes around each string, or space-separated series of strings.

When your esub runs at job submission, LSF checks the LSB_SUB_MODIFY_FILE and applies changes so that the job runs with the revised option values.

Restriction:

LSB_SUB_ADDITIONAL is the only option that an esub cannot change or add at job submission.

LSB_SUB_MODIFY_ENVFILE

Points to the file that esub uses to modify the user environment variables with which the job is submitted (not specified by bsub options). You can change these environment variables by having your esub write the values to the LSB_SUB_MODIFY_ENVFILE in any order, using the format variable_name=value, or variable_name="string".

LSF uses the LSB_SUB_MODIFY_ENVFILE to change the environment variables on the submission host. When your esub runs at job submission, LSF checks the LSB_SUB_MODIFY_ENVFILE and applies changes so that the job is submitted with the new environment variable values. LSF associates the new user environment with the job so that the job runs on the execution host with the new user environment.

LSB_SUB_ABORT_VALUE

Indicates to LSF that a job should be rejected. For example, if you want LSF to reject a job, your esub should contain the line

```
exit $LSB_SUB_ABORT_VALUE
```

Restriction:

When an esub exits with the LSB_SUB_ABORT_VALUE,
esub must not write to LSB_SUB_MODIFY_FILE or to
LSB_SUB_MODIFY_ENVFILE.

If multiple esubs are specified and one of the esubs exits with a value of
LSB_SUB_ABORT_VALUE, LSF rejects the job without running the remaining
esubs and returns a value of LSB_SUB_ABORT_VALUE.

LSB_INVOKE_CMD

Specifies the name of the LSF command that most recently invoked an external
executable.

Environment variables used by eexec

When you write an eexec, you can use the following environment variables in addition to all
user-environment or application-specific variables.

LS_EXEC_T

Indicates the stage or type of job execution. LSF sets LS_EXEC_T to:

- START at the beginning of job execution
- END at job completion
- CHKPNT at job checkpoint start

LS_JOBPID

Stores the process ID of the LSF process that invoked eexec. If eexec is intended to
monitor job execution, eexec must spawn 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.

Job submission and execution controls behavior

The following examples illustrate how customized esub and eexec executables can control
job submission and execution.

Validating job submission parameters using esub

When a user submits a job using bsub-P, LSF accepts any project name entered by the user
and associates that project name with the job. This example shows an esub that supports
project-based accounting by enforcing the use of valid project names for jobs submitted by
users who are eligible to charge to those projects. If a user submits a job to any project other
than proj1 or proj2, or if the user name is not user1 or user2, LSF rejects the job based on the
exit value of LSB_SUB_ABORT_VALUE.

```
#!/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_SUB_PROJECT_NAME != "proj1" -o $LSB_SUB_PROJECT_NAME != "proj2" ];
then
 echo "Incorrect project name specified"
   exit $LSB_SUB_ABORT_VALUE
fi
USER=`whoami`
if [ $LSB_SUB_PROJECT_NAME="proj1" ]; then
# Only user1 and user2 can charge to proj1
 if [$USER != "user1" -a $USER != "user2" ]; then
```

```
            echo "You are not allowed to charge to this project"
            exit $LSB_SUB_ABORT_VALUE
        fi
fi
```

Changing job submission parameters using esub

The following example shows an esub that modifies job submission options and environment variables based on the user name that submits a job. This esub writes the changes to LSB_SUB_MODIFY_FILE for userA and to LSB_SUB_MODIFY_ENVFILE for userB. LSF rejects all jobs submitted by userC without writing to either file:

```
#!/bin/sh
. $LSB_SUB_PARM_FILE
# Redirect stderr to stdout so echo can be used for error messages exec 1>&2
USER=`whoami`
# Make sure userA is using the right queue queueA
if [ $USER="userA" -a $LSB_SUB_QUEUE != "queueA" ]; then
    echo "userA has submitted a job to an incorrect queue"
    echo "...submitting to queueA"
    echo 'LSB_SUB_QUEUE="queueA"' > $LSB_SUB_MODIFY_FILE
fi
# Make sure userB is using the right shell (/bin/sh)
if [ $USER="userB" -a $SHELL != "/bin/sh" ]; then
    echo "userB has submitted a job using $SHELL"
    echo "...using /bin/sh instead"
    echo 'SHELL="/bin/sh"' > $LSB_SUB_MODIFY_ENVFILE
fi
# Deny userC the ability to submit a job
if [ $USER="userC" ]; then
    echo "You are not permitted to submit a job."
    exit $LSB_SUB_ABORT_VALUE
 fi
```

Monitoring the execution environment using eexec

This example shows how you can use an eexec to monitor job execution:

```
#!/bin/sh
# eexec
# Example script to monitor the number of jobs executing through RES.
# This script works in cooperation with an elim that counts the
# number of files in the TASKDIR directory. Each RES process on a host
# will have a file in the TASKDIR directory.
# Don't want to monitor lsbatch jobs.
if [ "$LSB_JOBID" != "" ] ; then
    exit 0
fi
TASKDIR="/tmp/RES_dir"
# directory containing all the task files
#for the host.
# you can change this to whatever
# directory you wish, just make sure anyone
# has read/write permissions.
# if TASKDIR does not exist create it
if [ "test -d $TASKDIR" != "0" ] ; then
    mkdir $TASKDIR > /dev/null 2>&1
fi
# Need to make sure LS_JOBPID, and USER are defined
# exit normally
if [ "test -z $LS_JOBPID"="0" ] ; then
    exit 0
elif [ "test -z $USER" =  "0" ] ; then
    exit 0
fi
taskFile="$TASKDIR/$LS_JOBPID.$USER"
# Fork grandchild to stay around for the duration of the task
touch $taskFile >/dev/null 2>&1
(
        (while : ;
        do
```

```
                              kill -0 $LS_JOBPID >/dev/null 2>&1
                              if [ $? -eq 0 ] ; then
                                      sleep 10  # this is the poll interval
                                                # increase it if you want but
                                                 # see the elim for its
                                                # corresponding update interval
                       else
                                      rm $taskFile >/dev/null 2>&1
                                      exit 0
                              fi
                  done) &
        ) &
        wait
```

Passing data between esub and eexec

A combination of esub and eexec executables can be used to pass AFS/DCE tokens from the submission host to the execution host. LSF passes data from the standard output of esub to the standard input of eexec. A daemon wrapper script can be used to renew the tokens.

Configuration to modify job submission and execution controls

There are configuration parameters that modify various aspects of job submission and execution controls behavior by:

- Defining a mandatory esub that applies to all jobs in the cluster
- Specifying the eexec user account (UNIX only)

Configuration to define a mandatory esub

Configuration file	Parameter and syntax	Behavior
lsf.conf	LSB_ESUB_METHOD="*esub_application* [*esub_application*] ..."	• The specified esub or esubs run for all jobs submitted to the cluster, in addition to any esub specified by the user in the command line • For example, to specify a mandatory esub named esub.fluent, define LSB_ESUB_METHOD=fluent

Configuration to specify the eexec user account

The eexec executable runs under the submission user account. You can modify this behavior for UNIX hosts by specifying a different user account.

Configuration file	Parameter and syntax	Behavior
lsf.sudoers	LSF_EEXEC_USER= *user_name*	• Changes the user account under which eexec runs

Job submission and execution controls commands

Commands for submission

Command	Description
bsub -a *esub_application* [*esub_application*] ...	• Specifies one or more esub executables to run at job submission • For example, to specify the esub named esub.fluent, use **bsub -a fluent** • LSF runs any esub executables defined by LSB_ESUB_METHOD, followed by the executable named "esub" if it exists in LSF_SERVERDIR, followed by the esub executables specified by the -a option • LSF runs eexec if an executable file with that name exists in LSF_SERVERDIR
brestart	• Restarts a checkpointed job and runs the esub executables specified when the job was submitted • LSF runs any esub executables defined by LSB_ESUB_METHOD, followed by the executable named "esub" if it exists in LSF_SERVERDIR, followed by the esub executables specified by the -a option • LSF runs eexec if an executable file with that name exists in LSF_SERVERDIR
lsrun	• Submits an interactive task; LSF runs eexec if an eexec executable exists in LSF_SERVERDIR • LSF runs eexec only at task startup (LS_EXEC_T=START)
lsgrun	• Submits an interactive task to run on a set of hosts; LSF runs eexec if an eexec executable exists in LSF_SERVERDIR • LSF runs eexec only at task startup (LS_EXEC_T=START)

Commands to monitor

Not applicable: There are no commands to monitor the behavior of this feature.

Commands to control

Command	Description
bmod -a *esub_application* [*esub_application*] ...	• Resubmits a job and changes the esubs previously associated with the job • LSF runs any esub executables defined by LSB_ESUB_METHOD, followed by the executable named "esub" if it exists in LSF_SERVERDIR, followed by the esub executables specified by the -a option of bmod • LSF runs eexec if an executable file with that name exists in LSF_SERVERDIR

Command	Description
bmod -an	• Dissociates from a job all esub executables that were previously associated with the job • LSF runs any esub executables defined by LSB_ESUB_METHOD, followed by the executable named "esub" if it exists in LSF_SERVERDIR • LSF runs eexec if an executable file with that name exists in LSF_SERVERDIR

Commands to display configuration

Command	Description
badmin showconf	• Displays all configured parameters and their values set in lsf.conf or ego.conf that affect mbatchd and sbatchd. Use a text editor to view other parameters in the lsf.conf or ego.conf configuration files. • In a MultiCluster environment, badmin showconf only displays the parameters of daemons on the local cluster.

Use a text editor to view the lsf.sudoers configuration file.

Feature: Job migration

The job migration feature enables you to move checkpointable and rerunnable jobs from one host to another. Job migration makes use of job checkpoint and restart so that a migrated checkpointable job restarts on the new host from the point at which the job stopped on the original host.

Contents

- About job migration
- Scope
- Configuration to enable job migration
- Job migration behavior
- Configuration to modify job migration
- Job migration commands

About job migration

Job migration refers to the process of moving a checkpointable or rerunnable job from one host to another. This facilitates load balancing by moving jobs from a heavily-loaded host to a lightly-loaded host.

You can initiate job migration on demand (bmi g) or automatically. To initiate job migration automatically, you configure a migration threshold at the host or queue level.

Default behavior (job migration not enabled)
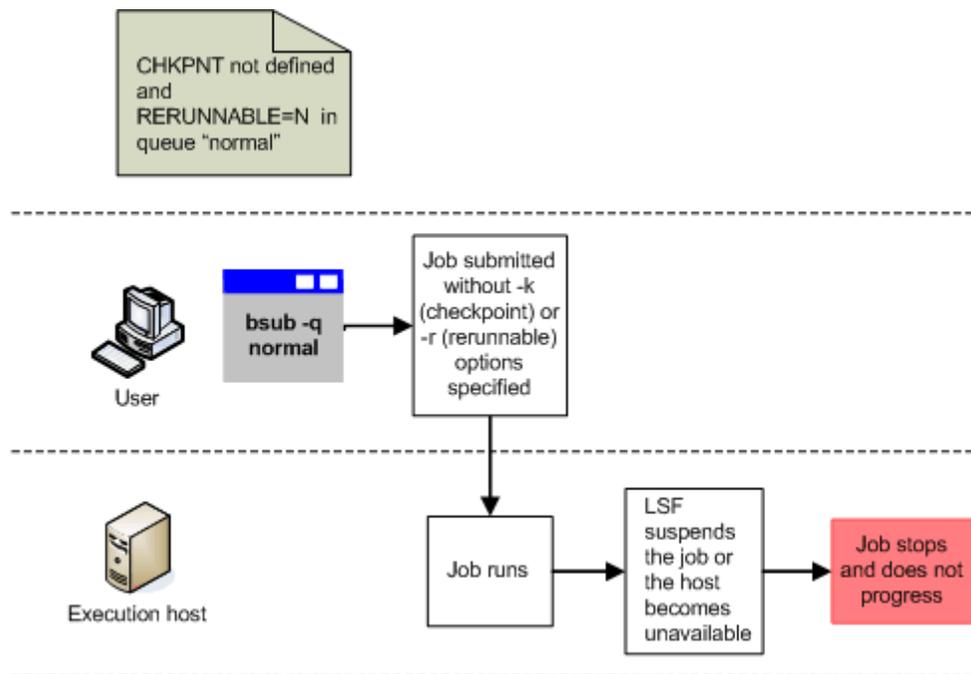
With automatic job migration enabled
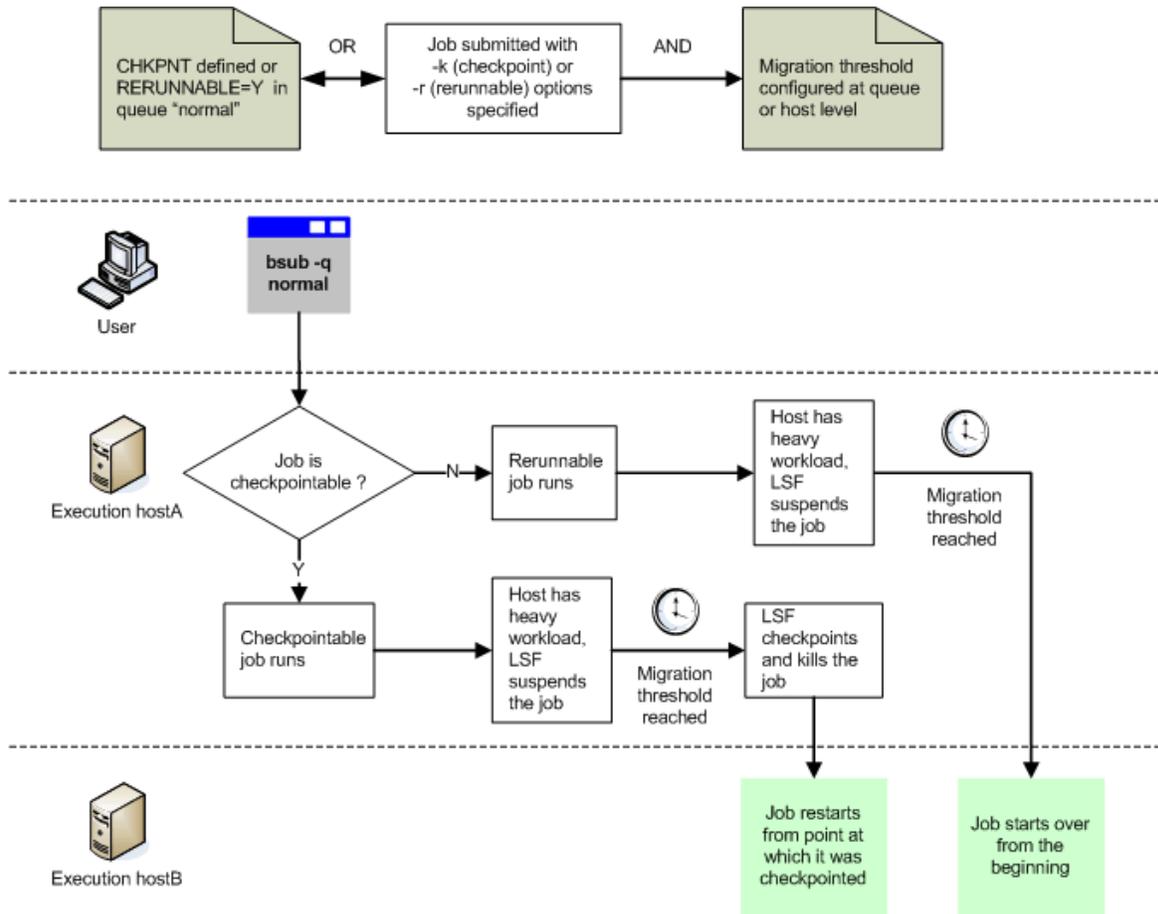
Scope

Applicability	Details
Operating system	• UNIX • Linux • Windows
Job types	• Non-interactive batch jobs submitted with bsub or bmod, including chunk jobs

Applicability	Details

Dependencies	• UNIX and Windows user accounts must be valid on all hosts in the cluster, or the correct type of account mapping must be enabled:
	
	   • For a mixed UNIX/Windows cluster, UNIX/Windows user account mapping must be enabled
	   • For a cluster with a non-uniform user name space, between-host account mapping must be enabled
	   • For a MultiCluster environment with a non-uniform user name space, cross-cluster user account mapping must be enabled
	
	• Both the original and the new hosts must:
	
	   • Be binary compatible
	   • Run the same dot version of the operating system for predictable results
	   • Have network connectivity and read/execute permissions to the checkpoint and restart executables (in LSF_SERVERDIR by default)
	   • Have network connectivity and read/write permissions to the checkpoint directory and the checkpoint file
	   • Have access to all files open during job execution so that LSF can locate them using an absolute path name

Configuration to enable job migration

The job migration feature requires that a job be made checkpointable or rerunnable at the job, application, or queue level. An LSF user can make a job

- Checkpointable, using bsub -k and specifying a checkpoint directory and checkpoint period, and an optional initial checkpoint period
- Rerunnable, using bsub -r

Feature: Job migration

Configuration file	Parameter and syntax	Behavior

lsb.queues	CHKPNT=*chkpnt_dir* [*chkpnt_period*]	• All jobs submitted to the queue are checkpointable.
		
		   • The specified checkpoint directory must already exist. LSF will not create the checkpoint directory.
		   • The user account that submits the job must have read and write permissions for the checkpoint directory.
		   • For the job to restart on another execution host, both the original and new hosts must have network connectivity to the checkpoint directory.
		• If the queue administrator specifies a checkpoint period, in minutes, LSF creates a checkpoint file every *chkpnt_period* during job execution.
		• If a user specifies a checkpoint directory and checkpoint period at the job level with bsub -k, the job-level values override the queue-level values.

	RERUNNABLE=Y	• If the execution host becomes unavailable, LSF reruns the job from the beginning on a different host.
lsb.applications	CHKPNT_DIR=*chkpnt_dir*	• Specifies the checkpoint directory for automatic checkpointing for the application. To enable automatic checkpoint for the application profile, administrators must specify a checkpoint directory in the configuration of the application profile. • If CHKPNT_PERIOD, CHKPNT_INITPERIOD or CHKPNT_METHOD was set in an application profile but CHKPNT_DIR was not set, a warning message is issued and and those settings are ignored. • The checkpoint directory is the directory where the checkpoint files are created. Specify an absolute path or a path relative to the current working directory for the job. Do not use environment variables in the directory path. • If checkpoint-related configuration is specified in both the queue and an application profile, the application profile setting overrides queue level configuration.
	CHKPNT_INITPERIOD= *init_chkpnt_period*	

Configuration file	Parameter and syntax	Behavior
	CHKPNT_PERIOD=*chkpnt_period*	
	CHKPNT_METHOD= *chkpnt_method*	

Configuration to enable automatic job migration

Automatic job migration assumes that if a job is system-suspended (SSUSP) for an extended period of time, the execution host is probably heavily loaded. Configuring a queue-level or host-level migration threshold lets the job to resume on another less loaded host, and reduces the load on the original host. You can use bmig at any time to override a configured migration threshold.

Configuration file	Parameter and syntax	Behavior
lsb.queues lsb.applications	MIG=*minutes*	- LSF automatically migrates jobs that have been in the SSUSP state for more than the specified number of minutes - Specify a value of 0 to migrate jobs immediately upon suspension - Applies to all jobs submitted to the queue - Job-level command line migration threshold (bsub - mig) overrides threshold configuration in application profile and queue. Application profile configuration overrides queue level configuration.
lsb.hosts	HOST_NAME MIG *host_name* *minutes*	- LSF automatically migrates jobs that have been in the SSUSP state for more than the specified number of minutes - Specify a value of 0 to migrate jobs immediately upon suspension - Applies to all jobs running on the host

Note:

When a host migration threshold is specified, and is lower than the value for the job, the queue, or the application, the host value is used.

Job migration behavior

LSF migrates a job by performing the following actions:

1. Stops the job if it is running
2. Checkpoints the job if the job is checkpointable
3. Kills the job on the current host
4. Restarts or reruns the job on the first available host, bypassing all pending jobs

Configuration to modify job migration

You can configure LSF to requeue a migrating job rather than restart or rerun the job.

Configuration file	Parameter and syntax	Behavior
lsf.conf	LSB_MIG2PEND=1	• LSF requeues a migrating job rather than restarting or rerunning the job • LSF requeues the job as pending in order of the original submission time and priority • In a MultiCluster environment, LSF ignores this parameter
	LSB_REQUEUE_TO_BOTTOM=1	• When LSB_MIG2PEND=1, LSF requeues a migrating job to the bottom of the queue, regardless of the original submission time and priority • If the queue defines APS scheduling, migrated jobs keep their APS information and compete with other pending jobs based on the APS value

Job migration commands

Commands for submission

Job migration applies to checkpointable or rerunnable jobs submitted with a migration threshold, or that have already started and are either running or suspended.

Command	Description
bsub -mig *migration_threshold*	• Submits the job with the specified migration threshold for checkpointable or rerunnable jobs. Enables automatic job migration and specifies the migration threshold, in minutes. A value of 0 (zero) specifies that a suspended job should be migrated immediately. • Command-level job migration threshold overrides application profile and queue-level settings. • Where a host migration threshold is also specified, and is lower than the job value, the host value is used.

Commands to monitor

Command	Description
bhist -l	• Displays the actions that LSF took on a completed job, including migration to another host
bjobs -l	• Displays information about pending, running, and suspended jobs

Commands to control

Command	Description
bmig	• Migrates one or more running jobs from one host to another. The jobs must be checkpointable or rerunnable • Checkpoints, kills, and restarts one or more checkpointable jobs—bmig combines the functionality of the bchkpnt and brestart commands into a single command • Migrates the job on demand even if you have configured queue-level or host-level migration thresholds • When absolute job priority scheduling (APS) is configured in the queue, LSF schedules migrated jobs before pending jobs—for migrated jobs, LSF maintains the existing job priority
bmod -mig *migration_threshold* | -mign	• Modifies or cancels the migration threshold specified at job submission for checkpointable or rerunnable jobs. Enables or disables automatic job migration and specifies the migration threshold, in minutes. A value of 0 (zero) specifies that a suspended job should be migrated immediately. • Command-level job migration threshold overrides application profile and queue-level settings. • Where a host migration threshold is also specified, and is lower than the job value, the host value is used.

Commands to display configuration

Command	Description
bhosts -l	• Displays information about hosts configured in lsb.hosts, including the values defined for migration thresholds in minutes
bqueues -l	• Displays information about queues configured in lsb.queues, including the values defined for migration thresholds **Note:** The bqueues command displays the migration threshold in seconds—the lsb.queues MIG parameter defines the migration threshold in minutes.
badmin showconf	• Displays all configured parameters and their values set in lsf.conf or ego.conf that affect mbatchd and sbatchd. Use a text editor to view other parameters in the lsf.conf or ego.conf configuration files. • In a MultiCluster environment, badmin showconf only displays the parameters of daemons on the local cluster.

Feature: Job checkpoint and restart

The job checkpoint and restart feature enables you to stop jobs and then restart them from the point at which they stopped, which optimizes resource usage. LSF can periodically capture the state of a running job and the data required to restart it. This feature provides fault tolerance and allows LSF administrators and users to migrate jobs from one host to another to achieve load balancing.

Contents

- About job checkpoint and restart
- Scope
- Configuration to enable job checkpoint and restart
- Job checkpoint and restart behavior
- Configuration to modify job checkpoint and restart
- Job checkpoint and restart commands

## About job checkpoint and restart

Checkpointing enables LSF users to restart a job on the same execution host or to migrate a job to a different execution host. LSF controls checkpointing and restart by means of interfaces named echkpnt and erestart. By default, when a user specifies a checkpoint directory using `bsub -k` or `bmod -k` or submits a job to a queue that has a checkpoint directory specified, echkpnt sends checkpoint instructions to an executable named `echkpnt.default`.

When LSF checkpoints a job, the `echkpnt` interface creates a checkpoint file in the directory *checkpoint_dir/ job_ID*, and then checkpoints and resumes the job. The job continues to run, even if checkpointing fails.

When LSF restarts a stopped job, the `erestart` interface recovers job state information from the checkpoint file, including information about the execution environment, and restarts the job from the point at which the job stopped. At job restart, LSF

1. Resubmits the job to its original queue and assigns a new job ID
2. Dispatches the job when a suitable host becomes available (not necessarily the original execution host)
3. Re-creates the execution environment based on information from the checkpoint file
4. Restarts the job from its most recent checkpoint

Default behavior (job checkpoint and restart not enabled)
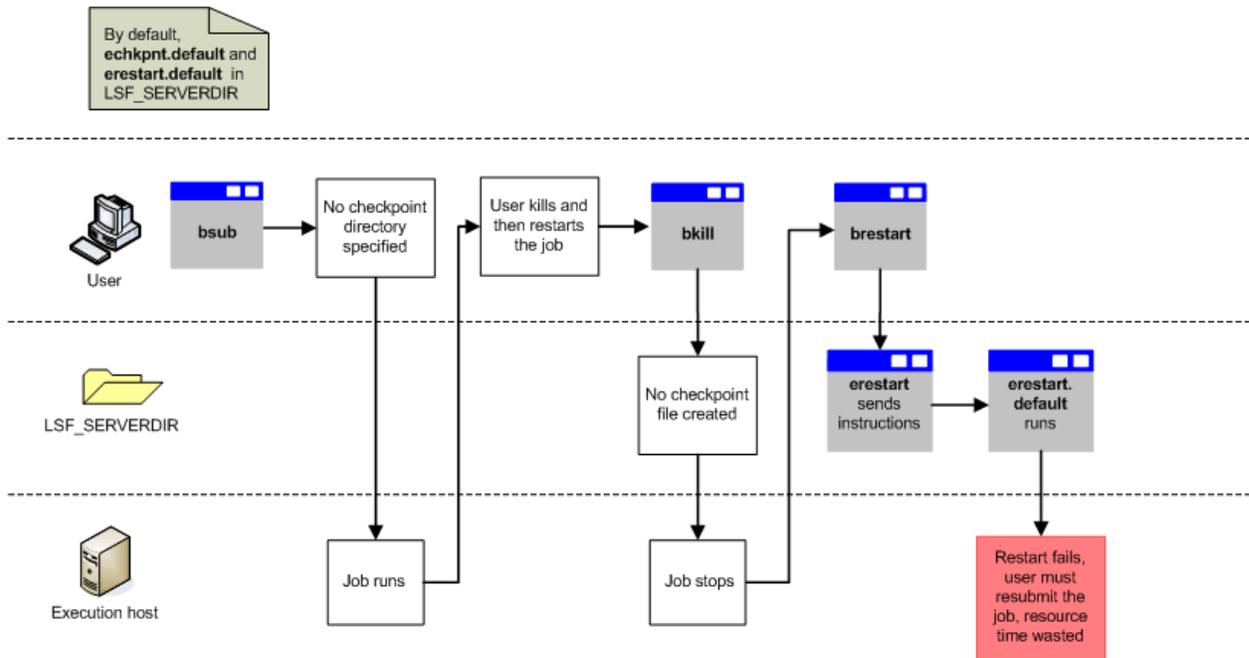
With job checkpoint and restart enabled
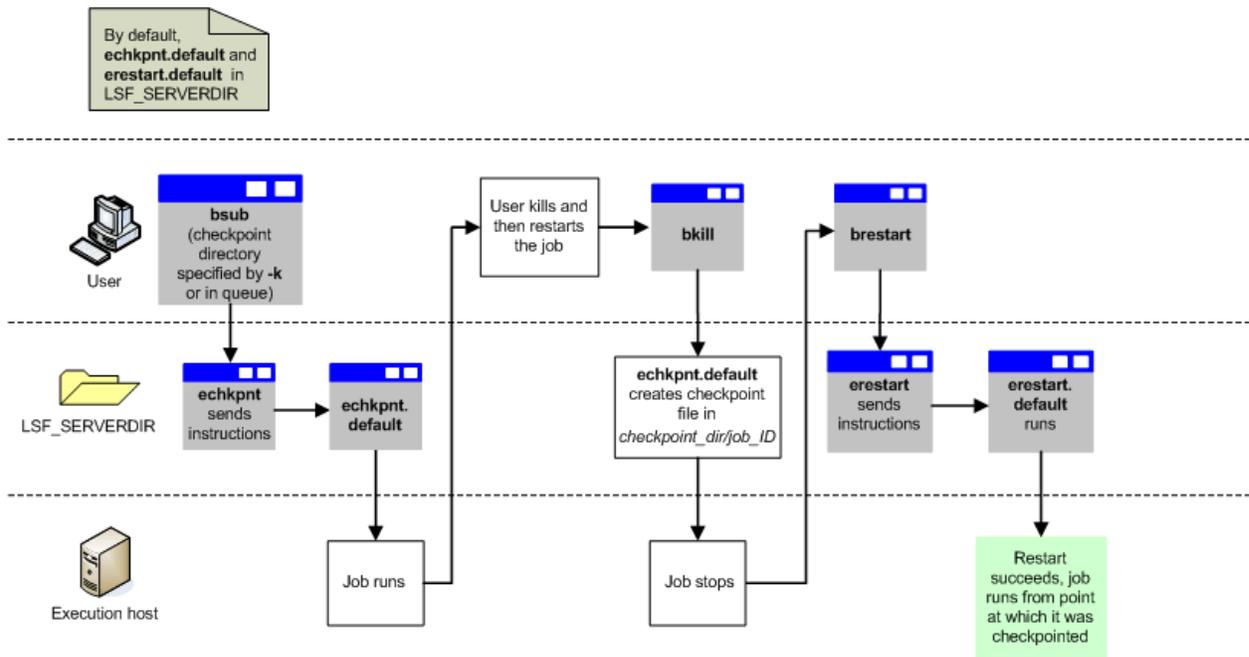
Kernel-level checkpoint and restart

The operating system provides checkpoint and restart functionality that is transparent to your applications and enabled by default. To implement job checkpoint and restart at the kernel level, the LSF echkpnt and erestart executables invoke operating system-specific calls.

LSF uses the default executables echkpnt.default and erestart.default for kernel-level checkpoint and restart.

User-level checkpoint and restart

For systems that do not support kernel-level checkpoint and restart, LSF provides a job checkpoint and restart implementation that is transparent to your applications and does not require you to rewrite code. User-level job checkpoint and restart is enabled by linking your application files to the LSF checkpoint libraries in LSF_LIBDIR. LSF uses the default executables echkpnt.default and erestart.default for user-level checkpoint and restart.

Application-level checkpoint and restart

Different applications have different checkpointing implementations that require the use of customized external executables (echkpnt.*application* and erestart.*application*). Application-level checkpoint and restart enables you to configure LSF to use specific echkpnt.*application* and erestart.*application* executables for a job, queue, or cluster. You can write customized checkpoint and restart executables for each application that you use.

LSF uses a combination of corresponding checkpoint and restart executables. For example, if you use echkpnt.fluent to checkpoint a particular job, LSF will use erestart.fluent to restart the checkpointed job. You cannot override this behavior or configure LSF to use a specific restart executable.

Scope

Applicability	Details

Operating system	• Kernel-level checkpoint and restart using the LSF checkpoint libraries works only with supported operating system versions and architecture for:
	
	  • SGI IRIX 6.4 and later
	  • SGI Altix ProPack 3 and later

Job types	• Non-interactive batch jobs submitted with bsub or bmod • Non-interactive batch jobs, including chunk jobs, checkpointed with bchkpnt • Non-interactive batch jobs migrated with bmig • Non-interactive batch jobs restarted with brestart

Applicability	Details
Dependencies	• UNIX and Windows user accounts must be valid on all hosts in the cluster, or the correct type of account mapping must be enabled. • For a mixed UNIX/Windows cluster, UNIX/Windows user account mapping must be enabled. • For a cluster with a non-uniform user name space, between-host account mapping must be enabled. • For a MultiCluster environment with a non-uniform user name space, cross-cluster user account mapping must be enabled. • The checkpoint and restart executables run under the user account of the user who submits the job. User accounts must have the correct permissions to • Successfully run executables located in LSF_SERVERDIR or LSB_ECHKPNT_METHOD_DIR • Write to the checkpoint directory • The `erestart.` *application* executable must have access to the original command line used to submit the job. • For user-level checkpoint and restart, you must have access to your application object (`.o`) files. • To allow restart of a checkpointed job on a different host than the host on which the job originally ran, both the original and the new hosts must: • Be binary compatible • Run the same dot version of the operating system for predictable results • Have network connectivity and read/execute permissions to the checkpoint and restart executables (in LSF_SERVERDIR by default) • Have network connectivity and read/write permissions to the checkpoint directory and the checkpoint file • Have access to all files open during job execution so that LSF can locate them using an absolute path name
Limitations	• `bmod` cannot change the `echkpnt` and `erestart` executables associated with a job. • Linux 32, AIX, and HP platforms with NFS (network file systems), checkpoint directories (including path and file name) must be shorter than 1000 characters. • Linux 64 with NFS (network file systems), checkpoint directories (including path and file name) must be shorter than 2000 characters.

Configuration to enable job checkpoint and restart

The job checkpoint and restart feature requires that a job be made checkpointable at the job or queue level. LSF users can make jobs checkpointable by submitting jobs using `bsub -k` and specifying a checkpoint directory. Queue administrators can make all jobs in a queue checkpointable by specifying a checkpoint directory for the queue.

Configuration file	Parameter and syntax	Behavior

lsb.queues	CHKPNT=*chkpnt_dir* [*chkpnt_period*]	• All jobs submitted to the queue are checkpointable. LSF writes the checkpoint files, which contain job state information, to the checkpoint directory. The checkpoint directory can contain checkpoint files for multiple jobs.
		
		  • The specified checkpoint directory must already exist. LSF will not create the checkpoint directory.
		  • The user account that submits the job must have read and write permissions for the checkpoint directory.
		  • For the job to restart on another execution host, both the original and new hosts must have network connectivity to the checkpoint directory.
		
		• If the queue administrator specifies a checkpoint period, in minutes, LSF creates a checkpoint file every *chkpnt_period* during job execution.
		
		Note:
		There is no default value for checkpoint period. You must specify a checkpoint period if you want to enable periodic checkpointing.
		
		• If a user specifies a checkpoint directory and checkpoint period at the job level with bsub -k, the job-level values override the queue-level values.
		• The file path of the checkpoint directory can contain up to 4000 characters for UNIX and Linux, or up to 255 characters for Windows, including the directory and file name.

lsb.application s		

Configuration to enable kernel-level checkpoint and restart

Kernel-level checkpoint and restart is enabled by default. LSF users make a job checkpointable by either submitting a job using bsub -k and specifying a checkpoint directory or by submitting a job to a queue that defines a checkpoint directory for the CHKPNT parameter.

Configuration to enable user-level checkpoint and restart

To enable user-level checkpoint and restart, you must link your application object files to the LSF checkpoint libraries provided in LSF_LIBDIR. You do not have to change any code within your application. For instructions on how to link application files, see the *Platform LSF Programmer's Guide.*

Configuration to enable application-level checkpoint and restart

Application-level checkpointing requires the presence of at least one echkpnt. *application* executable in the directory specified by the parameter LSF_SERVERDIR in lsf.conf. Each echkpnt. *application* must have a corresponding erestart. *application*.

Important:

The erestart.*application* executable must:

- Have access to the command line used to submit or modify the job
- Exit with a return value without running an application; the erestart interface runs the application to restart the job

Executable file	UNIX naming convention	Windows naming convention
echkpnt	LSF_SERVERDIR/echkpnt. *application*	LSF_SERVERDIR\echkpnt. *application*. exe
		LSF_SERVERDIR\echkpnt. *application*. bat
erestart	LSF_SERVERDIR/erestart. *application*	LSF_SERVERDIR \erestart. *application*. exe
		LSF_SERVERDIR \erestart. *application*. bat

Restriction:

The names echkpnt. default and erestart. default are reserved. Do not use these names for application-level checkpoint and restart executables.

Valid file names contain only alphanumeric characters, underscores (_), and hyphens (-).

For application-level checkpoint and restart, once the LSF_SERVERDIR contains one or more checkpoint and restart executables, users can specify the external checkpoint executable associated with each checkpointable job they submit. At restart, LSF invokes the corresponding external restart executable.

Requirements for application-level checkpoint and restart executables

- The executables must be written in C or Fortran.
- The directory/name combinations must be unique within the cluster. For example, you can write two different checkpoint executables with the name echkpnt. fluent and save them as LSF_SERVERDIR/echkpnt. fluent and my_execs/echkpnt. fluent. To run checkpoint and restart executables from a directory other than LSF_SERVERDIR, you must configure the parameter LSB_ECHKPNT_METHOD_DIR in lsf.conf.
- Your executables must return the following values.

- An echkpnt.*application* must return a value of 0 when checkpointing succeeds and a non-zero value when checkpointing fails.
- The erestart interface provided with LSF restarts the job using a restart command that erestart.*application* writes to a file. The return value indicates whether erestart.*application* successfully writes the parameter definition LSB_RESTART_CMD=*restart_command* to the file *checkpoint_dir/job_ID/*.restart_cmd.
  - A non-zero value indicates that erestart.*application* failed to write to the .restart_cmd file.
  - A return value of 0 indicates that erestart.*application* successfully wrote to the .restart_cmd file, or that the executable intentionally did not write to the file.
- Your executables must recognize the syntax used by the echkpnt and erestart interfaces, which communicate with your executables by means of a common syntax.
  - echkpnt.*application* syntax:

    ```
    echkpnt [-c] [-f] [-k | -s] [-d checkpoint_dir] [-x] process_group_ID
    ```

    Restriction:

    The -k and -s options are mutually exclusive.
  - erestart.*application* syntax:

    ```
    erestart [-c] [-f] checkpoint_dir
    ```

Option or variable	Description	Operating systems
-c	Copies all files in use by the checkpointed process to the checkpoint directory.	Some, such as SGI systems running IRIX and Altix
-f	Forces a job to be checkpointed even under non-checkpointable conditions, which are specific to the checkpoint implementation used. This option could create checkpoint files that do not provide for successful restart.	Some, such as SGI systems running IRIX and Altix
-k	Kills a job after successful checkpointing. If checkpoint fails, the job continues to run.	All operating systems that LSF supports
-s	Stops a job after successful checkpointing. If checkpoint fails, the job continues to run.	Some, such as SGI systems running IRIX and Altix
-d *checkpoint_dir*	Specifies the checkpoint directory as a relative or absolute path.	All operating systems that LSF supports
-x	Identifies the cpr (checkpoint and restart) process as type HID. This identifies the set of processes to checkpoint as a process hierarchy (tree) rooted at the current PID.	Some, such as SGI systems running IRIX and Altix
process_group_ID	ID of the process or process group to checkpoint.	All operating systems that LSF supports

Job checkpoint and restart behavior

LSF invokes the echkpnt interface when a job is

- Automatically checkpointed based on a configured checkpoint period
- Manually checkpointed with `bchkpnt`
- Migrated to a new host with `bmig`

After checkpointing, LSF invokes the `erestart` interface to restart the job. LSF also invokes the `erestart` interface when a user

- Manually restarts a job using `brestart`
- Migrates the job to a new host using `bmig`

All checkpoint and restart executables run under the user account of the user who submits the job.

> **Note:**
>
> By default, LSF redirects standard error and standard output to `/dev/null` and discards the data.

Checkpoint directory and files

LSF identifies checkpoint files by the checkpoint directory and job ID. For example:

bsub -k my_dir
```
Job <123> is submitted to default queue <default>
```

LSF writes the checkpoint file to `my_dir/123`.

LSF maintains all of the checkpoint files for a single job in one location. When a job restarts, LSF creates both a new subdirectory based on the new job ID and a symbolic link from the old to the new directory. For example, when job 123 restarts on a new host as job 456, LSF creates `my_dir/456` and a symbolic link from `my_dir/123` to `my_dir/456`.

The file path of the checkpoint directory can contain up to 4000 characters for UNIX and Linux, or up to 255 characters for Windows, including the directory and file name.

Precedence of job, queue, application, and cluster-level checkpoint values

LSF handles checkpoint and restart values as follows:

1. *Checkpoint directory and checkpoint period*—values specified at the job level override values for the queue. Values specified in an application profile setting overrides queue level configuration.

    If checkpoint-related configuration is specified in the queue, application profile, and at job level:

    - Application-level and job-level parameters are merged. If the same parameter is defined at both job-level and in the application profile, the job-level value overrides the application profile value.
    - The merged result of job-level and application profile settings override queue-level configuration.
2. *Checkpoint and restart executables*—the value for *checkpoint_method* specified at the job level overrides the application-level CHKPNT_METHOD, and the cluster-level value for LSB_ECHKPNT_METHOD specified in `lsf.conf` or as an environment variable.
3. *Configuration parameters and environment variables*—values specified as environment variables override the values specified in `lsf.conf`

If the command line is…	And…	Then…
bsub -k "my_dir 240"	In `lsb.queues`, `CHKPNT=other_dir 360`	• LSF saves the checkpoint file to *my_dir/ job_ID* every 240 minutes
bsub -k "my_dir fluent"	In `lsf.conf`, `LSB_ECHKPNT_METHOD=myapp`	• LSF invokes `echkpnt.fluent` at job checkpoint and `erestart.fluent` at job restart
bsub -k "my_dir"	In `lsb.applications`, `CHKPNT_PERIOD=360`	• LSF saves the checkpoint file to *my_dir/ job_ID* every 360 minutes
bsub -k "240"	In `lsb.applications`, `CHKPNT_DIR=app_dir` `CHKPNT_PERIOD=360` In `lsb.queues`, `CHKPNT=other_dir`	• LSF saves the checkpoint file to *app_dir/ job_ID* every 240 minutes

Configuration to modify job checkpoint and restart

There are configuration parameters that modify various aspects of job checkpoint and restart behavior by:

- Specifying mandatory application-level checkpoint and restart executables that apply to all checkpointable batch jobs in the cluster
- Specifying the directory that contains customized application-level checkpoint and restart executables
- Saving standard output and standard error to files in the checkpoint directory
- Automatically checkpointing jobs before suspending or terminating them
- For Cray systems only, copying all open job files to the checkpoint directory

Configuration to specify mandatory application-level executables

You can specify mandatory checkpoint and restart executables by defining the parameter LSB_ECHKPNT_METHOD in `lsf.conf` or as an environment variable.

Configuration file	Parameter and syntax	Behavior
lsf.conf	LSB_ECHKPNT_METHOD="*echkpnt_application*"	• The specified echkpnt runs for all batch jobs submitted to the cluster. At restart, the corresponding erestart runs. • For example, if LSB_ECHKPNT_METHOD=fluent, at checkpoint, LSF runs echkpnt.fluent and at restart, LSF runs erestart.fluent. • If an LSF user specifies a different *echkpnt_application* at the job level using bsub -k or bmod -k, the job level value overrides the value in lsf.conf.

Configuration to specify the directory for application-level executables

By default, LSF looks for application-level checkpoint and restart executables in LSF_SERVERDIR. You can modify this behavior by specifying a different directory as an environment variable or in lsf.conf.

Configuration file	Parameter and syntax	Behavior
lsf.conf	LSB_ECHKPNT_METHOD_DIR=*path*	• Specifies the absolute path to the directory that contains the echkpnt.*application* and erestart.*application* executables • User accounts that run these executables must have the correct permissions for the LSB_ECHKPNT_METHOD_DIR directory.

Configuration to save standard output and standard error

By default, LSF redirects the standard output and standard error from checkpoint and restart executables to /dev/null and discards the data. You can modify this behavior by defining the parameter LSB_ECHKPNT_KEEP_OUTPUT as an environment variable or in lsf.conf.

Configuration file	Parameter and syntax	Behavior

`lsf.conf`	LSB_ECHKPNT_KEEP_OUTPUT=Y | y	• The stdout and stderr for `echkpnt.`*application* or `echkpnt.default` are redirected to *checkpoint_dir/job_ID/*
		
		  • `echkpnt.out`
		  • `echkpnt.err`
		
		• The stdout and stderr for `erestart.`*application* or `erestart.default` are redirected to *checkpoint_dir/job_ID/*
		
		  • `erestart.out`
		  • `erestart.err`

Configuration to checkpoint jobs before suspending or terminating them

LSF administrators can configure LSF at the queue level to checkpoint jobs before suspending or terminating them.

Configuration file	Parameter and syntax	Behavior
`lsb.queues`	JOB_CONTROLS=SUSPEND CHKPNT TERMINATE	• LSF checkpoints jobs before suspending or terminating them • When suspending a job, LSF checkpoints the job and then stops it by sending the SIGSTOP signal • When terminating a job, LSF checkpoints the job and then kills it

Configuration to copy open job files to the checkpoint directory

For hosts that use the Cray operating system, LSF administrators can configure LSF at the host level to copy all open job files to the checkpoint directory every time the job is checkpointed.

Configuration file	Parameter and syntax	Behavior
`lsb.hosts`	HOST_NAME CHKPNT *host_name* C	• LSF copies all open job files to the checkpoint directory when a job is checkpointed

Job checkpoint and restart commands

Commands for submission

Command	Description
bsub -k "*checkpoint_dir* [*checkpoint_period*] [method=*echkpnt_application*]"	• Specifies a relative or absolute path for the checkpoint directory and makes the job checkpointable. • If the specified checkpoint directory does not already exist, LSF creates the checkpoint directory. • If a user specifies a checkpoint period (in minutes), LSF creates a checkpoint file every *chkpnt_period* during job execution. • The command-line values for the checkpoint directory and checkpoint period override the values specified for the queue. • If a user specifies an *echkpnt_application*, LSF runs the corresponding restart executable when the job restarts. For example, for **bsub -k "my_dir method=fluent"** LSF runs echkpnt.fluent at job checkpoint and erestart.fluent at job restart. • The command-line value for *echkpnt_application* overrides the value specified by LSB_ECHKPNT_METHOD in lsf.conf or as an environment variable. Users can override LSB_ECHKPNT_METHOD and use the default checkpoint and restart executables by defining **method=default**.

Commands to monitor

Command	Description
bacct -l	• Displays accounting statistics for finished jobs, including termination reasons. TERM_CHKPNT indicates that a job was checkpointed and killed. • If JOB_CONTROL is defined for a queue, LSF does not display the result of the action.
bhist -l	• Displays the actions that LSF took on a completed job, including job checkpoint, restart, and migration to another host.
bjobs -l	• Displays information about pending, running, and suspended jobs, including the checkpoint directory, the checkpoint period, and the checkpoint method (either *application* or default).

Commands to control

Command	Description
bmod -k "*checkpoint_dir* [*checkpoint_period*] [method=*echkpnt_application*]"	• Resubmits a job and changes the checkpoint directory, checkpoint period, and the checkpoint and restart executables associated with the job.
bmod -kn	• Dissociates the checkpoint directory from a job, which makes the job no longer checkpointable.

| Command	Description
bchkpnt	• Checkpoints the most recently submitted checkpointable job. Users can specify particular jobs to checkpoint by including various bchkpnt options.
bchkpnt -p *checkpoint_period job_ID*	• Checkpoints a job immediately and changes the checkpoint period for the job.
bchkpnt -k *job_ID*	• Checkpoints a job immediately and kills the job.
bchkpnt -p 0 *job_ID*	• Checkpoints a job immediately and disables periodic checkpointing.
brestart	• Restarts a checkpointed job on the first available host.
brestart -m	• Restarts a checkpointed job on the specified host or host group.
bmig	• Migrates one or more running jobs from one host to another. The jobs must be checkpointable or rerunnable. • Checkpoints, kills, and restarts one or more checkpointable jobs.

Commands to display configuration

Command	Description
bqueues -l	• Displays information about queues configured in lsb.queues, including the values defined for checkpoint directory and checkpoint period. **Note:** The bqueues command displays the checkpoint period in seconds; the lsb.queues CHKPNT parameter defines the checkpoint period in minutes.
badmin showconf	• Displays all configured parameters and their values set in lsf.conf or ego.conf that affect mbatchd and sbatchd. Use a text editor to view other parameters in the lsf.conf or ego.conf configuration files. • In a MultiCluster environment, badmin showconf only displays the parameters of daemons on the local cluster.

Feature: Resizable Jobs

Enabling resizable jobs allows LSF to run a job with minimum and maximum slots requested and have it dynamically use the number of slots available at any given time.

By default, if a job specifies minimum and maximum slots requests (bsub -n min, max), LSF makes a one time allocation and schedules the job. You can configure resizable jobs, where LSF dispatches jobs as long as minimum slot request is satisfied. After the job successfully starts, LSF continues to schedule and allocate additional resources to satisfy the maximum slot request for the job. For example, a job asks for -n 4,32 processors. The Job starts to run and gets 20 slots at time t0. After that, LSF continues to allocate more job resources; for instance, 4 slots to jobs at time t1. Then, another 8 slots at time t2, which finally satisfies 32 slot requirement.
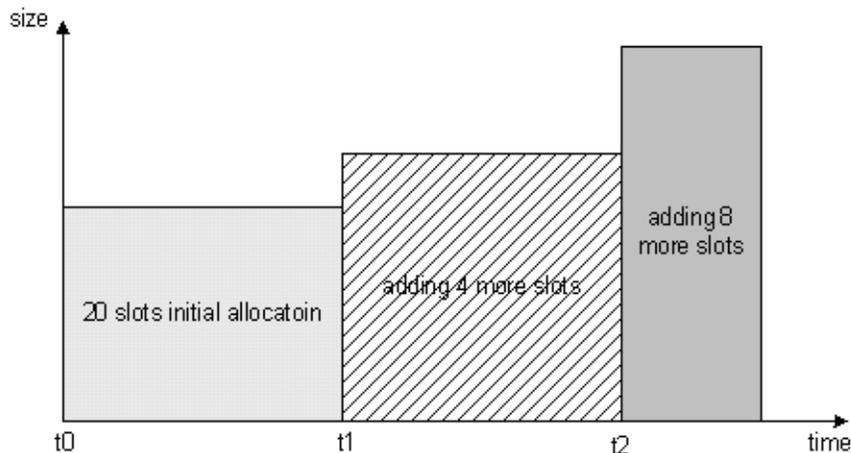
About resizable jobs

Resizable Job

A job whose job slot allocation can grow and shrink during its run time. The allocation change request may be triggered automatically or by the bresize command. For example, after the job starts, you can explicitly cancel resize allocation requests or have the job release idle resources back to the LSF.

Autoresizable job

A resizable job with a minimum and maximum slot request. LSF automatically schedules and allocates additional resources to satisfy job maximum request as the job runs.

For autoresizable jobs, LSF automatically calculates the pending allocation requests. The maximum pending allocation request is calculated based on the maximum number of requested slots minus the number of allocated slots. And the minimum pending allocation request is always 1. B ecause the job is running and its previous minimum request is already satisfied, LSF is able to allocate any number of additional slots to the running job. For instance, if job requests -n 4, 32, if LSF allocates 20 slots to the job initially, its active pending allocation request is 1 to 12. 1 is minimum slot request. 12 is maximum slot request. After LSF assigns another 4 slots, the pending allocation request is 1 to 8.

Pending allocation request

An additional resource request attached to a resizable job. Only running jobs can have pending allocation requests. At any given time, the job only has one allocation request.

LSF creates a new pending allocation request and schedules it after job physically starts on the remote host (after LSF receives the JOB_EXECUTE event from sbatchd) or notification successfully completes.

Notification command

A notification command is an executable that is invoked on the first execution host of a job in response to an allocation (grow or shrink) event. It can be used to inform the running application for allocation change. Due to the various implementations of applications, each resizable application may have its own notification command provided by the application developer.

The notification command runs under the same user ID environment, home, and working directory as the actual job. The standard input, output, and error of the program are redirected to the NULL device. If the notification command is not in the user's normal execution path (the $PATH variable), the full path name of the command must be specified.

A notification command exits with one of the following values:

```
LSB_RESIZE_NOTIFY_OK=0
```

```
LSB_RESIZE_NOTIFY_FAIL=1
```

LSF sets these environment variables in notification command environment. LSB_RESIZE_NOTIFY_OK indicates notification succeeds. For allocation both "grow" and "shrink" events, LSF updates the job allocation to reflect the new allocation.

LSB_RESIZE_NOTIFY_FAIL indicates notification failure. For allocation "grow" event, LSF reschedules the pending allocation request. For allocation "shrink" event, LSF fails the alloction release request.

Configuration to enable resizable jobs

The resizable jobs feature is enabled by defining an application profile using the RESIZABLE_JOBS parameter in lsb.applications.

Configuration file	Parameter and syntax	Behavior
lsb.applications	RESIZABLE_JOBS=Y|N|auto	• When RESIZABLE_JOBS=Y jobs submitted to the application profile are resizable. • When RESIZABLE_JOBS=auto jobs submitted to the application profile are automatically resizable. • To enable cluster-wide resizable behavior by default, define RESIZABLE_JOBS=Y in the default application profile.
	RESIZE_NOTIFY_CMD =notify_cmd	RESIZE_NOTIFY_CMD specifies an application-level resize notification command. The resize notification command is invoked on the first execution host of a running resizable job when a resize event occurs, including releasing resources and adding resources. LSF set appropriate environment variables to indicate the event type and before running the notification command.

Configuration to modify resizable job behavior

There is no configuration to modify resizable job behavior.

Resizable job commands

Commands for submission

Command	Description
bsub -app *application_profile_name*	Submits the job to the specified application profile configured for resizable jobs
bsub -app *application_profile_name* -rnc *resize_notification_command*	Submits the job to the specified application profile configured for resizable jobs, with the specified resize notification command.The job-level resize notification command overrides the application-level RESIZE_NOTIFY_CMD setting.
bsub -ar -app *application_profile_name*	Submits the job to the specified application profile configured for resizable jobs, as an autoresizable job. The job-level -ar option overrides the application-level RESIZABLE_JOBS setting. For example, if the application profile is not autoresizable, job level bsub -ar will make the job autoresizable.

Commands to monitor

Command	Description
bacct	• Displays resize notification command. • Displays resize allocation changes.
bhist	• Displays resize notification command. • Displays resize allocation changes. • Displays the job-level autoresizable attribute.

Command	Description
bjobs -l	• Displays resize notification command. • Displays resize allocation changes. • Displays the job-level autoresizable attribute. • Displays pending resize allocation requests.

Commands to control

Command	Description
bmod -ar|-arn	Add or remove the job-level autoresizable attribute. bmod only updates the autoresizable attribute for pending jobs.
bmod -rnc *resize_notification_cmd*|-rncn	Modify or remove resize notification command for submitted job.
bresize release	Release allocated resources from a running resizable job. • Release all slots except one slot from the first execution node. • Release all hosts except the first execution node. • Release a list of hosts and different slots for each explicitly. • Specify a resize notification command to be invoked on the first execution host of the job. To release resources from a running job, the job must be submitted to an application profile configured as resizable. • By default, only cluster administrators, queue administrators, root and the job owner are allowed to run bresize to change job allocations. • User group administrators are allowed to run bresize to change the allocation of jobs within their user groups.
bresize cancel	Cancel a pending allocation request. The active pending allocation request is from r auto-resize request generated by LSF automatically. If job does not have active pending request, the command fails with an error message.
bresize release -rnc *resize_notification_cmd*	Specify or remove a resize notification command. The resize notification is invoked on the job first execution node. The resize notification command only applies to the release request and overrides the corresponding resize notification parameters defined in either the application profile (RESIZE_NOTIFY_CMD in lsb.applications) and job level (bsub -rnc *notify_cmd*). If the resize notification command completes successfully, LSF considers the allocation release done and updates the job allocation. If the resize notification command fails, LSF does not update the job allocation. The *resize_notification_cmd* specifies the name of the executable to be invoked on the first execution host when the job's allocation has been modified. The resize notification command runs under the user account of job. -rncn removes the resize notification command in both job-level and application-level.

Command	Description
`bresize release -c`	By default, if the job has an active pending allocation request, LSF does not allow users to release resource. Use the bresize release -c command to cancel the active pending resource request when releasing slots from existing allocation. By default, the command only releases slots. If a job still has an active pending allocation request, but you do not want to allocate more resources to the job, use the bresize cancel command to cancel allocation request. Only the job owner, cluster administrators, queue administrators, user group administrators, and root are allowed to cancel pending resource allocation requests.

Commands to display configuration

Command	Description
`bapp`	Displays the value of parameters defined in `lsb.applications`.

Feature: External load indices

External load indices report the values of dynamic external resources. A dynamic external resource is a customer-defined resource with a numeric value that changes over time, such as the space available in a directory. Use the external load indices feature to make the values of dynamic external resources available to LSF, or to override the values reported for an LSF built-in load index.

About external load indices

LSF bases job scheduling and host selection decisions on the resources available within your cluster. A *resource* is a characteristic of a host (such as available memory) or a cluster (such as the number of shared software licenses) that LSF uses to make job scheduling and host selection decisions.

A *static resource* has a value that does not change, such as a host's maximum swap space. A *dynamic resource* has a numeric value that changes over time, such as a host's currently available swap space. *Load indices* supply the values of dynamic resources to a host's load information manager (LIM), which periodically collects those values.

LSF has a number of built-in load indices that measure the values of dynamic, *host-based resources* (resources that exist on a single host)—for example, CPU, memory, disk space, and I/O. You can also define *shared resources* (resources that hosts in your cluster share, such as floating software licenses) and make these values available to LSF to use for job scheduling decisions.

If you have specific workload or resource requirements at your site, the LSF administrator can define *external resources*. You can use both built-in and external resources for LSF job scheduling and host selection.

To supply the LIM with the values of dynamic external resources, either host-based or shared, the LSF administrator writes a site-specific executable called an *external load information manager* (elim) executable. The LSF administrator programs the elim to define external load indices, populate those indices with the values of dynamic external resources, and return the indices and their values to stdout. An elim can be as simple as a small script, or as complicated as a sophisticated C program.

> **Note:**
>
> LSF does not include a default elim; you should write your own executable to meet the requirements of your site.

The following illustrations show the benefits of using the external load indices feature. In these examples, jobs require the use of floating software licenses.

Default behavior (feature not enabled)

LSF_SERVERDIR

No **elim**

In lsf.shared, no external resource defined.

In lsf.cluster.*cluster_name*, no external resource mapped.

user submits a job → job → **bsub**

-R option specifies a resource requirement for a floating license.

LSF accepts job submission and looks for hosts with required resource.

LSF cannot find the floating license resource

job

The job pends indefinitely.
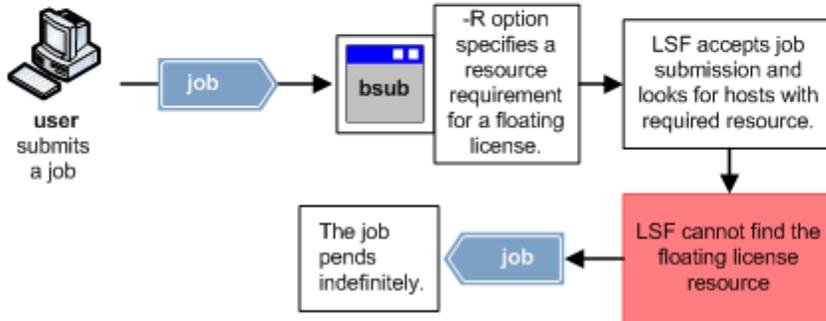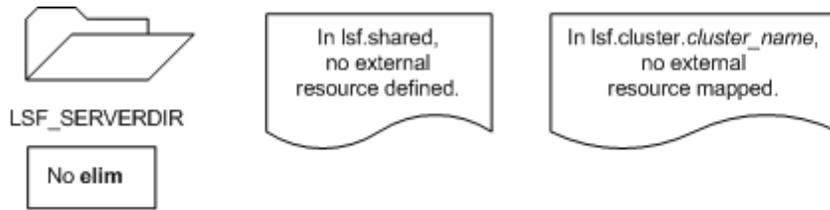
With external load indices enabled

LSF_SERVERDIR

elim

In lsf.shared, "floating_license" defined

In lsf.cluster.*cluster_name*, "floating_license" mapped to master host.

user submits a job → job → **bsub**

-R option specifies a resource requirement for a floating license.

LSF accepts job submission and looks for hosts with required resource.

elim

elim starts and sends load indices to LSF.

LSF uses the load indices to identify hosts with the required resource available.

HostA has floating license available.

Job runs on HostA.

Scope

Applicability	Details
Operating system	- UNIX - Windows - A mix of UNIX and Windows hosts
Dependencies	- UNIX and Windows user accounts must be valid on all hosts in the cluster and must have the correct permissions to successfully run jobs. - All elim executables run under the same user account as the load information manager (LIM)—by default, the LSF administrator (lsfadmin) account. - External dynamic resources (host-based or shared) must be defined in lsf.shared.

Configuration to enable external load indices

To enable the use of external load indices, you must

- Define the dynamic external resources in lsf.shared. By default, these resources are host-based (local to each host) until the LSF administrator configures a resource-to-host-mapping in the ResourceMap section of lsf.cluster.*cluster_name*. The presence of the dynamic external resource in lsf.shared and lsf.cluster.*cluster_name* triggers LSF to start the elim executables.
- Map the external resources to hosts in your cluster in lsf.cluster.*cluster_name*.

   Important:

   You must run the command **lsadmin reconfig** followed by **badmin mbdrestart** to apply changes.

- Create one or more elim executables in the directory specified by the parameter LSF_SERVERDIR. LSF does not include a default elim; you should write your own executable to meet the requirements of your site. The section Create an elim executable provides guidelines for writing an elim.

Define a dynamic external resource

To define a dynamic external resource for which elim collects an external load index value, define the following parameters in the Resource section of lsf.shared:

Configuration file	Parameter and syntax	Description
lsf.shared	RESOURCENAME *resource_name*	• Specifies the name of the external resource.
	TYPE Numeric	• Specifies the type of external resource: Numeric resources have numeric values. • Specify Numeric for all dynamic resources.
	INTERVAL *seconds*	• Specifies the interval for data collection by an elim. • For numeric resources, defining an interval identifies the resource as a dynamic resource with a corresponding external load index. **Important:** You must specify an interval: LSF treats a numeric resource with no interval as a static resource and, therefore, does not collect load index values for that resource.

	INCREASING
	Y | N
		• Specifies whether a larger value indicates a greater load.
		  • Y—a larger value indicates a greater load. For example, if you define an external load index for the number of shared software licenses *in use*, the larger the value, the heavier the load.
		  • N—a larger value indicates a lighter load. For example, if you define an external load index for the number of shared software licenses *currently available*, the larger the value, the lighter the load, and the more licenses are available.

	RELEASE
	Y | N
		• For shared resources only, specifies whether LSF releases the resource when a job that uses the resource is suspended.
		  • Y—Releases the resource when a job is suspended.
		  • N—Holds the resource when a job is suspended.

	DESCRIPTION *description*	• Brief description of the resource. Enter a description that enables you to easily identify the type and purpose of the resource. • The lsinfo command and the ls_info() API call return the contents of the DESCRIPTION parameter.

Map an external resource

Once external resources are defined in lsf.shared, they must be mapped to hosts in the ResourceMap section of lsf.cluster.*cluster_name*.

Configuration file	Parameter and syntax	Default behavior
lsf.cluster.*cluster_name*	RESOURCENAME*resource_name*	• Specifies the name of the external resource as defined in the Resource section of lsf.shared.
	LOCATION • ([all]) | ([all ~*host_name* …])	• Maps the resource to the master host only; all hosts share a single instance of the dynamic external resource. • To prevent specific hosts from accessing the resource, use the not operator (~) and specify one or more host names. All other hosts can access the resource.
	• [default]	• Maps the resource to all hosts in the cluster; every host has an instance of the dynamic external resource. • If you use the default keyword for any external resource, all elim executables in LSF_SERVERDIR run on all hosts in the cluster. For information about how to control which elim executables run on each host, see the section How LSF determines which hosts should run an elim executable.
	• ([*host_name* …]) | ([*host_name* …] [*host_name* …])	• Maps the resource to one or more specific hosts. • To specify sets of hosts that share a dynamic external resource, enclose each set in square brackets ([ ]) and use a space to separate each host name.

Create an elim executable

You can write one or more elim executables. The load index names defined in your elim executables must be the same as the external resource names defined in the lsf.shared configuration file.

All elim executables must

• Be located in LSF_SERVERDIR and follow these naming conventions:

Operating system	Naming convention
UNIX	LSF_SERVERDIR\elim.*application*

Operating system	Naming convention
Windows	LSF_SERVERDIR\elim.*application*.exe or LSF_SERVERDIR\elim.*application*.bat

---

Restriction:

The name elim.user is reserved for backward compatibility. Do not use the name elim.user for your application-specific elim.

Note:

LSF invokes any elim that follows this naming convention,—move backup copies out of LSF_SERVERDIR or choose a name that does not follow the convention. For example, use elim_backup instead of elim.backup.

- Exit upon receipt of a SIGTERM signal from the load information manager (LIM).
- Periodically output a *load update string* to stdout in the format *number_indices index_name index_value* [*index_name index_value* …] where

Value	Defines
number_indices	• The number of external load indices collected by the elim.
index_name	• The name of the external load index.
index_value	• The external load index value returned by your elim.

For example, the string

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

reports three indices: tmp2, nio, and licenses, with values 47.5, 344.0, and 5, respectively.

- • The load update string must report values between -INFINIT_LOAD and INFINIT_LOAD as defined in the lsf.h header file.
  - The elim should ensure that the entire load update string is written successfully to stdout. Program the elim to exit if it fails to write the load update string to stdout.

    - If the elim executable is a C program, check the return value of printf(3s).
    - If the elim executable is a shell script, check the return code of /bin/echo(1).
  - If the elim executable is implemented as a C program, use setbuf(3) during initialization to send unbuffered output to stdout.
  - Each LIM sends updated load information to the master LIM every 15 seconds; the elim executable should write the load update string at most once every 15 seconds. If the external load index values rarely change, program the elim to report the new values only when a change is detected.

If you map any external resource as default in lsf.cluster.*cluster_name*, all elim executables in LSF_SERVERDIR run on all hosts in the cluster. If LSF_SERVERDIR contains more than one elim executable, you should include a header that checks whether the elim

is programmed to report values for the resources expected on the host. For detailed information about using a checking header, see the section How environment variables determine elim hosts.

Overriding built-in load indices

An `elim` executable can be used to override the value of a built-in load index. For example, if your site stores temporary files in the `/usr/tmp` directory, you might want to monitor the amount of space available in that directory. An `elim` can report the space available in the `/usr/tmp` directory as the value for the tmp built-in load index. However, the value reported by an `elim` must less than the maximum size of `/usr/tmp`.

To override a built-in load index value, you must:

- Write an `elim` executable that periodically measures the value of the dynamic external resource and writes the numeric value to standard output. The external load index must correspond to a numeric, dynamic external resource as defined by TYPE and INTERVAL in `lsf.shared`.
- Configure an external resource in `lsf.shared` and map the resource in `lsf.cluster.cluster_name`, even though you are overriding a built-in load index. Use a name other than the built-in load index, for example, mytmp rather than `tmp`.
- Program your `elim` to output the formal name of the built-in index (for example, r1m, it, ls, or swp), *not* the resource name alias (cpu, idle, login, or swap). For example, an `elim` that collects the value of the external resource mytmp reports the value as tmp (the built-in load index) in the load update string: `1 tmp 20`.

Setting up an ELIM to support JSDL

To support the use of Job Submission Description Language (JSDL) files at job submission, LSF collects the following load indices:

Attribute name	Attribute type	Resource name
OperatingSystemName	string	osname
OperatingSystemVersion	string	osver
CPUArchitectureName	string	cpuarch
IndividualCPUSpeed	int64	cpuspeed
IndividualNetworkBandwidth	int64	bandwidth (This is the maximum bandwidth).

The file `elim.jsdl` is automatically configured to collect these resources. To enable the use of `elim.jsdl`, uncomment the lines for these resources in the ResourceMap section of the file `lsf.cluster.cluster_name`.

Example of an elim executable

See the section How environment variables determine elim hosts for an example of a simple `elim` script.

You can find additional elim examples in the LSF_MISC/examples directory. The elim.c file is an elim written in C. You can modify this example to collect the external load indices required at your site.

External load indices behavior

How LSF manages multiple elim executables

The LSF administrator can write one elim executable to collect multiple external load indices, or the LSF administrator can divide external load index collection among multiple elim executables. On each host, the load information manager (LIM) starts a master elim (MELIM), which manages all elim executables on the host and reports the external load index values to the LIM. Specifically, the MELIM

- Starts elim executables on the host. The LIM checks the ResourceMap section LOCATION settings (default, all, or host list) and directs the MELIM to start elim executables on the corresponding hosts.

  > **Note:**
  >
  > If the ResourceMap section contains even one resource mapped as default, and if there are multiple elim executables in LSF_SERVERDIR, the MELIM starts all of the elim executables in LSF_SERVERDIR on all hosts in the cluster. Not all of the elim executables continue to run, however. Those that use a checking header could exit with ELIM_ABORT_VALUE if they are not programmed to report values for the resources listed in LSF_RESOURCES.

- Restarts an elim if the elim exits. To prevent system-wide problems in case of a fatal error in the elim, the maximum restart frequency is once every 90 seconds. The MELIM does *not* restart any elim that exits with ELIM_ABORT_VALUE.
- Collects the load information reported by the elim executables.
- Checks the syntax of load update strings before sending the information to the LIM.
- Merges the load reports from each elim and sends the merged load information to the LIM. If there is more than one value reported for a single resource, the MELIM reports the latest value.
- Logs its activities and data into the log file LSF_LOGDIR/melim.log.*host_name*
- Increases system reliability by buffering output from multiple elim executables; failure of one elim does not affect other elim executables running on the same host.

How LSF determines which hosts should run an elim executable

LSF provides configuration options to ensure that your elim executables run only when they can report the resources values expected on a host. This maximizes system performance and simplifies the implementation of external load indices. To control which hosts run elim executables, you

- Must map external resource names to locations in lsf.cluster.*cluster_name*
- Optionally, use the environment variables LSF_RESOURCES, LSF_MASTER, and ELIM_ABORT_VALUE in your elim executables

How resource mapping determines elim hosts

The following table shows how the resource mapping defined in
lsf.cluster.*cluster_name* determines the hosts on which your elim executables start.

If the specified LOCATION is …	Then the elim executables start on …
• ([all]) | ([all ~*host_name* …])	• The master host, because all hosts in the cluster (except those identified by the not operator [~]) share a single instance of the external resource.
• [default]	• Every host in the cluster, because the default setting identifies the external resource as host-based. • If you use the default keyword for any external resource, all elim executables in LSF_SERVERDIR run on all hosts in the cluster. For information about how to program an elim to exit when it cannot collect information about resources on a host, see How environment variables determine elim hosts.

• ([*host_name* …]) | ([*host_name* …] [*host_name* …])	• On the specified hosts.
	• If you specify a set of hosts, the elim executables start on the first host in the list. For example, if the LOCATION in the ResourceMap section of lsf.cluster.*cluster_name* is ([hostA hostB hostC] [hostD hostE hostF]):
	  ◦ LSF starts the elim executables on hostA and hostD to report values for the resources shared by that set of hosts.
	  ◦ If the host reporting the external load index values becomes unavailable, LSF starts the elim executables on the next available host in the list. In this example, if hostA becomes unavailable, LSF starts the elim executables on hostB.
	  ◦ If hostA becomes available again, LSF starts the elim executables on hostA and shuts down the elim executables on hostB.

How environment variables determine elim hosts

If you use the default keyword for any external resource in lsf.cluster.*cluster_name*, all elim executables in LSF_SERVERDIR run on all hosts in the cluster. You can control the hosts on which your elim executables run by using the environment variables LSF_MASTER, LSF_RESOURCES, and ELIM_ABORT_VALUE. These environment variables provide a way to ensure that elim executables run only when they are programmed to report the values for resources expected on a host.

- LSF_MASTER—You can program your elim to check the value of the LSF_MASTER environment variable. The value is Y on the master host and N on all other hosts. An elim executable can use this parameter to check the host on which the elim is currently running.

- LSF_RESOURCES—When the LIM starts an MELIM on a host, the LIM checks the resource mapping defined in the ResourceMap section of lsf.cluster.*cluster_name*. Based on the mapping (default, all, or a host list), the LIM sets LSF_RESOURCES to the list of resources expected on the host. Use

LSF_RESOURCES in a checking header to verify that an elim is programmed to collect values for at least one of the resources listed in LSF_RESOURCES.

- ELIM_ABORT_VALUE—An elim should exit with ELIM_ABORT_VALUE if the elim is not programmed to collect values for at least one of the resources listed in LSF_RESOURCES. The MELIM does not restart an elim that exits with ELIM_ABORT_VALUE.

The following sample code shows how to use a header to verify that an elim is programmed to collect load indices for the resources expected on the host. If the elim is not programmed to report on the requested resources, the elim does not need to run on the host.

```sh
#!/bin/sh
# list the resources that the elim can report to lim
my_resource="myrsc"
# do the check when $LSF_RESOURCES is defined by lim
if [ -n "$LSF_RESOURCES" ]; then
# check if the resources elim can report are listed in $LSF_RESOURCES
res_ok=`echo " $LSF_RESOURCES " | /bin/grep " $my_resource " `
# exit with $ELIM_ABORT_VALUE if the elim cannot report on at least
# one resource listed in $LSF_RESOURCES
    if [ "$res_ok" = " " ] ; then
        exit $ELIM_ABORT_VALUE
    fi
 fi
while [ 1 ];do
# set the value for resource "myrsc"
val="1"
# create an output string in the format:
# number_indices index1_name index1_value...
reportStr="1 $my_resource $val"
 echo "$reportStr"
# wait for 30 seconds before reporting again
sleep 30
done
```

Configuration to modify external load indices

Configuration file	Parameter and syntax	Behavior

`lsf.cluster.`
`cluster_name`
 	
Parameters section
	ELIMARGS=*cmd_line_args*	• Specifies the command-line arguments required by an `elim` on startup.

	ELIM_POLL_INTERVAL=*seconds*	• Specifies the frequency with which the LIM samples external load index information from the MELIM.
	LSF_ELIM_BLOCKTIME=*seconds*	• UNIX only. Specifies how long the MELIM waits before restarting an `elim` that fails to send a complete load update string. • The MELIM does not restart an `elim` that exits with ELIM_ABORT_VALUE.
	LSF_ELIM_DEBUG=y	• UNIX only. Used for debugging; logs all load information received from `elim` executables to the MELIM log file (`melim.log.`*host_name*).
	LSF_ELIM_RESTARTS=*integer*	• UNIX only. Limits the number of times an `elim` can be restarted. • You must also define either LSF_ELIM_DEBUG or LSF_ELIM_BLOCKTIME. • Defining this parameter prevents an ongoing restart loop in the case of a faulty `elim`.

External load indices commands

Commands to submit workload

Command	Description

bsub -R "*res_req***" [-R "res_req"] …**	• Runs the job on a host that meets the specified resource requirements.
	• If you specify a value for a dynamic external resource in the resource requirements string, LSF uses the most recent values provided by your `elim` executables for host selection.
	• For example:
	
	  • Define a dynamic external resource called "usr_tmp" that represents the space available in the `/usr/tmp` directory.
	  • Write an `elim` executable to report the value of usr_tmp to LSF.
	  • To run the job on hosts that have more than 15 MB available in the `/usr/tmp` directory, run the command **bsub -R "usr_tmp > 15" myjob**
	  • LSF uses the external load index value for usr_tmp to locate a host with more than 15 MB available in the `/usr/tmp` directory.

Commands to monitor

Command	Description
lsload	• Displays load information for all hosts in the cluster on a per host basis.
lsload **-R "***res_req***"**	• Displays load information for specific resources.

Commands to control

Command	Description
lsadmin reconfig followed by **badmin mbdrestart**	• Applies changes when you modify lsf.shared or lsf.cluster.*cluster_name*.

Commands to display configuration

Command	Description
lsinfo	• Displays configuration information for all resources, including the external resources defined in lsf.shared.
lsinfo **-l**	• Displays detailed configuration information for external resources.
lsinfo ***resource_name ...***	• Displays configuration information for the specified resources.
bhosts **-s**	• Displays information about numeric shared resources, including which hosts that share each resource.
bhosts **-s** *shared_resource_name ...*	• Displays configuration information for the specified resources.

Feature: External host and user groups

Use the external host and user groups feature to maintain group definitions for your site in a location external to LSF, and to import the group definitions on demand.

About external host and user groups

LSF provides you with the option to configure host groups, user groups, or both. When the membership of a host or user group changes frequently, or when the group contains a large number of members, you can use an external executable called `egroup` to retrieve a list of members rather than having to configure the group membership manually. You can write a site-specific `egroup` executable that retrieves host or user group names and the hosts or users that belong to each group.

You can write your `egroup` executable to retrieve group members for:

- One or more host groups
- One or more user groups
- Any combination of host and user groups

LSF does not include a default `egroup`; you should write your own executable to meet the requirements of your site.

Default behavior (feature not enabled)

The following illustrations show the benefits of using the external host and user groups feature.
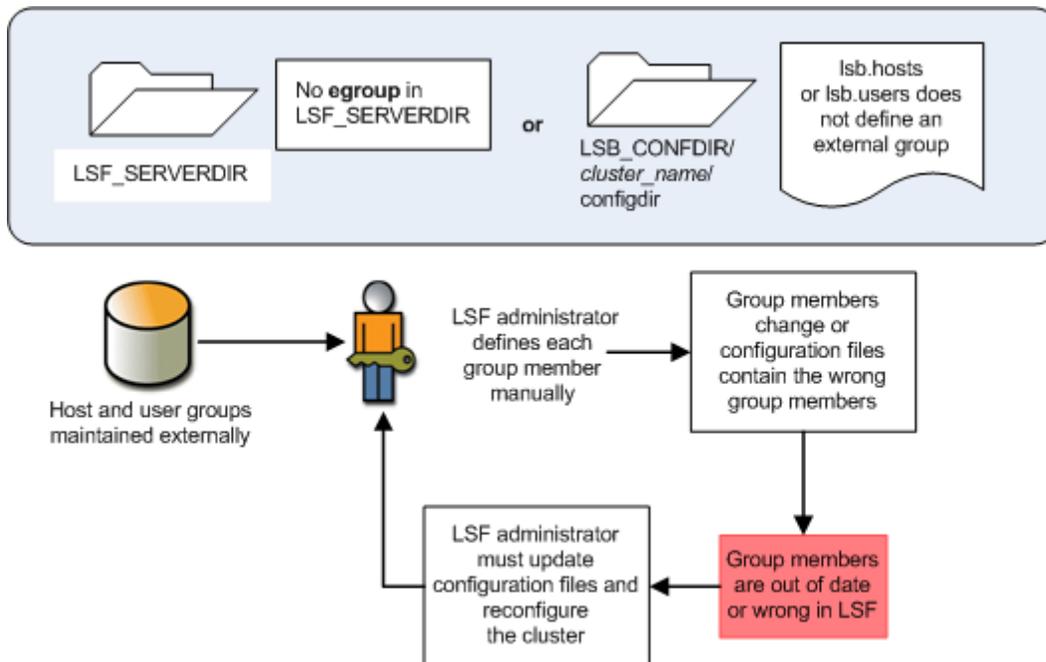
With external host and user groups enabled

Scope

Applicability	Details
Operating system	- UNIX - Windows - A mix of UNIX and Windows hosts
Dependencies	- UNIX and Windows user accounts must be valid on all hosts in the cluster and must have the correct permissions to successfully run jobs. - You must reconfigure the cluster using `badmin reconfig` each time you want to run the `egroup` executable to retrieve host or user group members.
Limitations	The egroup executable works with static hosts only; you cannot use an `egroup` executable to add a dynamically added host to a host group.
Not used with	Host groups when you have configured EGO-enabled service-level agreement (SLA) scheduling, because EGO resource groups replace LSF host groups.

Configuration to enable external host and user groups

To enable the use of external host and user groups, you must

- Define the host group in lsb.hosts, or the user group in lsb.users, and put an exclamation mark (!) in the GROUP_MEMBER column.
- Create an egroup executable in the directory specified by the parameter LSF_SERVERDIR in lsf.conf. LSF does not include a default egroup; you should write your own executable to meet the requirements of your site.
- Run the command badmin reconfig to reconfigure the cluster and import your external host and user groups.

Define an external host or user group

External host groups are defined in lsb.hosts, and external user groups are defined in lsb.users. Your egroup executable must define the same group names that you use in the lsb.hosts and lsb.users configuration files.

Configuration file	Parameter and syntax	Default behavior
lsb.hosts	GROUP_NAME GROUP_MEMBER *hostgroup_name* (!)	• Enables the use of an egroup executable to retrieve external host group members. • The *hostgroup_name* specified in lsb.hosts must correspond to the group name defined by the egroup executable. • You can configure one or more host groups to use the egroup executable. • LSF does not support the use of external host groups that contain dynamically added hosts.
lsb.users	GROUP_NAME GROUP_MEMBER *usergroup_name* (!)	• Enables the use of an egroup executable to retrieve external user group members. • The *usergroup_name* specified in lsb.users must correspond to the group name defined by the egroup executable. • You can configure one or more user groups to use the egroup executable.

Create an egroup executable

The egroup executable must

- Be located in LSF_SERVERDIR and follow these naming conventions:

Operating system	Naming convention
UNIX	LSF_SERVERDIR\egroup
Windows	LSF_SERVERDIR\egroup.exe
	or
	LSF_SERVERDIR\egroup.bat

- • Run when invoked by the commands egroup –m *hostgroup_name* and egroup –u *usergroup_name*. When mbatchd finds an exclamation mark (!) in the GROUP_MEMBER column of lsb.hosts or lsb.users, mbatchd runs the egroup command to invoke your egroup executable.
- • Output a space-delimited list of group members (hosts, users, or both) to stdout.
- • Retrieve a list of static hosts only. You cannot use the egroup executable to retrieve hosts that have been dynamically added to the cluster.

The following example shows a simple egroup script that retrieves both host and user group members:

```
#!/bin/sh
if [ "$1"="-m" ]; then                            #host group
if [ "$2"="linux_grp" ]; then                     #Linux hostgroup
        echo "linux01 linux02 linux03 linux04"
    elif [ "$2"="sol_grp" ]; then                 #Solaris hostgroup
        echo "Sol02 Sol02 Sol03 Sol04"
 fi
else                                              #user group
 if [ "$2"="srv_grp" ]; then                      #srvgrp user group
        echo "userA userB userC userD"
    elif [ "$2"="dev_grp" ]; then                 #devgrp user group
        echo "user1 user2 user3 user4"
 fi
fi
```

External host and user groups behavior

On restart and reconfiguration, mbatchd invokes the egroup executable to retrieve external host and user groups and then creates the groups in memory; mbatchd does *not* write the groups to lsb.hosts or lsb.users. The egroup executable runs under the same user account as mbatchd. By default, this is the root account on UNIX hosts and the cluster administrator account on Windows hosts.

Once LSF creates the groups in memory, the external host and user groups work the same way as any other LSF host and user groups, including configuration and batch command usage.

Configuration to modify external host and user groups

Not applicable: There are no parameters that modify the behavior of the egroup executable.

By defining additional parameters in lsb.hosts and lsb.users, however, you can configure the behavior of your external host and user groups the same way as you would for any LSF host or user group.

External host and user groups commands

Commands to submit workload

Command	Description
bsub -m *host_group*	• Submits a job to run on any host that belongs to the specified host group.
bsub -G *user_group*	• For fairshare scheduling only. Associates the job with the specified group. Specify any group that you belong to that does not contain subgroups.

Commands to monitor

Although you cannot monitor egroup behavior directly, you can display information about running jobs for specific host or user groups.

Command	Description
bjobs -m *host_group*	• Displays jobs submitted to run on any host that belongs to the specified host group.
bjobs -G *user_group*	• Displays jobs submitted using bsub -G for the specified user group.
bjobs -u *user_group*	• Displays jobs submitted by users that belong to the specified user group.

Commands to control

Command	Description
badmin reconfig	• Imports group members on demand from your external host and user group lists.

Commands to display configuration

Command	Description
bmgroup	• Displays a list of host groups and the names of hosts that belong to each group.
bugroup	• Displays a list of user groups and the names of users that belong to each group.

Use a text editor to view the lsb.hosts and lsb.users configuration files.

II
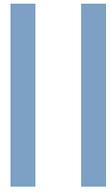
Configuration Files

Important:

Specify any domain names in all uppercase letters in all configuration files.

bld.license.acct

The `bld.license.acct` file is the license and accounting file for LSF License Scheduler.

bld.license.acct structure

The license accounting log file is an ASCII file with one record per line. The fields of a record are separated by blanks. LSF License Scheduler adds a new record to the file every hour.

File properties

Location

> The default location of this file is `LSF_SHAREDIR/db`. Use LSF_LICENSE_ACCT_PATH in `lsf.conf` to specify another location.

Owner

> The primary LSF License Scheduler admin is the owner of this file.

Permissions

`-rw-r--r--`

Records and fields

The fields in order of occurrence are as follows:

timestamp (%d)

> Time stamp of the logged event (in seconds since the epoch).

type (%s)

> The LSF product type. For LSF License Scheduler, this is `LICENSE_SCHEDULER`.

version (%s)

> The version of the LSF License Scheduler product.

value (%d)

> The total number of tokens that LSF License Scheduler is using.

status (%s)

> The results of the license usage check. The valid values are as follows:
>
> • `OK`
>
> Token usage is less than the currently licensed amount
>
> • `OVERUSE`
>
> Token usage is more than the currently licensed amount

hash (%s)

> Line encryption used to authenticate the record.

bld.license.acct

Example record format

```
1107961731 LICENSE_SCHEDULER 7.0 0 OK 335a33c2bd9c9428140a61e57bd06da02b623a42
1107961792 LICENSE_SCHEDULER 7.0 2 OK 58e45b891f371811edfcceb6f5270059a74ee31a
1126639979 LICENSE_SCHEDULER 7.0 0 5 OK b3efd43ee28346f2d125b445fd16aa96875da35
1126640028 LICENSE_SCHEDULER 7.0 6 5 OVERUSE 2865775920372225fa7f8ed4b9a8eb2b15
```

See also

- LSF_LOGDIR in lsf.conf
- LSF_LICENSE_ACCT_PATH in lsf.conf
- lsf.cluster_name.license.acct

cshrc.lsf and profile.lsf

About cshrc.lsf and profile.lsf

The user environment shell files `cshrc.lsf` and `profile.lsf` set the LSF operating environment on an LSF host. They define machine-dependent paths to LSF commands and libraries as environment variables:

- `cshrc.lsf` sets the C shell (`csh` or `tcsh`) user environment for LSF commands and libraries
- `profile.lsf` sets and exports the Bourne shell/Korn shell (`sh`, `ksh`, or `bash`) user environment for LSF commands and libraries

---

Tip:

LSF Administrators should make sure that `cshrc.lsf` or `profile.lsf` are available for users to set the LSF environment variables correctly for the host type running LSF.

Location

`cshrc.lsf` and `profile.lsf` are created by `lsfinstall` during installaton. After installation, they are located in LSF_CONFDIR (`LSF_TOP/conf/`).

Format

`cshrc.lsf` and `profile.lsf` are conventional UNIX shell scripts:

- `cshrc.lsf` runs under `/bin/csh`
- `profile.lsf` runs under `/bin/sh`

What cshrc.lsf and profile.lsf do

`cshrc.lsf` and `profile.lsf` determine the binary type (BINARY_TYPE) of the host and set environment variables for the paths to the following machine-dependent LSF directories, according to the LSF version (LSF_VERSION) and the location of the top-level installation directory (LSF_TOP) defined at installation:

- LSF_BINDIR
- LSF_SERVERDIR
- LSF_LIBDIR
- XLSF_UIDDIR

`cshrc.lsf` and `profile.lsf` also set the following user environment variables:

- LSF_ENVDIR
- LD_LIBRARY_PATH
- PATH to include the paths to:
  - LSF_BINDIR
  - LSF_SERVERDIR
- MANPATH to include the path to the LSF man pages

If Platform EGO is enabled

If Platform EGO is enabled in the LSF cluster (LSF_ENABLE_EGO=Y and LSF_EGO_ENVDIR are defined in `lsf.conf`), `cshrc.lsf` and `profile.lsf` set the following environment variables.

- EGO_BINDIR
- EGO_CONFDIR
- EGO_ESRVDIR
- EGO_LIBDIR
- EGO_LOCAL_CONFDIR
- EGO_SERVERDIR
- EGO_TOP

See the *Platform EGO Reference* for more information about these variables.

Setting the LSF environment with cshrc.lsf and profile.lsf

Before using LSF, you must set the LSF execution environment.

After logging on to an LSF host, use one of the following shell environment files to set your LSF environment:

- For example, in `csh` or `tcsh`:
  source /usr/share/lsf/lsf_7/conf/cshrc.lsf
- For example, in `sh`, `ksh`, or `bash`:
  . /usr/share/lsf/lsf_7/conf/profile.lsf

Making your cluster available to users with cshrc.lsf and profile.lsf

To set the LSF user environment, run one of the following two shell files:

- `LSF_CONFDIR/cshrc.lsf` (for `csh`, `tcsh`)
- `LSF_CONFDIR/profile.lsf` (for `sh`, `ksh`, or `bash`)

> **Tip:**
>
> LSF administrators should make sure all LSF users include one of these files at the end of their own `.cshrc` or `.profile` file, or run one of these two files before using LSF.

For csh or tcsh

Add `cshrc.lsf` to the end of the `.cshrc` file for all users:

- Copy the `cshrc.lsf` file into `.cshrc`, or
- Add a line similar to the following to the end of `.cshrc`:
  `source /usr/share/lsf/lsf_7/conf/cshrc.lsf`

After running `cshrc.lsf`, use `setenv` to see the environment variable settings. For example:

```
setenv PATH=/usr/share/lsf/lsf_7/7.0/linux2.6-glibc2.3-x86/bin:/usr/share/lsf/lsf_7/7.0/linux2.6-
glibc2.3-x86/etc:/home/user1/bin:/local/private/user1/bin:/etc:/usr/etc:/usr/local/bin:/usr/
local/sbin:/bin:/usr/bin:/usr/sbin:/opt/local/bin:/local/share/bin:/opt/gnu/bin:/sbin:/usr/bin/
X11:/usr/bsd:/usr/ucb:/local/bin/X11:/usr/hosts:/usr/openwin/bin:/usr/ccs/bin:/usr/vue/bin:.
... MANPATH=/usr/share/lsf/lsf_7/7.0/man:/home/user1/man:/opt/SUNWhpc/man:/usr/man:/usr/local/
man:/usr/softbench/man:/usr/openwin/man:/opt/SUNWmotif/man:/opt/ansic/share/man:/opt/hpnp/man:/
usr/share/man:/usr/share/catman
...
LSF_BINDIR=/usr/share/lsf/lsf_7/7.0/linux2.6-glibc2.3-x86/bin LSF_SERVERDIR=/usr/share/lsf/
lsf_7/7.0/linux2.6-glibc2.3-x86/etc LSF_LIBDIR=/usr/share/lsf/lsf_7/7.0/linux2.6-glibc2.3-x86/
lib LD_LIBRARY_PATH=/usr/share/lsf/lsf_7/7.0/linux2.6-glibc2.3-x86/lib XLSF_UIDDIR=/usr/share/
lsf/lsf_7/7.0/linux2.6-glibc2.3-x86/lib/uid LSF_ENVDIR=/usr/share/lsf/lsf_7/conf
```

> **Note:**

These variable settings are an example only. Your system may set additional variables.

For sh, ksh, or bash

Add `profile.lsf` to the end of the `.profile` file for all users:

- Copy the `profile.lsf` file into `.profile`, or
- Add a line similar to following to the end of `.profile`:

  ```
  . /usr/share/lsf/lsf_7/conf/profile.lsf
  ```

After running `profile.lsf`, use the `set` command to see the environment variable settings. For example:

```
set
...
LD_LIBRARY_PATH=/usr/share/lsf/lsf_7/7.0/linux2.6-glibc2.3-x86/lib LSF_BINDIR=/usr/share/lsf/
lsf_7/7.0/linux2.6-glibc2.3-x86/bin LSF_ENVDIR=/usr/share/lsf/lsf_7/conf LSF_LIBDIR=/usr/share/
lsf/lsf_7/7.0/linux2.6-glibc2.3-x86/lib LSF_SERVERDIR=/usr/share/lsf/lsf_7/7.0/linux2.6-glibc2.3-
x86/etc MANPATH=/usr/share/lsf/lsf_7/7.0/man:/home/user1/man:/opt/SUNWhpc/man:/usr/man:/usr/
local/man:/usr/softbench/man:/usr/openwin/man:/opt/SUNWmotif/man:/opt/ansic/share/man:/opt/hpnp/
man:/usr/share/man:/usr/share/catman PATH=/usr/share/lsf/lsf_7/7.0/linux2.6-glibc2.3-x86/bin:/
usr/share/lsf/lsf_7/7.0/linux2.6-glibc2.3-x86/etc:/home/user1/bin:/local/private/user1/bin:/etc:/
usr/etc:/usr/local/bin:/usr/local/sbin:/bin:/usr/bin:/usr/sbin:/opt/local/bin:/local/share/bin:/
opt/gnu/bin:/sbin:/usr/bin/X11:/usr/bsd:/usr/ucb:/local/bin/X11:/usr/hosts:/usr/openwin/bin:/usr/
ccs/bin:/usr/vue/bin:.
...
XLSF_UIDDIR=/usr/share/lsf/lsf_7/7.0/linux2.6-glibc2.3-x86/lib/uid
...
```

Note:

These variable settings are an example only. Your system may set additional variables.

cshrc.lsf and profile.lsf on dynamically added LSF slave hosts

Dynamically added LSF hosts that will not be master candidates are *slave hosts*. Each dynamic slave host has its own LSF binaries and local `lsf.conf` and shell environment scripts (`cshrc.lsf` and `profile.lsf`).

LSF environment variables set by cshrc.lsf and profile.lsf

LSF_BINDIR

Syntax

LSF_BINDIR=*dir*

Description

Directory where LSF user commands are installed.

Examples

- Set in `csh` and `tcsh` by `cshrc.lsf`:

```
setenv LSF_BINDIR /usr/share/lsf/lsf_7/7.0/linux2.6-glibc2.3-x86/bin
```

- Set and exported in `sh`, `ksh`, or `bash` by `profile.lsf`:

  ```
  LSF_BINDIR=/usr/share/lsf/lsf_7/7.0/linux2.6-glibc2.3-x86/bin
  ```

Values

- In `cshrc.lsf` for `csh` and `tcsh`:
  ```
  setenv LSF_BINDIR $LSF_TOP/$LSF_VERSION/$BINARY_TYPE/bin
  ```
- Set and exported in `profile.lsf` for sh, ksh, or bash:
  ```
  LSF_BINDIR=$LSF_TOP/$LSF_VERSION/$BINARY_TYPE/bin
  ```

LSF_ENVDIR

Syntax

LSF_ENVDIR=*dir*

Description

Directory containing the `lsf.conf` file.

By default, `lsf.conf` is installed by creating a shared copy in LSF_CONFDIR and adding a symbolic link from `/etc/lsf.conf` to the shared copy. If LSF_ENVDIR is set, the symbolic link is installed in `LSF_ENVDIR/lsf.conf`.

The `lsf.conf` file is a global environment configuration file for all LSF services and applications. The LSF default installation places the file in LSF_CONFDIR.

Examples

- Set in `csh` and `tcsh` by `cshrc.lsf`:
  ```
  setenv LSF_ENVDIR /usr/share/lsf/lsf_7/conf
  ```
- Set and exported in sh, ksh, or bash by `profile.lsf`:
  ```
  LSF_ENVDIR=/usr/share/lsf/lsf_7/conf
  ```

Values

- In `cshrc.lsf` for `csh` and `tcsh`:
  ```
  setenv LSF_ENVDIR $LSF_TOP/conf
  ```
- Set and exported in `profile.lsf` for sh, ksh, or bash:
  ```
  LSF_DIR=$LSF_TOP/conf
  ```

LSF_LIBDIR

Syntax

LSF_LIBDIR=*dir*

Description

Directory where LSF libraries are installed. Library files are shared by all hosts of the same type.

Examples

- Set in `csh` and `tcsh` by `cshrc.lsf`:
```
setenv LSF_LIBDIR /usr/share/lsf/lsf_7/7.0/linux2.6-glibc2.3-x86/lib
```
- Set and exported in sh, ksh, or bash by `profile.lsf`:
  ```
  LSF_LIBDIR=/usr/share/lsf/lsf_7/7.0/linux2.6-glibc2.3-x86/lib
  ```

Values

- In `cshrc.lsf` for `csh` and `tcsh`:
  `setenv LSF_LIBDIR $LSF_TOP/$LSF_VERSION/$BINARY_TYPE/lib`
- Set and exported in `profile.lsf` for `sh`, `ksh`, or `bash`:
  `LSF_LIBDIR=$LSF_TOP/$LSF_VERSION/$BINARY_TYPE/lib`

LSF_SERVERDIR

Syntax

LSF_SERVERDIR=*dir*

Description

Directory where LSF server binaries and shell scripts are installed.

These include `lim`, `res`, `nios`, `sbatchd`, `mbatchd`, and `mbschd`. If you use `elim`, `eauth`, `eexec`, `esub`, etc, they are also installed in this directory.

Examples

- Set in `csh` and `tcsh` by `cshrc.lsf`:
`setenv LSF_SERVERDIR /usr/share/lsf/lsf_7/7.0/linux2.6-glibc2.3-x86/etc`
- Set and exported in `sh`, `ksh`, or `bash` by `profile.lsf`:
  `LSF_SERVERDIR=/usr/share/lsf/lsf_7/7.0/linux2.6-glibc2.3-x86/etc`

Values

- In `cshrc.lsf` for `csh` and `tcsh`:
  `setenv LSF_SERVERDIR $LSF_TOP/$LSF_VERSION/$BINARY_TYPE/etc`
- Set and exported in `profile.lsf` for `sh`, `ksh`, or `bash`:
  `LSF_SERVERDIR=$LSF_TOP/$LSF_VERSION/$BINARY_TYPE/etc`

XLSF_UIDDIR

Syntax

XLSF_UIDDIR=*dir*

Description

(UNIX and Linux only) Directory where Motif User Interface Definition files are stored.

These files are platform-specific.

Examples

- Set in `csh` and `tcsh` by `cshrc.lsf`:
`setenv XLSF_UIDDIR /usr/share/lsf/lsf_7/7.0/linux2.6-glibc2.3-x86/lib/uid`
- Set and exported in `sh`, `ksh`, or `bash` by `profile.lsf`:
  `XLSF_UIDDIR=/usr/share/lsf/lsf_7/7.0/linux2.6-glibc2.3-x86/lib/uid`

Values

- In `cshrc.lsf` for csh and `tcsh`:
  ```
  setenv XLSF_UIDDIR $LSF_TOP/$LSF_VERSION/$BINARY_TYPE/lib/uid
  ```
- Set and exported in `profile.lsf` for sh, ksh, or bash:
  ```
  XLSF_UIDDIR=$LSF_TOP/$LSF_VERSION/$BINARY_TYPE/lib/uid
  ```

Platform EGO environment variables set by cshrc.lsf and profile.lsf

See the *Platform EGO Reference* for more information about these variables.

EGO_BINDIR

Syntax

EGO_BINDIR=*dir*

Description

Directory where Platform EGO user commands are installed.

Examples

- Set in csh and `tcsh` by `cshrc.lsf`:
```
setenv EGO_BINDIR /usr/share/lsf/lsf_7/7.0/linux2.6-glibc2.3-x86/bin
```
- Set and exported in sh, ksh, or bash by `profile.lsf`:
  ```
  EGO_BINDIR=/usr/share/lsf/lsf_7/7.0/linux2.6-glibc2.3-x86/bin
  ```

Values

- In `cshrc.lsf` for csh and `tcsh`:
  ```
  setenv EGO_BINDIR $LSF_BINDIR
  ```
- Set and exported in `profile.lsf` for sh, ksh, or bash:
  ```
  EGO_BINDIR=$LSF_BINDIR
  ```

EGO_CONFDIR

Syntax

EGO_CONFDIR=*dir*

Description

Directory containing the `ego.conf` file.

Examples

- Set in csh and `tcsh` by `cshrc.lsf`:
  ```
  setenv EGO_CONFDIR /usr/share/lsf/lsf_7/conf/ego/lsf1.2.3/kernel
  ```
- Set and exported in sh, ksh, or bash by `profile.lsf`:
  ```
  EGO_CONFDIR=/usr/share/lsf/lsf_7/conf/ego/lsf1.2.3/kernel
  ```

Values

- In `cshrc.lsf` for csh and `tcsh`:
  ```
  setenv EGO_CONFDIR /usr/share/lsf/lsf_7/conf/ego/lsf1.2.3/kernel
  ```
- Set and exported in `profile.lsf` for sh, ksh, or bash:
  ```
  EGO_CONFDIR=/usr/share/lsf/lsf_7/conf/ego/lsf1.2.3/kernel
  ```

EGO_ESRVDIR

Syntax

EGO_ESRVDIR=*dir*

Description

Directory where the EGO the service controller configuration files are stored.

Examples

- Set in csh and `tcsh` by `cshrc.lsf`:
  ```
  setenv EGO_ESRVDIR /usr/share/lsf/lsf_7/conf/ego/lsf702/eservice
  ```
- Set and exported in sh, ksh, or bash by `profile.lsf`:
  ```
  EGO_ESRVDIR=/usr/share/lsf/lsf_7/conf/ego/lsf702/eservice
  ```

Values

- In `cshrc.lsf` for csh and `tcsh`:
  ```
  setenv EGO_ESRVDIR /usr/share/lsf/lsf_7/conf/ego/lsf702/eservice
  ```
- Set and exported in `profile.lsf` for sh, ksh, or bash:
  ```
  EGO_ESRVDIR=/usr/share/lsf/lsf_7/conf/ego/lsf702/eservice
  ```

EGO_LIBDIR

Syntax

EGO_LIBDIR=*dir*

Description

Directory where EGO libraries are installed. Library files are shared by all hosts of the same type.

Examples

- Set in csh and `tcsh` by `cshrc.lsf`:
  ```
  setenv EGO_LIBDIR /usr/share/lsf/lsf_7/7.0/linux2.6-glibc2.3-x86/lib
  ```
- Set and exported in sh, ksh, or bash by `profile.lsf`:
  ```
  EGO_LIBDIR=/usr/share/lsf/lsf_7/7.0/linux2.6-glibc2.3-x86/lib
  ```

Values

- In `cshrc.lsf` for csh and `tcsh`:
  ```
  setenv EGO_LIBDIR $LSF_LIBDIR
  ```

- Set and exported in `profile.lsf` for sh, ksh, or bash:

  `EGO_LIBDIR=$LSF_LIBDIR`

EGO_LOCAL_CONFDIR

Syntax

EGO_LOCAL_CONFDIR=*dir*

Description

The local EGO configuration directory containing the `ego.conf` file.

Examples

- Set in `csh` and `tcsh` by `cshrc.lsf`:

  `setenv EGO_LOCAL_CONFDIR /usr/share/lsf/lsf_7/conf/ego/lsf1.2.3/kernel`
- Set and exported in sh, ksh, or bash by `profile.lsf`:

  `EGO_LOCAL_CONFDIR=/usr/share/lsf/lsf_7/conf/ego/lsf1.2.3/kernel`

Values

- In `cshrc.lsf` for csh and `tcsh`:

  `setenv EGO_LOCAL_CONFDIR /usr/share/lsf/lsf_7/conf/ego/lsf1.2.3/kernel`
- Set and exported in `profile.lsf` for sh, ksh, or bash:

  `EGO_LOCAL_CONFDIR=/usr/share/lsf/lsf_7/conf/ego/lsf1.2.3/kernel`

EGO_SERVERDIR

Syntax

EGO_SERVERDIR=*dir*

Description

Directory where EGO server binaries and shell scripts are installed. These include `vemkd`, `pem`, `egosc`, and shell scripts for EGO startup and shutdown.

Examples

- Set in `csh` and `tcsh` by `cshrc.lsf`:

`setenv EGO_SERVERDIR /usr/share/lsf/lsf_7/7.0/linux2.6-glibc2.3-x86/etc`
- Set and exported in sh, ksh, or bash by `profile.lsf`:

  `EGO_SERVERDIR=/usr/share/lsf/lsf_7/7.0/linux2.6-glibc2.3-x86/etc`

Values

- In `cshrc.lsf` for csh and `tcsh`:

  `setenv EGO_SERVERDIR $LSF_SERVERDIR`
- Set and exported in `profile.lsf` for sh, ksh, or bash:

  `EGO_SERVERDIR=$LSF_SERVERDIR`

EGO_TOP

Syntax

EGO_TOP=*dir*

Description

The the top-level installation directory. The path to EGO_TOP must be shared and accessible to all hosts in the cluster. Equivalent to LSF_TOP.

Examples

- Set in csh and tcsh by cshrc.lsf:
  ```
  setenv EGO_TOP /usr/share/lsf/lsf_7
  ```
- Set and exported in sh, ksh, or bash by profile.lsf:
  ```
  EGO_TOP=/usr/share/lsf/lsf_7
  ```

Values

- In cshrc.lsf for csh and tcsh:
  ```
  setenv EGO_TOP /usr/share/lsf/lsf_7
  ```
- Set and exported in profile.lsf for sh, ksh, or bash:
  ```
  EGO_TOP=/usr/share/lsf/lsf_7
  ```

hosts

For hosts with multiple IP addresses and different official host names configured at the system level, this file associates the host names and IP addresses in LSF.

By default, LSF assumes each host in the cluster:

- Has a unique "official" host name
- Can resolve its IP address from its name
- Can resolve its official name from its IP address

Hosts with only one IP address, or hosts with multiple IP addresses that already resolve to a unique official host name should not be configured in this file: they are resolved using the default method for your system (for example, local configuration files like /etc/hosts or through DNS.)

The LSF hosts file is used in environments where:

- Machines in cluster have multiple network interfaces and cannot be set up in the system with a unique official host name
- DNS is slow or not configured properly
- Machines have special topology requirements; for example, in HPC systems where it is desirable to map multiple actual hosts to a single "head end" host

The LSF hosts file is not installed by default. It is usually located in the directory specified by LSF_CONFDIR. The format of LSF_CONFDIR/hosts is similar to the format of the /etc/hosts file on UNIX machines.

hosts file structure

One line for each IP address, consisting of the IP address, followed by the official host name, optionally followed by host aliases, all separated by spaces or tabs. Each line has the form:

```
ip_address official_name [alias [alias ...]]
```

IP addresses can have either a dotted quad notation (IPv4) or IP Next Generation (IPv6) format. You can use IPv6 addresses if you define the parameter LSF_ENABLE_SUPPORT_IPV6 in lsf.conf; you do not have to map IPv4 addresses to an IPv6 format.

Use consecutive lines for IP addresses belonging to the same host. You can assign different aliases to different addresses.

Use a pound sign (#) to indicate a comment (the rest of the line is not read by LSF). Do not use #if as this is reserved syntax for time-based configuration.

IP address

Written using an IPv4 or IPv6 format. LSF supports both formats; you do not have to map IPv4 addresses to an IPv6 format (if you define the parameter LSF_ENABLE_SUPPORT_IPV6 in lsf.conf).

- IPv4 format: nnn.nnn.nnn.nnn
- IPv6 format: nnnn:nnnn:nnnn:nnnn:nnnn:nnnn:nnnn:nnnn

Official host name

The official host name. Single character names are not allowed.

Specify -GATEWAY or -GW as part of the host name if the host serves as a GATEWAY.

Specify -TAC as the last part of the host name if the host is a TAC and is a DoD host.

Specify the host name in the format defined in Internet RFC 952, which states:

A "name" (Net, Host, Gateway, or Domain name) is a text string up to 24 characters drawn from the alphabet (A-Z), digits (0-9), minus sign (-), and period (.). Periods are only allowed when they serve to delimit components of "domain style names". (See RFC 921, "Domain Name System Implementation Schedule", for background). No blank or space characters are permitted as part of a name. No distinction is made between upper and lower case. The first character must be an alpha character. The last character must not be a minus sign or a period.

RFC 952 has been modified by RFC 1123 to relax the restriction on the first character being a digit.

For maximum interoperability with the Internet, you should use host names no longer than 24 characters for the host portion (exclusive of the domain component).

Aliases

Optional. Aliases to the host name.

The default host file syntax

```
ip_address official_name [alias [alias ...]]
```

is powerful and flexible, but it is difficult to configure in systems where a single host name has many aliases, and in multihomed host environments.

In these cases, the hosts file can become very large and unmanageable, and configuration is prone to error.

The syntax of the LSF hosts file supports host name ranges as aliases for an IP address. This simplifies the host name alias specification.

To use host name ranges as aliases, the host names must consist of a fixed node group name prefix and node indices, specified in a form like:

```
host_name[index_x-index_y, index_m, index_a-index_b]
```

For example:

```
atlasD0[0-3, 4, 5-6, ...]
```

is equivalent to:

```
atlasD0[0-6, ...]
```

The node list does not need to be a continuous range (some nodes can be configured out). Node indices can be numbers or letters (both upper case and lower case).

For example, some systems map internal compute nodes to single LSF host names. A host file might contains 64 lines, each specifying an LSF host name and 32 node names that correspond to each LSF host:

```
...
177.16.1.1 atlasD0 atlas0 atlas1 atlas2 atlas3 atlas4 ... atlas31
177.16.1.2 atlasD1 atlas32 atlas33 atlas34 atlas35 atlas36 ... atlas63
...
```

In the new format, you still map the nodes to the LSF hosts, so the number of lines remains the same, but the format is simplified because you only have to specify ranges for the nodes, not each node individually as an alias:

```
...
177.16.1.1 atlasD0 atlas[0-31]
177.16.1.2 atlasD1 atlas[32-63]
...
```

You can use either an IPv4 or an IPv6 format for the IP address (if you define the parameter LSF_ENABLE_SUPPORT_IPV6 in lsf.conf).

IPv4 Example

```
192.168.1.1 hostA hostB 192.168.2.2 hostA hostC host-C
```

In this example, hostA has 2 IP addresses and 3 aliases. The alias hostB specifies the first address, and the aliases hostC and host-C specify the second address. LSF uses the official host name, hostA, to identify that both IP addresses belong to the same host.

IPv6 Example

`3ffe:b80:3:1a91::2 hostA hostB 3ffe:b80:3:1a91::3 hostA hostC host-C`

In this example, hostA has 2 IP addresses and 3 aliases. The alias hostB specifies the first address, and the aliases hostC and host-C specify the second address. LSF uses the official host name, hostA, to identify that both IP addresses belong to the same host.

install.config

About install.config

The install.config file contains options for LSF installation and configuration. Use lsfinstall -f install.config to install LSF using the options specified in install.config.

Template location

A template install.config is included in the installation script tar file lsf7Update5_lsfinstall.tar.Z and is located in the lsf7Update5_lsfinstall directory created when you uncompress and extract installation script tar file. Edit the file and uncomment the options you want in the template file. Replace the example values with your own settings to specify the options for your new installation.

> **Important:**
>
> The sample values in the install.config template file are examples only. They are not default installation values.

After installation, the install.config containing the options you specified is located in LSF_TOP/7.0/install/.

Format

Each entry in install.config has the form:

```
NAME="STRING1 STRING2 ..."
```

The equal sign = must follow each NAME even if no value follows and there should be no spaces around the equal sign.

A value that contains multiple strings separated by spaces must be enclosed in quotation marks.

Blank lines and lines starting with a pound sign (#) are ignored.

Parameters

- DERBY_DB_HOST
- EGO_DAEMON_CONTROL
- EGO_PERF_CONTROL
- EGO_PMC_CONTROL
- ENABLE_DYNAMIC_HOSTS
- ENABLE_EGO
- ENABLE_HPC_CONFIG
- EP_BACKUP
- LSF_ADD_SERVERS
- LSF_ADD_CLIENTS
- LSF_ADMINS
- LSF_CLUSTER_NAME
- LSF_DYNAMIC_HOST_WAIT_TIME
- LSF_LICENSE
- LSF_MASTER_LIST

- LSF_QUIET_INST
- LSF_TARDIR
- LSF_TOP
- PATCH_BACKUP_DIR
- PATCH_HISTORY_DIR
- PERF_HOST
- PMC_HOST

DERBY_DB_HOST

Syntax

DERBY_DB_HOST="*host_name***"**

Description

Reporting database host. This parameter is used when you install the Platform Management Console package for the first time, and is ignored for all other cases.

Specify the name of a reliable host where the Derby database for Reporting data collection will be installed. You must specify a host from LSF_MASTER_LIST. Leave this parameter undefined if you will use another database for Reporting.

Example

DERBY_DB_HOST="hostd"

Default

Database is undefined.

EGO_DAEMON_CONTROL

Syntax

EGO_DAEMON_CONTROL="Y" | **"N"**

Description

Enables EGO to control LSF res and sbatchd. Set the value to "Y" if you want EGO Service Controller to start res and sbatchd, and restart if they fail. To avoid conflicts, leave this parameter undefined if you use a script to start up LSF daemons.

> **Note:**
>
> If you specify EGO_ENABLE="N", this parameter is ignored.

Example

EGO_DAEMON_CONTROL="N"

Default

N (res and sbatchd are started manually)

EGO_PERF_CONTROL

Syntax

EGO_PERF_CONTROL="Y" | "N"

Description

Enables EGO Service Controller to control PERF daemons. Set the value to "N" if you want to control PERF daemons manually. If you do this, you must define PERF_HOST in this file.

> **Note:**
>
> If you specify EGO_ENABLE="N", this parameter is ignored.

> **Note:**
>
> This parameter only takes effect when you install the Platform Management Console package for the first time.

Example

EGO_PERF_CONTROL="N"

Default

Y (PERF daemons are controlled by EGO unless EGO is disabled)

EGO_PMC_CONTROL

Syntax

EGO_PMC_CONTROL="Y" | "N"

Description

Enables EGO Service Controller to control the Platform Management Console. Set the value to "N" if you want to control the Platform Management Console manually.

> **Note:**
>
> If you specify EGO_ENABLE="N", this parameter is ignored.

> **Note:**
>
> This parameter only takes effect when you install the Platform Management Console package for the first time.

Example

EGO_PMC_CONTROL="N"

Default

Y (Platform Management Console is controlled by EGO unless EGO is disabled)

ENABLE_DYNAMIC_HOSTS

Syntax

ENABLE_DYNAMIC_HOSTS="Y" | **"N"**

Description

Enables dynamically adding and removing hosts. Set the value to "Y" if you want to allow dynamically added hosts.

If you enable dynamic hosts, any host can connect to cluster. To enable security, configure LSF_HOST_ADDR_RANGE in lsf.cluster.*cluster_name* after installation and restrict the hosts that can connect to your cluster.

Example

ENABLE_DYNAMIC_HOSTS="N"

Default

N (dynamic hosts not allowed)

ENABLE_EGO

Syntax

ENABLE_EGO="Y" | **"N"**

Description

Enables Platform EGO functionality in the LSF cluster.

ENABLE_EGO="Y" causes lsfinstall uncomment LSF_EGO_ENVDIR and sets LSF_ENABLE_EGO="Y" in lsf.conf.

ENABLE_EGO="N" causes lsfinstall to comment out LSF_EGO_ENVDIR and sets LSF_ENABLE_EGO="N" in lsf.conf.

Set the value to "N" if you do not want to take advantage of the following LSF features that depend on EGO:

- LSF daemon control by EGO Service Controller
- EGO-enabled SLA scheduling
- Platform Management Console (PMC)
- LSF reporting

Default

Y (EGO is enabled in the LSF cluster)

ENABLE_HPC_CONFIG

Syntax

ENABLE_HPC_CONFIG="Y" | **"N"**

Description

Set the value to "Y" to add LSF HPC configuration parameters to the cluster.

Default

Y (Platform LSF HPC is enabled.)

EP_BACKUP

Syntax

EP_BACKUP="Y" | **"N"**

Description

Enables backup and rollback for enhancement packs. Set the value to "N" to disable backups when installing enhancement packs (you will not be able to roll back to the previous patch level after installing an EP, but you will still be able to roll back any fixes installed on the new EP).

You may disable backups to speed up install time, to save disk space, or because you have your own methods to back up the cluster.

Default

Y (backup and rollback are fully enabled)

LSF_ADD_SERVERS

Syntax

LSF_ADD_SERVERS="*host_name* [*host_name*...]**"**

Description

List of additional LSF server hosts.

The hosts in LSF_MASTER_LIST are always LSF servers. You can specify additional server hosts. Specify a list of host names two ways:

- Host names separated by spaces
- Name of a file containing a list of host names, one host per line.

Valid Values

Any valid LSF host name.

Example 1

List of host names:

```
LSF_ADD_SERVERS="hosta hostb hostc hostd"
```

Example 2

Host list file:

```
LSF_ADD_SERVERS=:lsf_server_hosts
```

The file `lsf_server_hosts` contains a list of hosts:

```
hosta
hostb
hostc
hostd
```

Default

Only hosts in LSF_MASTER_LIST are LSF servers.

LSF_ADD_CLIENTS

Syntax

LSF_ADD_CLIENTS="*host_name* [*host_name...*]**"**

Description

List of LSF client-only hosts.

> **Tip:**
>
> After installation, you must manually edit
> `lsf.cluster.`*cluster_name* to include the host model and type
> of each client listed in LSF_ADD_CLIENTS.

Valid Values

Any valid LSF host name.

Example 1

List of host names:

```
LSF_ADD_CLIENTS="hoste hostf"
```

Example 2

Host list file:

```
LSF_ADD_CLIENTS=:lsf_client_hosts
```

The file `lsf_client_hosts` contains a list of hosts:

```
hoste
hostf
```

Default

No client hosts installed.

LSF_ADMINS

Syntax

LSF_ADMINS="*user_name* [*user_name ...*]**"**

Description

Required. List of LSF administrators.

The first user account name in the list is the primary LSF administrator. It cannot be the root user account.

Typically this account is named lsfadmin. It owns the LSF configuration files and log files for job events. It also has permission to reconfigure LSF and to control batch jobs submitted by other users. It typically does not have authority to start LSF daemons. Usually, only root has permission to start LSF daemons.

All the LSF administrator accounts must exist on all hosts in the cluster before you install LSF. Secondary LSF administrators are optional.

Caution:

You should *not* configure the root account as the primary LSF administrator.

Valid Values

Existing user accounts

Example

LSF_ADMINS="lsfadmin user1 user2"

Default

None—required variable

LSF_CLUSTER_NAME

Syntax

LSF_CLUSTER_NAME="*cluster_name***"**

Description

Required. The name of the LSF cluster.

Example

LSF_CLUSTER_NAME="cluster1"

Valid Values

Any alphanumeric string containing no more than 39 characters. The name cannot contain white spaces.

Important:

Do not use the name of any host, user, or user group as the name of your cluster.

Default

None—required variable

LSF_DYNAMIC_HOST_WAIT_TIME

Syntax

LSF_DYNAMIC_HOST_WAIT_TIME=*seconds*

Description

Time in seconds slave LIM waits after startup before calling master LIM to add the slave host dynamically.

This parameter only takes effect if you set ENABLE_DYNAMIC_HOSTS="Y" in this file. If the slave LIM receives the master announcement while it is waiting, it does not call the master LIM to add itself.

Recommended value

Up to 60 seconds for every 1000 hosts in the cluster, for a maximum of 15 minutes. Selecting a smaller value will result in a quicker response time for new hosts at the expense of an increased load on the master LIM.

Example

LSF_DYNAMIC_HOST_WAIT_TIME=60

Hosts will wait 60 seconds from startup to receive an acknowledgement from the master LIM. If it does not receive the acknowledgement within the 60 seconds, it will send a request for the master LIM to add it to the cluster.

Default

Slave LIM waits forever

LSF_LICENSE

Syntax

LSF_LICENSE="/*path***/***license_file***"**

Description

Full path to the name of the LSF license file, license.dat.

You must have a valid license file to install LSF.

Caution:

If you do not specify LSF_LICENSE, or lsfinstall cannot find a valid license file in the default location, lsfinstall exits.

Example

LSF_LICENSE="/usr/share/lsf_distrib/license.dat"

Default

The parent directory of the current working directory. For example, if lsfinstall is running under usr/share/lsf_distrib/lsf_lsfinstall the LSF_LICENSE default value is usr/share/lsf_distrib/license.dat.

LSF_MASTER_LIST

Syntax

LSF_MASTER_LIST="*host_name* [*host_name* ...]**"**

Description

Required for a first-time installation. List of LSF server hosts to be master or master candidates in the cluster.

You must specify at least one valid server host to start the cluster. The first host listed is the LSF master host.

During upgrade, specify the existing value.

Valid Values

LSF server host names

Example

LSF_MASTER_LIST="hosta hostb hostc hostd"

Default

None — required variable

LSF_QUIET_INST

Syntax

LSF_QUIET_INST="Y" | **"N"**

Description

Enables quiet installation.

Set the value to Y if you want to hide the LSF installation messages.

Example

LSF_QUIET_INST="Y"

Default

N (installer displays messages during installation)

install.config

LSF_TARDIR

Syntax

LSF_TARDIR="/*path***"**

Description

Full path to the directory containing the LSF distribution tar files.

Example

LSF_TARDIR="/usr/share/lsf_distrib"

Default

The parent directory of the current working directory. For example, if lsfinstall is running under usr/share/lsf_distrib/lsf_lsfinstall the LSF_TARDIR default value is usr/share/lsf_distrib.

LSF_TOP

Syntax

LSF_TOP="/*path***"**

Description

Required. Full path to the top-level LSF installation directory.

Valid Value

The path to LSF_TOP must be shared and accessible to all hosts in the cluster. It cannot be the root directory (/). The file system containing LSF_TOP must have enough disk space for all host types (approximately 300 MB per host type).

Example

LSF_TOP="/usr/share/lsf"

Default

None — required variable

PATCH_BACKUP_DIR

Syntax

PATCH_BACKUP_DIR="/*path***"**

Description

Full path to the patch backup directory. This parameter is used when you install a new cluster for the first time, and is ignored for all other cases.

The file system containing the patch backup directory must have sufficient disk space to back up your files (approximately 400 MB per binary type if you want to be able to install and roll back one enhancement pack and a few additional fixes). It cannot be the root directory (/).

If the directory already exists, it must be writable by the cluster administrator (lsfadmin).

If you need to change the directory after installation, edit PATCH_BACKUP_DIR in LSF_TOP/patch.conf and move the saved backup files to the new directory manually.

Example

PATCH_BACKUP_DIR="/usr/share/lsf/patch/backup"

Default

LSF_TOP/patch/backup

PATCH_HISTORY_DIR

Syntax

PATCH_HISTORY_DIR="/*path*"

Description

Full path to the patch history directory. This parameter is used when you install a new cluster for the first time, and is ignored for all other cases.

It cannot be the root directory (/). If the directory already exists, it must be writable by lsfadmin.

The location is saved as PATCH_HISTORY_DIR in LSF_TOP/patch.conf. Do not change the directory after installation.

Example

PATCH_BACKUP_DIR="/usr/share/lsf/patch"

Default

LSF_TOP/patch

PERF_HOST

Syntax

PERF_HOST="*host_name*"

Description

Dedicated host for PERF daemons. Required if EGO_PERF_CONTROL="N". To allow failover, we recommend that you leave this parameter undefined when EGO control is enabled for the PERF daemons.

Specify the name of one host that will run PERF daemons: plc, jobdt, and purger. If EGO controls PERF daemons, you must specify a host from LSF_MASTER_LIST.

> **Note:**
>
> This parameter only takes effect when you install the Platform
> Management Console package for the first time.

Example

PERF_HOST="hostp"

Default

Undefined.

PMC_HOST

Syntax

PMC_HOST="*host_name*"

Description

Dedicated host for Platform Management Console. Required if EGO_PMC_CONTROL="N".
To allow failover, we recommend that you leave this parameter undefined when EGO control
is enabled for the Platform Management Console.

Specify the name of one host that will always run the Platform Management Console. If EGO
controls PMC, you must specify a host from LSF_MASTER_LIST.

> **Note:**
>
> This parameter only takes effect when you install the Platform
> Management Console package for the first time.

Example

PMC_HOST="hostg"

Default

Undefined.

lim.acct

The `lim.acct` file is the log file for Load Information Manager (LIM). Produced by `lsmon`, `lim.acct` contains host load information collected and distributed by LIM.

lim.acct structure

The first line of `lim.acct` contains a list of load index names separated by spaces. This list of load index names can be specified in the `lsmon` command line. The default list is "r15s r1m r15m ut pg ls it swp mem tmp". Subsequent lines in the file contain the host's load information at the time the information was recorded.

Fields

Fields are ordered in the following sequence:

time (%ld)

> The time when the load information is written to the log file

host name (%s)

> The name of the host.

status of host (%d)

> An array of integers. The first integer marks the operation status of the host. Additional integers are used as a bit map to indicate load status of the host. An integer can be used for 32 load indices. If the number of user defined load indices is not more than 21, only one integer is used for both built-in load indices and external load indices. See the hostload structure in `ls_load(3)` for the description of these fields.

indexvalue (%f)

> A sequence of load index values. Each value corresponds to the index name in the first line of `lim.acct`. The order in which the index values are listed is the same as the order of the index names.

lsb.acct

The lsb.acct file is the batch job log file of LSF. The master batch daemon (see mbatchd(8)) generates a record for each job completion or failure. The record is appended to the job log file lsb.acct.

The file is located in LSB_SHAREDIR/cluster_name/logdir, where LSB_SHAREDIR must be defined in lsf.conf (5) and cluster_name is the name of the LSF cluster, as returned by lsid(1). See mbatchd(8) for the description of LSB_SHAREDIR.

The bacct command uses the current lsb.acct file for its output.

lsb.acct structure

The job log file is an ASCII file with one record per line. The fields of a record are separated by blanks. If the value of some field is unavailable, a pair of double quotation marks ("") is logged for character string, 0 for time and number, and -1 for resource usage.

Configuring automatic archiving

The following parameters in lsb.params affect how records are logged to lsb.acct:

ACCT_ARCHIVE_AGE=*days*

> Enables automatic archiving of LSF accounting log files, and specifies the archive interval. LSF archives the current log file if the length of time from its creation date exceeds the specified number of days.

> By default there is no limit to the age of lsb.acct.

ACCT_ARCHIVE_SIZE=*kilobytes*

> Enables automatic archiving of LSF accounting log files, and specifies the archive threshold. LSF archives the current log file if its size exceeds the specified number of kilobytes.

> By default, there is no limit to the size of lsb.acct.

ACCT_ARCHIVE_TIME=*hh:mm*

> Enables automatic archiving of LSF accounting log file lsb.acct, and specifies the time of day to archive the current log file.

> By default, no time is set for archiving lsb.acct.

MAX_ACCT_ARCHIVE_FILE=*integer*

> Enables automatic deletion of archived LSF accounting log files and specifies the archive limit.

> By default, lsb.acct.*n* files are not automatically deleted.

Records and fields

> The fields of a record are separated by blanks. The first string of an event record indicates its type. The following types of events are recorded:

> * JOB_FINISH

- EVENT_ADRSV_FINISH

JOB_FINISH

A job has finished.

If LSF_HPC_EXTENSIONS="SHORT_EVENTFILE" is specified in lsf.conf, older daemons and commands (pre-LSF Version 6.0) cannot recognize the lsb.acct file format.

The fields in order of occurrence are:

Event type (%s)

Which is "JOB_FINISH"

Version Number (%s)

Version number of the log file format

Event Time (%d)

Time the event was logged (in seconds since the epoch)

jobId (%d)

ID for the job

userId (%d)

UNIX user ID of the submitter

options (%d)

Bit flags for job processing

numProcessors (%d)

Number of processors initially requested for execution

submitTime (%d)

Job submission time

beginTime (%d)

Job start time – the job should be started at or after this time

termTime (%d)

Job termination deadline – the job should be terminated by this time

startTime (%d)

Job dispatch time – time job was dispatched for execution

userName (%s)

User name of the submitter

queue (%s)

Name of the job queue to which the job was submitted

resReq (%s)

Resource requirement specified by the user

dependCond (%s)

Job dependency condition specified by the user

preExecCmd (%s)

Pre-execution command specified by the user

fromHost (%s)

Submission host name

cwd (%s)

Current working directory (up to 4094 characters for UNIX or 512 characters for Windows)

inFile (%s)

Input file name (up to 4094 characters for UNIX or 512 characters for Windows)

outFile (%s)

output file name (up to 4094 characters for UNIX or 512 characters for Windows)

errFile (%s)

Error output file name (up to 4094 characters for UNIX or 512 characters for Windows)

jobFile (%s)

Job script file name

numAskedHosts (%d)

Number of host names to which job dispatching will be limited

askedHosts (%s)

List of host names to which job dispatching will be limited (%s for each); nothing is logged to the record for this value if the last field value is 0. If there is more than one host name, then each additional host name will be returned in its own field

numExHosts (%d)

Number of processors used for execution

If LSF_HPC_EXTENSIONS="SHORT_EVENTFILE" is specified in lsf.conf, the value of this field is the number of .hosts listed in the execHosts field.

Logged value reflects the allocation at job finish time.

execHosts (%s)

List of execution host names (%s for each); nothing is logged to the record for this value if the last field value is 0.

If LSF_HPC_EXTENSIONS="SHORT_EVENTFILE" is specified in lsf.conf, the value of this field is logged in a shortened format.

The logged value reflects the allocation at job finish time.

jStatus (%d)

Job status. The number 32 represents EXIT, 64 represents DONE

hostFactor (%f)

CPU factor of the first execution host.

jobName (%s)

Job name (up to 4094 characters for UNIX or 512 characters for Windows).

command (%s)

Complete batch job command specified by the user (up to 4094 characters for UNIX or 512 characters for Windows).

lsfRusage (%f)

The following fields contain resource usage information for the job (see `getrusage` (2)). If the value of some field is unavailable (due to job exit or the difference among the operating systems), -1 will be logged. Times are measured in seconds, and sizes are measured in KB.

ru_utime (%f)

User time used

ru_stime (%f)

System time used

ru_maxrss (%f)

Maximum shared text size

ru_ixrss (%f)

Integral of the shared text size over time (in KB seconds)

ru_ismrss (%f)

Integral of the shared memory size over time (valid only on Ultrix)

ru_idrss (%f)

Integral of the unshared data size over time

ru_isrss (%f)

Integral of the unshared stack size over time

ru_minflt (%f)

Number of page reclaims

ru_majflt (%f)

Number of page faults

ru_nswap (%f)

Number of times the process was swapped out

ru_inblock (%f)

Number of block input operations

ru_oublock (%f)

Number of block output operations

ru_ioch (%f)

Number of characters read and written (valid only on HP-UX)

ru_msgsnd (%f)

Number of System V IPC messages sent

ru_msgrcv (%f)

Number of messages received

ru_nsignals (%f)

Number of signals received

ru_nvcsw (%f)

Number of voluntary context switches

ru_nivcsw (%f)

Number of involuntary context switches

ru_exutime (%f)

Exact user time used (valid only on ConvexOS)

mailUser (%s)

Name of the user to whom job related mail was sent

projectName (%s)

LSF project name

exitStatus (%d)

UNIX exit status of the job

maxNumProcessors (%d)

Maximum number of processors specified for the job

loginShell (%s)

Login shell used for the job

timeEvent (%s)

Time event string for the job - JobScheduler only

idx (%d)

Job array index

maxRMem (%d)

Maximum resident memory usage in the unit specified by LSF_UNIT_FOR_LIMITS in `lsf.conf` of all processes in the job

maxRSwap (%d)

Maximum virtual memory usage in the unit specified by LSF_UNIT_FOR_LIMITS in `lsf.conf` of all processes in the job

inFileSpool (%s)

Spool input file (up to 4094 characters for UNIX or 512 characters for Windows)

commandSpool (%s)

Spool command file (up to 4094 characters for UNIX or 512 characters for Windows)

rsvId %s

Advance reservation ID for a user group name less than 120 characters long; for example, "`user2#0`"

If the advance reservation user group name is longer than 120 characters, the rsvId field output appears last.

sla (%s)

SLA service class name under which the job runs

exceptMask (%d)

Job exception handling

Values:
- `J_EXCEPT_OVERRUN 0x02`
- `J_EXCEPT_UNDERUN 0x04`
- `J_EXCEPT_IDLE 0x80`

additionalInfo (%s)

Placement information of HPC jobs

exitInfo (%d)

Job termination reason, mapped to corresponding termination keyword displayed by `bacct.`

warningTimePeriod (%d)

Job warning time period in seconds

warningAction (%s)

Job warning action

chargedSAAP (%s)

SAAP charged to a job

licenseProject (%s)

LSF License Scheduler project name

options3 (%d)

>   Bit flags for job processing

app (%s)

>   Application profile name

postExecCmd (%s)

>   Post-execution command to run on the execution host after the job finishes

runtimeEstimation (%d)

>   Estimated run time for the job

jobGroupName (%s)

>   Job group name

resizeNotifyCmd

>   Resize notification command to be invoked on the first execution host upon a resize request.

lastResizeTime

>   Last resize time. The latest wall clock time when a job allocation is changed.

rsvId %s

>   Advance reservation ID for a user group name more than 120 characters long.
>
>   If the advance reservation user group name is longer than 120 characters, the rsvId field output appears last.

EVENT_ADRSV_FINISH

>   An advance reservation has expired. The fields in order of occurrence are:

Event type (%s)

>   Which is "EVENT_ADRSV_FINISH"

Version Number (%s)

>   Version number of the log file format

Event Logging Time (%d)

>   Time the event was logged (in seconds since the epoch); for example, "1038942015"

Reservation Creation Time (%d)

>   Time the advance reservation was created (in seconds since the epoch); for example, "1038938898"

Reservation Type (%d)

>   Type of advance reservation request:
>
>   • User reservation (RSV_OPTION_USER, defined as 0x001)
>   • User group reservation (RSV_OPTION_GROUP, defined as 0x002)

- System reservation (RSV_OPTION_SYSTEM, defined as 0x004)
- Recurring reservation (RSV_OPTION_RECUR, defined as 0x008)

For example, "9" is a recurring reservation created for a user.

Creator ID (%d)

UNIX user ID of the reservation creator; for example, "30408"

Reservation ID (rsvId %s)

For example, "user2#0"

User Name (%s)

User name of the reservation user; for example, "user2"

Time Window (%s)

Time window of the reservation:

- One-time reservation in seconds since the epoch; for example, "1033761000- 1033761600"
- Recurring reservation; for example, "17: 50- 18: 00"

Creator Name (%s)

User name of the reservation creator; for example, "user1"

Duration (%d)

Duration of the reservation, in hours, minutes, seconds; for example, "600" is 6 hours, 0 minutes, 0 seconds

Number of Resources (%d)

Number of reserved resource pairs in the resource list; for example "2" indicates 2 resource pairs ("hostA 1 hostB 1")

Host Name (%s)

Reservation host name; for example, "hostA"

Number of CPUs (%d)

Number of reserved CPUs; for example "1"

JOB_RESIZE

When there is an allocation change, LSF logs the event after mbatchd receives "JOB_RESIZE_NOTIFY_DONE" event. From lastResizeTime and eventTime, people can easily calculate the duration of previous job allocation. The fields in order of occurrence are:

Version number (%s)

The version number.

Event Time (%d)

Time the event was logged (in seconds since the epoch).

jobId (%d)

ID for the job.

tdx (%d)

Job array index.

startTime (%d)

The start time of the running job.

userId (%d)

UNIX user ID of the user invoking the command

userName (%s)

User name of the submitter

resizeType (%d)

Resize event type, 0, grow, 1 shrink.

lastResizeTime(%d)

The wall clock time when job allocation is changed previously. The first lastResizeTime is the job start time.

numExecHosts (%d)

The number of execution hosts before allocation is changed. Support LSF_HPC_EXTENSIONS="SHORT_EVENTFILE".

execHosts (%s)

Execution host list before allocation is changed. Support LSF_HPC_EXTENSIONS="SHORT_EVENTFILE".

numResizeHosts (%d)

Number of processors used for execution during resize. If LSF_HPC_EXTENSIONS="SHORT_EVENTFILE" is specified in `lsf.conf`, the value of this field is the number of hosts listed in short format.

resizeHosts (%s)

List of execution host names during resize. If LSF_HPC_EXTENSIONS="SHORT_EVENTFILE" is specified in `lsf.conf`, the value of this field is logged in a shortened format.

lsb.applications

The lsb.applications file defines application profiles. Use application profiles to define common parameters for the same type of jobs, including the execution requirements of the applications, the resources they require, and how they should be run and managed.

This file is optional. Use the DEFAULT_APPLICATION parameter in lsb.params to specify a default application profile for all jobs. LSF does not automatically assign a default application profile.

This file is installed by default in LSB_CONFDIR/cluster_name/configdir.

Changing lsb.applications configuration

After making any changes to lsb.applications, run badmin reconfig to reconfigure mbatchd. Configuration changes apply to pending jobs only. Running jobs are not affected.

lsb.applications structure

Each application profile definition begins with the line Begin Application and ends with the line End Application. The application name must be specified. All other parameters are optional.

Example

```
Begin Application

NAME        = catia

DESCRIPTION  = CATIA V5

CPULIMIT     = 24:0/hostA        # 24 hours of host hostA

FILELIMIT    = 20000

DATALIMIT    = 20000             # jobs data segment limit

CORELIMIT    = 20000

PROCLIMIT    = 5                 # job processor limit

REQUEUE_EXIT_VALUES = 55 34 78

End Application
```

See the lsb.applications template file for additional application profile examples.

Parameters

- ABS_RUNLIMIT
- BIND_JOB
- CHKPNT_DIR
- CHKPNT_INITPERIOD
- CHKPNT_PERIOD
- CHKPNT_METHOD
- CHUNK_JOB_SIZE
- CORELIMIT
- CPULIMIT
- DATALIMIT
- DESCRIPTION
- DJOB_COMMFAIL_ACTION

- DJOB_ENV_SCRIPT
- DJOB_HB_INTERVAL
- DJOB_RESIZE_GRACE_PERIOD
- DJOB_RU_INTERVAL
- JOB_INCLUDE_POSTPROC
- JOB_POSTPROC_TIMEOUT
- FILELIMIT
- JOB_STARTER
- LOCAL_MAX_PREEXEC_RETRY
- MAX_JOB_PREEMPT
- MAX_JOB_REQUEUE
- MAX_PREEXEC_RETRY
- MEMLIMIT
- MEMLIMIT_TYPE
- MIG
- NAME
- NO_PREEMPT_FINISH_TIME
- NO_PREEMPT_RUN_TIME
- PERSISTENT_HOST_ORDER
- POST_EXEC
- PRE_EXEC
- PROCESSLIMIT
- PROCLIMIT
- REMOTE_MAX_PREEXEC_RETRY
- REQUEUE_EXIT_VALUES
- RERUNNABLE
- RES_REQ
- RESIZABLE_JOBS
- RESIZE_NOTIFY_CMD
- RESUME_CONTROL
- RTASK_GONE_ACTION
- RUNLIMIT
- RUNTIME
- STACKLIMIT
- SUCCESS_EXIT_VALUES
- SUSPEND_CONTROL
- SWAPLIMIT
- TERMINATE_CONTROL
- THREADLIMIT
- USE_PAM_CREDS

ABS_RUNLIMIT

Syntax

ABS_RUNLIMIT=y | Y

Description

If set, absolute (wall-clock) run time is used instead of normalized run time for all jobs submitted with the following values:

- Run time limit specified by the -W option of bsub
- RUNLIMIT queue-level parameter in lsb.queues
- RUNLIMIT application-level parameter in lsb.applications
- RUNTIME parameter in lsb.applications

The runtime estimates and limits are not normalized by the host CPU factor.

Default

Not defined. Run limit and runtime estimate are normalized.

BIND_JOB

Syntax

BIND_JOB=NONE | **BALANCE** | **PACK** | **ANY** | **USER** | **USER_CPU_LIST**

Description

Specifies the processor binding policy for sequential and parallel job processes that run on a single host. On Linux execution hosts that support this feature, job processes are hard bound to selected processors.

If processor binding feature is not configured with the BIND_JOB parameter in an application profile in lsb.applications, the lsf.conf configuration setting takes effect. The application profile configuration for processor binding overrides the lsf.conf configuration.

For backwards compatibility:

- BIND_JOB=Y is interpreted as BIND_JOB=BALANCE
- BIND_JOB=N is interpreted as BIND_JOB=NONE

Supported platforms

Linux with kernel version 2.6 or higher

Default

Not defined. Processor binding is disabled.

CHKPNT_DIR

Syntax

CHKPNT_DIR=_chkpnt_dir_

Description

Specifies the checkpoint directory for automatic checkpointing for the application. To enable automatic checkpoint for the application profile, administrators must specify a checkpoint directory in the configuration of the application profile.

If CHKPNT_PERIOD, CHKPNT_INITPERIOD or CHKPNT_METHOD was set in an application profile but CHKPNT_DIR was not set, a warning message is issued and and those settings are ignored.

The checkpoint directory is the directory where the checkpoint files are created. Specify an absolute path or a path relative to the current working directory for the job. Do not use environment variables in the directory path.

If checkpoint-related configuration is specified in both the queue and an application profile, the application profile setting overrides queue level configuration.

If checkpoint-related configuration is specified in the queue, application profile, and at job level:

- Application-level and job-level parameters are merged. If the same parameter is defined at both job-level and in the application profile, the job-level value overrides the application profile value.
- The merged result of job-level and application profile settings override queue-level configuration.

To enable checkpointing of MultiCluster jobs, define a checkpoint directory in an application profile (CHKPNT_DIR, CHKPNT_PERIOD, CHKPNT_INITPERIOD, CHKPNT_METHOD in lsb.applications) of both submission cluster and execution cluster. LSF uses the directory specified in the execution cluster.

Checkpointing is not supported if a job runs on a leased host.

The file path of the checkpoint directory can contain up to 4000 characters for UNIX and Linux, or up to 255 characters for Windows, including the directory and file name.

Default

Not defined

CHKPNT_INITPERIOD

Syntax

CHKPNT_INITPERIOD=_init_chkpnt_period_

Description

Specifies the initial checkpoint period in minutes. CHKPNT_DIR must be set in the application profile for this parameter to take effect. The periodic checkpoint specified by CHKPNT_PERIOD does not happen until the initial period has elapse.

Specify a positive integer.

Job-level command line values override the application profile configuration.

If administrators specify an initial checkpoint period and do not specify a checkpoint period (CHKPNT_PERIOD), the job will only checkpoint once.

If the initial checkpoint period if a job is specified, and you run bchkpnt to checkpoint the job at a time before the initial checkpoint period, the initial checkpoint period is not changed by bchkpnt. The first automatic checkpoint still happens after the specified number of minutes.

Default

Not defined

CHKPNT_PERIOD

Syntax

CHKPNT_PERIOD=*chkpnt_period*

Description

Specifies the checkpoint period for the application in minutes. CHKPNT_DIR must be set in the application profile for this parameter to take effect. The running job is checkpointed automatically every checkpoint period.

Specify a positive integer.

Job-level command line values override the application profile and queue level configurations. Application profile level configuration overrides the queue level configuration.

Default

Not defined

CHKPNT_METHOD

Syntax

CHKPNT_METHOD=*chkpnt_method*

Description

Specifies the checkpoint method. CHKPNT_DIR must be set in the application profile for this parameter to take effect. Job-level command line values override the application profile configuration.

Default

Not defined

CHUNK_JOB_SIZE

Syntax

CHUNK_JOB_SIZE=*integer*

Description

Chunk jobs only. Allows jobs submitted to the same application profile to be chunked together and specifies the maximum number of jobs allowed to be dispatched together in a chunk. Specify a positive integer greater than or equal to 1.

All of the jobs in the chunk are scheduled and dispatched as a unit, rather than individually.

Specify CHUNK_JOB_SIZE=1 to disable job chunking for the application. This value overrides chunk job dispatch configured in the queue.

Use the CHUNK_JOB_SIZE parameter to configure application profiles that chunk small, short-running jobs. The ideal candidates for job chunking are jobs that have the same host and resource requirements and typically take 1 to 2 minutes to run.

The ideal candidates for job chunking are jobs that have the same host and resource requirements and typically take 1 to 2 minutes to run.

Job chunking can have the following advantages:

* Reduces communication between sbatchd and mbatchd and reduces scheduling overhead in mbschd.
* Increases job throughput in mbatchd and CPU utilization on the execution hosts.

However, throughput can deteriorate if the chunk job size is too big. Performance may decrease on profiles with CHUNK_JOB_SIZE greater than 30. You should evaluate the chunk job size on your own systems for best performance.

With MultiCluster job forwarding model, this parameter does not affect MultiCluster jobs that are forwarded to a remote cluster.

Compatibility

This parameter is ignored and jobs are not chunked under the following conditions:

* CPU limit greater than 30 minutes (CPULIMIT parameter in lsb.queues or lsb.applications)
* Run limit greater than 30 minutes (RUNLIMIT parameter in lsb.queues or lsb.applications)
* Runtime estimate greater than 30 minutes (RUNTIME parameter in lsb.applications)

If CHUNK_JOB_DURATION is set in lsb.params, chunk jobs are accepted regardless of the value of CPULIMIT, RUNLIMIT or RUNTIME.

Default

Not defined

CORELIMIT

Syntax

CORELIMIT=_integer_

Description

The per-process (soft) core file size limit for all of the processes belonging to a job from this application profile (see getrlimit(2)). Application-level limits override any default limit specified in the queue, but must be less than the hard limit of the submission queue. Job-level core limit (bsub -C) overrides queue-level and application-level limits.

By default, the limit is specified in KB. Use LSF_UNIT_FOR_LIMITS in lsf.conf to specify a larger unit for the the limit (MB, GB, TB, PB, or EB).

Default

Unlimited

CPULIMIT

Syntax

CPULIMIT=[[*hour*:]*minute*[*/host_name* | */host_model*]

Description

Normalized CPU time allowed for all processes of a job running in the application profile. The name of a host or host model specifies the CPU time normalization host to use.

Limits the total CPU time the job can use. This parameter is useful for preventing runaway jobs or jobs that use up too many resources.

When the total CPU time for the whole job has reached the limit, a SIGXCPU signal is sent to all processes belonging to the job. If the job has no signal handler for SIGXCPU, the job is killed immediately. If the SIGXCPU signal is handled, blocked, or ignored by the application, then after the grace period expires, LSF sends SIGINT, SIGTERM, and SIGKILL to the job to kill it.

If a job dynamically spawns processes, the CPU time used by these processes is accumulated over the life of the job.

Processes that exist for fewer than 30 seconds may be ignored.

By default, jobs submitted to the application profile without a job-level CPU limit (bsub -c) are killed when the CPU limit is reached. Application-level limits override any default limit specified in the queue.

The number of minutes may be greater than 59. For example, three and a half hours can be specified either as 3:30 or 210.

If no host or host model is given with the CPU time, LSF uses the default CPU time normalization host defined at the queue level (DEFAULT_HOST_SPEC in lsb.queues) if it has been configured, otherwise uses the default CPU time normalization host defined at the cluster level (DEFAULT_HOST_SPEC in lsb.params) if it has been configured, otherwise uses the host with the largest CPU factor (the fastest host in the cluster).

On Windows, a job that runs under a CPU time limit may exceed that limit by up to SBD_SLEEP_TIME. This is because sbatchd periodically checks if the limit has been exceeded.

On UNIX systems, the CPU limit can be enforced by the operating system at the process level.

You can define whether the CPU limit is a per-process limit enforced by the OS or a per-job limit enforced by LSF with LSB_JOB_CPULIMIT in lsf.conf.

Default

Unlimited

DATALIMIT

Syntax

DATALIMIT=*integer*

Description

The per-process (soft) data segment size limit (in KB) for all of the processes belonging to a job running in the application profile (see getrlimit(2)).

By default, jobs submitted to the application profile without a job-level data limit (bsub -D) are killed when the data limit is reached. Application-level limits override any default limit specified in the queue, but must be less than the hard limit of the submission queue.

Default

Unlimited

DESCRIPTION

Syntax

DESCRIPTION=*text*

Description

Description of the application profile. The description is displayed by bapp -l.

The description should clearly describe the service features of the application profile to help users select the proper profile for each job.

The text can include any characters, including white space. The text can be extended to multiple lines by ending the preceding line with a backslash (\). The maximum length for the text is 512 characters.

DJOB_COMMFAIL_ACTION

Syntax

DJOB_COMMFAIL_ACTION="KILL_TASKS"

Description

Defines the action LSF should take if it detects a communication failure with one or more remote parallel or distributed tasks. If defined, LSF tries to kill all the current tasks of a parallel or distributed job associated with the communication failure. If not defined, LSF terminates all tasks and shuts down the entire job.

This parameter only applies to the blaunch distributed application framework.

When defined in an application profile, the LSB_DJOB_COMMFAIL_ACTION variable is set when running bsub -app for the specified application.

Default

Not defined. Terminate all tasks, and shut down the entire job.

DJOB_DISABLED

Syntax

DJOB_DISABLED=Y | N

Description

Disables the bl aunch distributed application framework.

Default

Not defined. Distributed application framework is enabled.

DJOB_ENV_SCRIPT

Syntax

DJOB_ENV_SCRIPT=*script_name*

Description

Defines the name of a user-defined script for setting and cleaning up the parallel or distributed job environment.

The specified script must support a setup argument and a cleanup argument. The script is executed by LSF with the setup argument before launching a parallel or distributed job, and with argument cleanup after the job is finished.

The script runs as the user, and is part of the job.

If a full path is specified, LSF uses the path name for the execution. Otherwise, LSF looks for the executable from $LSF_BINDIR.

This parameter only applies to the bl aunch distributed application framework.

When defined in an application profile, the LSB_DJOB_ENV_SCRIPT variable is set when running bsub - app for the specified application.

The command path can contain up to 4094 characters for UNIX and Linux, or up to 255 characters for Windows, including the directory, file name, and expanded values for %J (*job_ID*) and %I (*index_ID*).

Default

Not defined.

DJOB_HB_INTERVAL

Syntax

DJOB_HB_INTERVAL=*seconds*

Description

Value in seconds used to calculate the heartbeat interval between the task RES and job RES of a parallel or distributed job.

This parameter only applies to the bl aunch distributed application framework.

When DJOB_HB_INTERVAL is specified, the interval is scaled according to the number of tasks in the job:

max(DJOB_HB_INTERVAL, 10) + *host_factor*

where

host_factor = 0.01 * *number of hosts allocated for the job*

Default

Not defined. Interval is equal to SBD_SLEEP_TIME in l sb. params, where the default value of SBD_SLEEP_TIME is 30 seconds.

DJOB_RESIZE_GRACE_PERIOD

Syntax

DJOB_RESIZE_GRACE_PERIOD = *seconds*

Description

When a resizable job releases resources, the LSF distributed parallel job framework terminates running tasks if a host has been completely removed. A DJOB_RESIZE_GRACE_PERIOD defines a grace period in seconds for the application to clean up tasks itself before LSF forcibly terminates them.

Default

No grace period.

DJOB_RU_INTERVAL

Syntax

DJOB_RU_INTERVAL=*seconds*

Description

Value in seconds used to calculate the resource usage update interval for the tasks of a parallel or distributed job.

This parameter only applies to the bl aunch distributed application framework.

When DJOB_RU_INTERVAL is specified, the interval is scaled according to the number of tasks in the job:

max(DJOB_RU_INTERVAL, 10) + *host_factor*

where

host_factor = 0.01 * *number of hosts allocated for the job*

Default

Not defined. Interval is equal to SBD_SLEEP_TIME in l sb. params, where the default value of SBD_SLEEP_TIME is 30 seconds.

JOB_INCLUDE_POSTPROC

Syntax

JOB_INCLUDE_POSTPROC=Y | N

Description

Specifies whether LSF includes the post-execution processing of the job as part of the job. When set to Y:

- Prevents a new job from starting on a host until post-execution processing is finished on that host
- Includes the CPU and run times of post-execution processing with the job CPU and run times
- sbatchd sends both job finish status (DONE or EXIT) and post-execution processing status (POST_DONE or POST_ERR) to mbatchd at the same time

The variable LSB_JOB_INCLUDE_POSTPROC in the user environment overrides the value of JOB_INCLUDE_POSTPROC in an application profile in lsb.applications. JOB_INCLUDE_POSTPROC in an application profile in lsb.applications overrides the value of JOB_INCLUDE_POSTPROC in lsb.params.

For SGI cpusets, if JOB_INCLUDE_POSTPROC=Y, LSF does not release the cpuset until post-execution processing has finished, even though post-execution processes are not attached to the cpuset.

Default

N. Post-execution processing is not included as part of the job, and a new job can start on the execution host before post-execution processing finishes.

JOB_POSTPROC_TIMEOUT

Syntax

JOB_POSTPROC_TIMEOUT=_minutes_

Description

Specifies a timeout in minutes for job post-execution processing. The specified timeout must be greater than zero

If post-execution processing takes longer than the timeout, sbatchd reports that post-execution has failed (POST_ERR status), and kills the process group of the job's post-execution processes. Only the parent process of the post-execution command is killed when the timeout expires. The child processes of the post-execution command are not killed.

If JOB_INCLUDE_POSTPROC=Y, and sbatchd kills the post-execution processes because the timeout has been reached, the CPU time of the post-execution processing is set to 0, and the job's CPU time does not include the CPU time of post-execution processing.

JOB_POSTPROC_TIMEOUT defined in an application profile in lsb.applications overrides the value in lsb.params. JOB_POSTPROC_TIMEOUT cannot be defined in user environment.

Default

Not defined. Post-execution processing does not time out.

FILELIMIT

Syntax

FILELIMIT=_integer_

Description

The per-process (soft) file size limit (in KB) for all of the processes belonging to a job running in the application profile (see getrlimit(2)). Application-level limits override any default limit specified in the queue, but must be less than the hard limit of the submission queue.

Default

Unlimited

JOB_STARTER

Syntax

JOB_STARTER=*starter* [*starter*] [**"%USRCMD"**] [*starter*]

Description

Creates a specific environment for submitted jobs prior to execution. An application-level job starter overrides a queue-level job starter.

starter is any executable that can be used to start the job (i.e., can accept the job as an input argument). Optionally, additional strings can be specified.

By default, the user commands run after the job starter. A special string, %USRCMD, can be used to represent the position of the user's job in the job starter command line. The %USRCMD string and any additional commands must be enclosed in quotation marks (" ").

Example

```
JOB_STARTER=csh -c "%USRCMD;sleep 10"
```

In this case, if a user submits a job
bsub myjob arguments

the command that actually runs is:
```
csh -c "myjob arguments;sleep 10"
```

Default

Not defined. No job starter is used,

LOCAL_MAX_PREEXEC_RETRY

Syntax

LOCAL_MAX_PREEXEC_RETRY=*integer*

Description

The maximum number of times to attempt the pre-execution command of a job on the local cluster.

Valid values

0 < MAX_PREEXEC_RETRY < INFINIT_INT

INFINIT_INT is defined in lsf.h.

Default

Not defined. The number of preexec retry times is unlimited

MAX_JOB_PREEMPT

Syntax

MAX_JOB_PREEMPT=*integer*

Description

The maximum number of times a job can be preempted. Applies to queue-level jobs only.

Valid values

0 < MAX_JOB_PREEMPT < INFINIT_INT

INFINIT_INT is defined in l sf . h.

Default

Not defined. The number of preemption times is unlimited.

MAX_JOB_REQUEUE

Syntax

MAX_JOB_REQUEUE=*integer*

Description

The maximum number of times to requeue a job automatically.

Valid values

0 < MAX_JOB_REQUEUE < INFINIT_INT

INFINIT_INT is defined in l sf . h.

Default

Not defined. The number of requeue times is unlimited

MAX_PREEXEC_RETRY

Syntax

MAX_PREEXEC_RETRY=*integer*

Description

MultiCluster job forwarding model only. The maximum number of times to attempt the pre-execution command of a job from a remote cluster.

If the job's pre-execution command fails all attempts, the job is returned to the submission cluster.

Valid values

0 < MAX_PREEXEC_RETRY < INFINIT_INT

INFINIT_INT is defined in lsf.h.

Default

5

MEMLIMIT

Syntax

MEMLIMIT=*integer*

Description

The per-process (soft) process resident set size limit for all of the processes belonging to a job running in the application profile.

Sets the maximum amount of physical memory (resident set size, RSS) that may be allocated to a process.

By default, the limit is specified in KB. Use LSF_UNIT_FOR_LIMITS in lsf.conf to specify a larger unit for the the limit (MB, GB, TB, PB, or EB).

By default, jobs submitted to the application profile without a job-level memory limit are killed when the memory limit is reached. Application-level limits override any default limit specified in the queue, but must be less than the hard limit of the submission queue.

LSF has two methods of enforcing memory usage:

* OS Memory Limit Enforcement
* LSF Memory Limit Enforcement

OS memory limit enforcement

OS memory limit enforcement is the default MEMLIMIT behavior and does not require further configuration. OS enforcement usually allows the process to eventually run to completion. LSF passes MEMLIMIT to the OS, which uses it as a guide for the system scheduler and memory allocator. The system may allocate more memory to a process if there is a surplus. When memory is low, the system takes memory from and lowers the scheduling priority (re-nice) of a process that has exceeded its declared MEMLIMIT. Only available on systems that support RLIMIT_RSS for setrlimit().

Not supported on:

* Sun Solaris 2.x
* Windows

LSF memory limit enforcement

To enable LSF memory limit enforcement, set LSB_MEMLIMIT_ENFORCE in lsf.conf to y. LSF memory limit enforcement explicitly sends a signal to kill a running process once it has allocated memory past MEMLIMIT.

You can also enable LSF memory limit enforcement by setting LSB_JOB_MEMLIMIT in lsf.conf to y. The difference between LSB_JOB_MEMLIMIT set to y and

LSB_MEMLIMIT_ENFORCE set to y is that with LSB_JOB_MEMLIMIT, only the per-job memory limit enforced by LSF is enabled. The per-process memory limit enforced by the OS is disabled. With LSB_MEMLIMIT_ENFORCE set to y, both the per-job memory limit enforced by LSF and the per-process memory limit enforced by the OS are enabled.

Available for all systems on which LSF collects total memory usage.

Default

Unlimited

MEMLIMIT_TYPE

Syntax

MEMLIMIT_TYPE=JOB [**PROCESS**] [**TASK**]

MEMLIMIT_TYPE=PROCESS [**JOB**] [**TASK**]

MEMLIMIT_TYPE=TASK [**PROCESS**] [**JOB**]

Description

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

By specifying a value in the application profile, you overwrite these three parameters: LSB_JOB_MEMLIMIT, LSB_MEMLIMIT_ENFORCE, LSF_HPC_EXTENSIONS (TASK_MEMLIMIT).

Note:

A task list is a list in LSF that keeps track of the default resource requirements for different applications and task eligibility for remote execution.

- PROCESS: Applies a memory limit by OS process, which is enforced by the OS on the slave machine (where the job is running). When the memory allocated to one process of the job exceeds the memory limit, LSF kills the job.
- TASK: Applies a memory limit based on the task list file. It is enforced by LSF. LSF terminates the entire parallel job if any single task exceeds the limit setting for memory and swap limits.
- JOB: Applies a memory limit identified in a job and enforced by LSF. When the sum of the memory allocated to all processes of the job exceeds the memory limit, LSF kills the job.
- PROCESS TASK: Enables both process-level memory limit enforced by OS and task-level memory limit enforced by LSF.
- PROCESS JOB: Enables both process-level memory limit enforced by OS and job-level memory limit enforced by LSF.
- TASK JOB: Enables both task-level memory limit enforced by LSF and job-level memory limit enforced by LSF.
- PROCESS TASK JOB: Enables process-level memory limit enforced by OS, task-level memory limit enforced by LSF, and job-level memory limit enforced by LSF.

Default

Not defined. The memory limit-level is still controlled by
LSF_HPC_EXTENSIONS=TASK_MEMLIMIT, LSB_JOB_MEMLIMIT,
LSB_MEMLIMIT_ENFORCE

MIG

Syntax

MIG=*minutes*

Description

Enables automatic job migration and specifies the migration threshold for checkpointable or
rerunnable jobs, in minutes.

LSF automatically migrates jobs that have been in the SSUSP state for more than the specified
number of minutes. A value of 0 specifies that a suspended job is migrated immediately. The
migration threshold applies to all jobs running on the host.

Job-level command line migration threshold overrides threshold configuration in application
profile and queue. Application profile configuration overrides queue level configuration.

When a host migration threshold is specified, and is lower than the value for the job, the queue,
or the application, the host value is used.

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

Does not affect MultiCluster jobs that are forwarded to a remote cluster.

Default

Not defined. LSF does not migrate checkpointable or rerunnable jobs automatically.

NAME

Syntax

NAME=*string*

Description

Required. Unique name for the application profile.

Specify any ASCII string up to 60 characters long. You can use letters, digits, underscores (_),
dashes (-), periods (.) or spaces in the name. The application profile name must be unique
within the cluster.

> **Note:**
>
> If you want to specify the ApplicationVersion in a JSDL file, include
> the version when you define the application profile name.
> Separate the name and version by a space, as shown in the
> following example:
> ```
> NAME=myapp 1.0
> ```

Default

You must specify this parameter to define an application profile. LSF does not automatically assign a default application profile name.

NO_PREEMPT_FINISH_TIME

Syntax

NO_PREEMPT_FINISH_TIME=*minutes* | *percentage*

Description

Prevents preemption of jobs that will finish within the specified number of minutes or the specified percentage of the estimated run time or run limit.

Specifies that jobs due to finish within the specified number of minutes or percentage of job duration should not be preempted, where *minutes* is wall-clock time, not normalized time. Percentage must be greater than 0 or less than 100% (between 1% and 99%).

For example, if the job run limit is 60 minutes and NO_PREEMPT_FINISH_TIME=10%, the job cannot be preempted after it running 54 minutes or longer.

If you specify percentage for NO_PREEMPT_RUN_TIME, requires a run time (bsub -We or RUNTIME in lsb.applications), or run limit to be specified for the job (bsub -W, or RUNLIMIT in lsb.queues, or RUNLIMIT in lsb.applications)

NO_PREEMPT_RUN_TIME

Syntax

NO_PREEMPT_RUN_TIME=*minutes* | *percentage*

Description

Prevents preemption of jobs that have been running for the specified number of minutes or the specified percentage of the estimated run time or run limit.

Specifies that jobs that have been running for the specified number of minutes or longer should not be preempted, where *minutes* is wall-clock time, not normalized time. Percentage must be greater than 0 or less than 100% (between 1% and 99%).

For example, if the job run limit is 60 minutes and NO_PREEMPT_RUN_TIME=50%, the job cannot be preempted after it running 30 minutes or longer.

If you specify percentage for NO_PREEMPT_RUN_TIME, requires a run time (bsub -We or RUNTIME in lsb.applications), or run limit to be specified for the job (bsub -W, or RUNLIMIT in lsb.queues, or RUNLIMIT in lsb.applications)

PERSISTENT_HOST_ORDER

Syntax

PERSISTENT_HOST_ORDER=Y | yes | N | no

Description

Applies when migrating parallel jobs in a multicluster environment. Setting PERSISTENT_HOST_ORDER=Y ensures that jobs are restarted on hosts based on

alphabetical names of the hosts, preventing them from being restarted on the same hosts that they ran on before migration.

Default

PERSISTENT_HOST_ORDER=N. Migrated jobs in a multicluster environment could run on the same hosts that they ran on before.

POST_EXEC

Syntax

POST_EXEC=*command*

Description

Enables post-execution processing at the application level. The POST_EXEC command runs on the execution host after the job finishes. Post-execution commands can be configured at the job, application, and queue levels.

If both application-level (POST_EXEC in lsb.applications) and job-level post-execution commands are specified, job level post-execution overrides application-level post-execution commands. Queue-level post-execution commands (POST_EXEC in lsb.queues) run after application-level post-execution and job-level post-execution commands.

The POST_EXEC command uses the same environment variable values as the job, and runs under the user account of the user who submits the job. To run post-execution commands under a different user account (such as root for privileged operations), configure the parameter LSB_PRE_POST_EXEC_USER in lsf.sudoers.

When a job exits with one of the application profile's REQUEUE_EXIT_VALUES, LSF requeues the job and sets the environment variable LSB_JOBPEND. The post-execution command runs after the requeued job finishes.

When the post-execution command is run, the environment variable LSB_JOBEXIT_STAT is set to the exit status of the job. If the execution environment for the job cannot be set up, LSB_JOBEXIT_STAT is set to 0 (zero).

The command path can contain up to 4094 characters for UNIX and Linux, or up to 255 characters for Windows, including the directory, file name, and expanded values for %J *(job_ID)* and %I (*index_ID*).

For UNIX:

- The pre- and post-execution commands run in the /tmp directory under /bin/sh -c, which allows the use of shell features in the commands. The following example shows valid configuration lines:

  ```
  PRE_EXEC= /usr/share/lsf/misc/testq_pre >> /tmp/pre.out
  ```

  ```
  POST_EXEC= /usr/share/lsf/misc/testq_post | grep -v "Hey!"
  ```
- LSF sets the PATH environment variable to

  ```
  PATH='/bin /usr/bin /sbin /usr/sbin'
  ```
- The stdin, stdout, and stderr are set to /dev/null
- To allow UNIX users to define their own post-execution commands, an LSF administrator specifies the environment variable $USER_POSTEXEC as the POST_EXEC command. A user then defines the post-execution command:

  ```
  setenv USER_POSTEXEC /path_name
  ```

Note:

The path name for the post-execution command must be an absolute path. Do not define POST_EXEC= $USER_POSTEXEC when LSB_PRE_POST_EXEC_USER=root.

For Windows:

- The pre- and post-execution commands run under cmd.exe /c
- The standard input, standard output, and standard error are set to NULL
- The PATH is determined by the setup of the LSF Service

Note:

For post-execution commands that execute on a Windows Server 2003, x64 Edition platform, users must have read and execute privileges for cmd.exe.

Default

Not defined. No post-execution commands are associated with the application profile.

PRE_EXEC

Syntax

PRE_EXEC=*command*

Description

Enables pre-execution processing at the application level. The PRE_EXEC command runs on the execution host before the job starts. If the PRE_EXEC command exits with a non-zero exit code, LSF requeues the job to the front of the queue.

Pre-execution commands can be configured at the application, queue, and job levels and run in the following order:

1. The queue-level command
2. The application-level or job-level command. If you specify a command at both the application and job levels, the job-level command overrides the application-level command; the application-level command is ignored.

The PRE_EXEC command uses the same environment variable values as the job, and runs under the user account of the user who submits the job. To run pre-execution commands under a different user account (such as root for privileged operations), configure the parameter LSB_PRE_POST_EXEC_USER in lsf.sudoers.

The command path can contain up to 4094 characters for UNIX and Linux, or up to 255 characters for Windows, including the directory, file name, and expanded values for %J (*job_ID*) and %I (*index_ID*).

For UNIX:

- The pre- and post-execution commands run in the `/tmp` directory under `/bin/sh -c`, which allows the use of shell features in the commands. The following example shows valid configuration lines:

```
PRE_EXEC= /usr/share/lsf/misc/testq_pre >> /tmp/pre.out
```

```
POST_EXEC= /usr/share/lsf/misc/testq_post | grep -v "Hey!"
```

- LSF sets the PATH environment variable to

```
PATH='/bin /usr/bin /sbin /usr/sbin'
```

- The stdin, stdout, and stderr are set to `/dev/null`

For Windows:

- The pre- and post-execution commands run under `cmd.exe /c`
- The standard input, standard output, and standard error are set to NULL
- The PATH is determined by the setup of the LSF Service

Note:

For pre-execution commands that execute on a Windows Server 2003, x64 Edition platform, users must have read and execute privileges for `cmd.exe`.

Default

Not defined. No pre-execution commands are associated with the application profile.

PROCESSLIMIT

Syntax

PROCESSLIMIT=*integer*

Description

Limits the number of concurrent processes that can be part of a job.

By default. jobs submitted to the application profile without a job-level process limit are killed when the process limit is reached. Application-level limits override any default limit specified in the queue.

SIGINT, SIGTERM, and SIGKILL are sent to the job in sequence when the limit is reached.

Default

Unlimited

PROCLIMIT

Syntax

PROCLIMIT=[*minimum_limit*] [*default_limit*]] *maximum_limit*

Description

Maximum number of slots that can be allocated to a job. For parallel jobs, the maximum number of processors that can be allocated to the job.

Optionally specifies the minimum and default number of job slots. All limits must be positive integers greater than or equal to 1 that satisfy the following relationship:

1 <= *minimum* <= *default* <= *maximum*

Job-level processor limits (bsub -n) override application-level PROCLIMIT, which overrides queue-level PROCLIMIT. Job-level limits must fall within the maximum and minimum limits of the application profile and the queue.

You can specify up to three limits in the PROCLIMIT parameter:

* One limit—Is the maximum processor limit. The minimum and default limits are set to 1.
* Two limits—The first is the minimum processor limit, and the second one is the maximum. The default is set equal to the minimum. The minimum must be less than or equal to the maximum.
* Three limits—The first is the minimum processor limit, the second is the default processor limit, and the third is the maximum. The minimum must be less than the default and the maximum.

Jobs that request fewer slots than the minimum PROCLIMIT or more slots than the maximum PROCLIMIT cannot use the application profile and are rejected. If the job requests minimum and maximum job slots, the maximum slots requested cannot be less than the minimum PROCLIMIT, and the minimum slots requested cannot be more than the maximum PROCLIMIT.

Default

Unlimited, the default number of slots is 1

REMOTE_MAX_PREEXEC_RETRY

Syntax

REMOTE_MAX_PREEXEC_RETRY=*integer*

Description

The maximum number of times to attempt the pre-execution command of a job on the remote cluster.

Valid values

0 < MAX_PREEXEC_RETRY < INFINIT_INT

INFINIT_INT is defined in lsf.h.

Default

5

REQUEUE_EXIT_VALUES

Syntax

REQUEUE_EXIT_VALUES=[*exit_code ...*] [**EXCLUDE(***exit_code ...***)**]

Description

Enables automatic job requeue and sets the LSB_EXIT_REQUEUE environment variable. Use spaces to separate multiple exit code values. Application-level exit values override queue-level values. Job-level exit values (bsub -Q) override application-level and queue-level values.

exit_code has the following form:

```
"[all] [~number ...] | [number ...]"
```

The reserved keyword all specifies all exit codes. Exit codes are typically between 0 and 255. Use a tilde (~) to exclude specified exit codes from the list.

Jobs running the same applications generally shared the same exit values under the same conditions. Setting REQUEUE_EXIT_VALUES in an application profile instead of in the queue allows different applications with different exit values to share the same queue.

Jobs are requeued to the head of the queue. The output from the failed run is not saved, and the user is not notified by LSF.

Define an exit code as EXCLUDE(*exit_code*) to enable exclusive job requeue. Exclusive job requeue does not work for parallel jobs.

If mbatchd is restarted, it does not remember the previous hosts from which the job exited with an exclusive requeue exit code. In this situation, it is possible for a job to be dispatched to hosts on which the job has previously exited with an exclusive exit code.

Example

REQUEUE_EXIT_VALUES=30 EXCLUDE(20)

means that jobs with exit code 30 are requeued, jobs with exit code 20 are requeued exclusively, and jobs with any other exit code are not requeued.

Default

Not defined, Jobs in the application profile are not requeued.

RERUNNABLE

Syntax

RERUNNABLE=yes | no

Description

If yes, enables automatic job rerun (restart) for any job associated with the application profile.

Rerun is disabled when RERUNNABLE is set to no. The yes and no arguments are not case-sensitive.

Members of a chunk job can be rerunnable. If the execution host becomes unavailable, rerunnable chunk job members are removed from the job chunk and dispatched to a different execution host.

Job level rerun (bsub -r) overrides the RERUNNABLE value specified in the application profile, which overrides the queue specification. bmod -rn to make rerunnable jobs non-rerunnable overrides both the application profile and the queue.

Default

Not defined.

RES_REQ

Syntax

RES_REQ=_res_req_

Description

Resource requirements used to determine eligible hosts. Specify a resource requirement string as usual. The resource requirement string lets you specify conditions in a more flexible manner than using the load thresholds.

Resource requirement strings can be simple (applying to the entire job) or compound (applying to the specified number of slots). When a compound resource requirement is set at the application-level, it will be ignored if any job-level resource requirements (simple or compound) are defined.

In the event no job-level resource requirements are set, the compound application-level requirements interact with queue resource requirement strings in the following ways:

In the event no job-level resource requirements are set, the compound application-level requirements interact with queue-level resource requirement strings in the following ways:

- If no queue-level resource requirement is defined or a compound queue-level resource requirement is defined, the compound application-level requirement is used.
- If a simple queue-level requirement is defined, the application-level and queue-level requirements combine as follows:

section	compound application and simple queue behavior
select	both levels satisfied; queue requirement applies to all compound terms
same	queue level ignored
order span	application-level section overwrites queue-level section (if a given level is present); queue requirement (if used) applies to all compound terms
rusage	- both levels merge - queue requirement if a job-based resource is applied to the first compound term, otherwise applies to all compound terms - if conflicts occur the application-level section overwrites the queue-level section. For example: if the application-level requirement is num1*{rusage[R1]} + num2*{rusage[R2]} and the queue-level requirement is rusage[RQ] where RQ is a job resource, the merged requirement is num1*{rusage[merge(R1, RQ)]} + num2*{rusage[R2]}

The following resource requirement sections are supported:

- select
- rusage
- order
- span

- same
- cu

Compound resource requirements do not support the cu section, multiple -R options, or the || operator within the rusage section.

For internal load indices and duration, jobs are rejected if they specify resource reservation requirements at the job or application level that exceed the requirements specified in the queue.

If RES_REQ is defined at the queue level and there are no load thresholds defined, the pending reasons for each individual load index are not be displayed by bjobs.

By default, memory (mem) and swap (swp) limits in select[] and rusage[] sections are specified in MB. Use LSF_UNIT_FOR_LIMITS in lsf.conf to specify a larger unit for the these limits (GB, TB, PB, or EB).

When LSF_STRICT_RESREQ=Y is configured in lsf.conf, resource requirement strings in select sections must conform to a more strict syntax. The strict resource requirement syntax only applies to the select section. It does not apply to the other resource requirement sections (order, rusage, same, span, or cu). When LSF_STRICT_RESREQ=Y in lsf.conf, LSF rejects resource requirement strings where an rusage section contains a non-consumable resource.

select section

For simple resource requirements, the select section defined at the application, queue, and job level must all be satisfied.

rusage section

The rusage section can specify additional requests. To do this, use the OR (||) operator to separate additional rusage strings. The job-level rusage section takes precedence. Compound resource requirements do not support use of the || operator within the component rusage simple resource requirements.

When both job-level and application-level rusage sections are defined using simple resource requirement strings, the rusage section defined for the job overrides the rusage section defined in the application profile. The rusage definitions are merged, with the job-level rusage taking precedence. Any queue-level requirements are then merged with that result.

For example:

Application-level RES_REQ:	`RES_REQ=rusage[mem=200:lic=1] ...`
	For the job submission:
	`bsub -R'rusage[mem=100]' ...`
	the resulting requirement for the job is
	`rusage[mem=100:lic=1]`
	where mem=100 specified by the job overrides mem=200 specified by the application profile. However, lic=1 from application profile is kept, since job does not specify it.
Application-level RES_REQ with decay and	`RES_REQ=rusage[mem=200:duration=20:decay=1] ...`
	For a job submission with no decay or duration:
	`bsub -R'rusage[mem=100]' ...`

duration defined:	the resulting requirement for the job is:
	`rusage[mem=100: duration=20: decay=1]`

Application-level duration and decay are merged with the job-level specification, and `mem=100` for the job overrides `mem=200` specified by the application profile. However, `duration=20` and `decay=1` from application profile are kept, since job does not specify them.

order section

For simple resource requirements the `order` section defined at the job-level overrides any application-level order section. An application-level order section overrides queue-level specification. The `order` section defined at the application level is ignored if any resource requirements are specified at the job level. If the no resource requirements include an `order` section, the default order `r15s: pg` is used.

span section

For simple resource requirements the `span` section defined at the job-level overrides an application-level span section, which overrides a queue-level span section.

Note:

Define span[hosts=-1] in the application profile or in bsub -R resource requirement string to disable the span section setting in the queue.

same section

For simple resource requirements all `same` sections defined at the job-level, application-level, and queue-level are combined before the job is dispatched.

Note:

Define span[hosts=-1] in the application profile or in `bsub -R` resource requirement string to disable the span section setting in the queue.

cu section

For simple resource requirements the job-level cu section overwrites the application-level, and the application-level cu section overwrites the queue-level.

Default

select[type==local] order[r15s:pg]

If this parameter is defined and a host model or Boolean resource is specified, the default type is any.

RESIZABLE_JOBS

Syntax

RESIZABLE_JOBS = [Y|N|auto]

Description

N|n: The resizable job feature is disabled in the application profile. Under this setting, all jobs attached to this application profile are not resizable. All bresize and bsub -ar commands will be rejected with a proper error message.

Y|y: Resize is enabled in the application profile and all jobs belonging to the application are resizable by default. Under this setting, users can run bresize commands to cancel pending resource allocation requests for the job or release resources from an existing job allocation, or use bsub to submit an autoresizable job.

auto: All jobs belonging to the application will be autoresizable.

Resizable jobs must be submitted with an application profile that defines RESIZABLE_JOBS as either auto or Y. If application defines RESIZABLE_JOBS=auto, but administrator changes it to N and reconfigures LSF, jobs without job-level auto resizable attribute become not autoresizable. For running jobs that are in the middle of notification stage, LSF lets current notification complete and stops scheduling. Changing RESIZABLE_JOBS configuration does not affect jobs with job-level autoresizable attribute. (This behavior is same as exclusive job, bsub -x and EXCLUSIVE parameter in queue level.)

Auto-resizable jobs cannot be submitted with compute unit resource requirements. In the event a bswitch call or queue reconfiguration results in an auto-resizable job running in a queue with compute unit resource requirements, the job will no longer be auto-resizable.

Resizable jobs cannot have compound resource requirements.

Default

If the parameter is undefined, the default value is N.

RESIZE_NOTIFY_CMD

Syntax

RESIZE_NOTIFY_CMD = *notification_command*

Description

Defines an executable command to be invoked on the first execution host of a job when a resize event occurs. The maximum length of notification command is 4 KB.

Default

Not defined. No resize notification command is invoked.

RESUME_CONTROL

Syntax

RESUME_CONTROL=*signal* | *command*

Remember:

Unlike the JOB_CONTROLS parameter in lsb.queues, the RESUME_CONTROL parameter does not require square brackets ([ ]) around the action.

- *signal* is a UNIX signal name. The specified signal is sent to the job. The same set of signals is not supported on all UNIX systems. To display a list of the symbolic names of the signals (without the SIG prefix) supported on your system, use the `kill -l` command.
- *command* specifies a `/bin/sh` command line to be invoked. Do not quote the command line inside an action definition. Do not specify a signal followed by an action that triggers the same signal. For example, do not specify RESUME_CONTROL=bresume. This causes a deadlock between the signal and the action.

Description

Changes the behavior of the RESUME action in LSF.

- The contents of the configuration line for the action are run with `/bin/sh -c` so you can use shell features in the command.
- The standard input, output, and error of the command are redirected to the NULL device, so you cannot tell directly whether the command runs correctly. The default null device on UNIX is `/dev/null`.
- The command is run as the user of the job.
- All environment variables set for the job are also set for the command action. The following additional environment variables are set:

  - LSB_JOBPGIDS — a list of current process group IDs of the job
  - LSB_JOBPIDS — a list of current process IDs of the job

The command path can contain up to 4094 characters for UNIX and Linux, or up to 255 characters for Windows, including the directory, file name, and expanded values for %J (*job_ID*) and %I (*index_ID*).

Default

- On UNIX, by default, RESUME sends SIGCONT.
- On Windows, actions equivalent to the UNIX signals have been implemented to do the default job control actions. Job control messages replace the SIGINT and SIGTERM signals, but only customized applications are able to process them.

RTASK_GONE_ACTION

Syntax

RTASK_GONE_ACTION="[**KILLJOB_TASKDONE** | **KILLJOB_TASKEXIT**]
[**IGNORE_TASKCRASH**]**"**

Description

Defines the actions LSF should take if it detects that a remote task of a parallel or distributed job is gone.

This parameter only applies to the `blaunch` distributed application framework.

IGNORE_TASKCRASH

A remote task crashes. LSF does nothing. The job continues to launch the next task.

KILLJOB_TASKDONE

A remote task exits with zero value. LSF terminates all tasks in the job.

KILLJOB_TASKEXIT

A remote task exits with non-zero value. LSF terminates all tasks in the job.

Environment variable

When defined in an application profile, the LSB_DJOB_RTASK_GONE_ACTION variable is set when running bsub -app for the specified application.

You can also use the environment variable LSB_DJOB_RTASK_GONE_ACTION to override the value set in the application profile.

Example

```
RTASK_GONE_ACTION="IGNORE_TASKCRASH KILLJOB_TASKEXIT"
```

Default

Not defined. LSF does nothing.

RUNLIMIT
Syntax

RUNLIMIT=[*hour:*]*minute*[*/host_name* | */host_model*]

Description

The default run limit. The name of a host or host model specifies the runtime normalization host to use.

By default, jobs that are in the RUN state for longer than the specified run limit are killed by LSF. You can optionally provide your own termination job action to override this default.

Jobs submitted with a job-level run limit (bsub -W) that is less than the run limit are killed when their job-level run limit is reached. Jobs submitted with a run limit greater than the maximum run limit are rejected. Application-level limits override any default limit specified in the queue.

Note:

If you want to provide an estimated run time for scheduling purposes without killing jobs that exceed the estimate, define the RUNTIME parameter in the application profile, or submit the job with -We instead of a run limit.

The run limit is in the form of [*hour:*]*minute*. The minutes can be specified as a number greater than 59. For example, three and a half hours can either be specified as 3:30, or 210.

The run limit you specify is the normalized run time. This is done so that the job does approximately the same amount of processing, even if it is sent to host with a faster or slower CPU. Whenever a normalized run time is given, the actual time on the execution host is the specified time multiplied by the CPU factor of the normalization host then divided by the CPU factor of the execution host.

If ABS_RUNLIMIT=Y is defined in lsb.params or in the application profile, the runtime limit is not normalized by the host CPU factor. Absolute wall-clock run time is used for all jobs submitted to an application profile with a run limit configured.

Optionally, you can supply a host name or a host model name defined in LSF. You must insert '/' between the run limit and the host name or model name. (See l si nf o(1) to get host model information.)

If no host or host model is given, LSF uses the default runtime normalization host defined at the queue level (DEFAULT_HOST_SPEC in l sb. queues) if it has been configured; otherwise, LSF uses the default CPU time normalization host defined at the cluster level (DEFAULT_HOST_SPEC in l sb. par ams) if it has been configured; otherwise, the host with the largest CPU factor (the fastest host in the cluster).

For MultiCluster jobs, if no other CPU time normalization host is defined and information about the submission host is not available, LSF uses the host with the largest CPU factor (the fastest host in the cluster).

Jobs submitted to a chunk job queue are not chunked if RUNLIMIT is greater than 30 minutes.

Default

Unlimited

RUNTIME

Syntax

RUNTIME=[*hour*:]*minute*[*/host_name* | */host_model*]

Description

The RUNTIME parameter specifies an estimated run time for jobs associated with an application. LSF uses the RUNTIME value for scheduling purposes only, and does not kill jobs that exceed this value unless the jobs also exceed a defined RUNLIMIT. The format of runtime estimate is same as the RUNLIMIT parameter.

The job-level runtime estimate specified by bsub - We overrides the RUNTIME setting in an application profile.

The following LSF features use the RUNTIME value to schedule jobs:

- Job chunking
- Advanced reservation
- SLA
- Slot reservation
- Backfill

Default

Not defined

STACKLIMIT

Syntax

STACKLIMIT=*integer*

Description

The per-process (soft) stack segment size limit for all of the processes belonging to a job from this queue (see getrlimit(2)). Application-level limits override any default limit specified in the queue, but must be less than the hard limit of the submission queue.

By default, the limit is specified in KB. Use LSF_UNIT_FOR_LIMITS in lsf.conf to specify a larger unit for the the limit (MB, GB, TB, PB, or EB).

Default

Unlimited

SUCCESS_EXIT_VALUES

Syntax

SUCCESS_EXIT_VALUES=[*exit_code* …]

Description

Specifies exit values used by LSF to determine if job was done successfully. Use spaces to separate multiple exit codes. Job-level success exit values specified with the LSB_SUCCESS_EXIT_VALUES environment variable override the configration in application profile.

Use SUCCESS_EXIT_VALUES for applications that successfully exit with non-zero values so that LSF does not interpret non-zero exit codes as job failure.

exit_code should be the value between 0 and 255. Use spaces to separate exit code values.

Default

Not defined, Jobs do not specify a success exit value.

SUSPEND_CONTROL

Syntax

SUSPEND_CONTROL=signal | command | **CHKPNT**

Remember:

Unlike the JOB_CONTROLS parameter in lsb.queues, the SUSPEND_CONTROL parameter does not require square brackets ([ ]) around the action.

- *signal* is a UNIX signal name (for example, SIGTSTP). The specified signal is sent to the job. The same set of signals is not supported on all UNIX systems. To display a list of the symbolic names of the signals (without the SIG prefix) supported on your system, use the kill -l command.
- *command* specifies a /bin/sh command line to be invoked.

  - Do not quote the command line inside an action definition.
  - Do not specify a signal followed by an action that triggers the same signal. For example, do not specify SUSPEND_CONTROL=bstop. This causes a deadlock between the signal and the action.

- CHKPNT is a special action, which causes the system to checkpoint the job. The job is checkpointed and then stopped by sending the SIGSTOP signal to the job automatically.

Description

Changes the behavior of the SUSPEND action in LSF.

- The contents of the configuration line for the action are run with /bin/sh -c so you can use shell features in the command.
- The standard input, output, and error of the command are redirected to the NULL device, so you cannot tell directly whether the command runs correctly. The default null device on UNIX is /dev/null.
- The command is run as the user of the job.
- All environment variables set for the job are also set for the command action. The following additional environment variables are set:

  - LSB_JOBPGIDS — a list of current process group IDs of the job
  - LSB_JOBPIDS —a list of current process IDs of the job
  - LSB_SUSP_REASONS — an integer representing a bitmap of suspending reasons as defined in lsbatch.h The suspending reason can allow the command to take different actions based on the reason for suspending the job.
  - LSB_SUSP_SUBREASONS — an integer representing the load index that caused the job to be suspended

When the suspending reason SUSP_LOAD_REASON (suspended by load) is set in LSB_SUSP_REASONS, LSB_SUSP_SUBREASONS is set to one of the load index values defined in lsf.h.

Use LSB_SUSP_REASONS and LSB_SUSP_SUBREASONS together in your custom job control to determine the exact load threshold that caused a job to be suspended.

- If an additional action is necessary for the SUSPEND command, that action should also send the appropriate signal to the application. Otherwise, a job can continue to run even after being suspended by LSF. For example, SUSPEND_CONTROL=bkill $LSB_JOBPIDS; *command*

The command path can contain up to 4094 characters for UNIX and Linux, or up to 255 characters for Windows, including the directory, file name, and expanded values for %J (*job_ID*) and %I (*index_ID*).

Default

- On UNIX, by default, SUSPEND sends SIGTSTP for parallel or interactive jobs and SIGSTOP for other jobs.
- On Windows, actions equivalent to the UNIX signals have been implemented to do the default job control actions. Job control messages replace the SIGINT and SIGTERM signals, but only customized applications are able to process them.

SWAPLIMIT

Syntax

SWAPLIMIT=*integer*

Description

Limits the amount of total virtual memory limit for the job.

This limit applies to the whole job, no matter how many processes the job may contain. Application-level limits override any default limit specified in the queue.

The action taken when a job exceeds its SWAPLIMIT or PROCESSLIMIT is to send SIGQUIT, SIGINT, SIGTERM, and SIGKILL in sequence. For CPULIMIT, SIGXCPU is sent before SIGINT, SIGTERM, and SIGKILL.

By default, the limit is specified in KB. Use LSF_UNIT_FOR_LIMITS in lsf.conf to specify a larger unit for the the limit (MB, GB, TB, PB, or EB).

Default

Unlimited

TERMINATE_CONTROL

Syntax

TERMINATE_CONTROL=*signal* | *command* | **CHKPNT**

Remember:

Unlike the JOB_CONTROLS parameter in lsb.queues, the TERMINATE_CONTROL parameter does not require square brackets ([ ]) around the action.

- *signal* is a UNIX signal name (for example, SIGTERM). The specified signal is sent to the job. The same set of signals is not supported on all UNIX systems. To display a list of the symbolic names of the signals (without the SIG prefix) supported on your system, use the kill -l command.
- *command* specifies a /bin/sh command line to be invoked.

  - Do not quote the command line inside an action definition.
  - Do not specify a signal followed by an action that triggers the same signal. For example, do not specify TERMINATE_CONTROL=bkill. This causes a deadlock between the signal and the action.
- CHKPNT is a special action, which causes the system to checkpoint the job. The job is checkpointed and killed automatically.

Description

Changes the behavior of the TERMINATE action in LSF.

- The contents of the configuration line for the action are run with /bin/sh -c so you can use shell features in the command.
- The standard input, output, and error of the command are redirected to the NULL device, so you cannot tell directly whether the command runs correctly. The default null device on UNIX is /dev/null.
- The command is run as the user of the job.
- All environment variables set for the job are also set for the command action. The following additional environment variables are set:

- LSB_JOBPGIDS — a list of current process group IDs of the job
- LSB_JOBPIDS —a list of current process IDs of the job

The command path can contain up to 4094 characters for UNIX and Linux, or up to 255 characters for Windows, including the directory, file name, and expanded values for %J (*job_ID*) and %I (*index_ID*).

Default

- On UNIX, by default, TERMINATE sends SIGINT, SIGTERM and SIGKILL in that order.
- On Windows, actions equivalent to the UNIX signals have been implemented to do the default job control actions. Job control messages replace the SIGINT and SIGTERM signals, but only customized applications are able to process them. Termination is implemented by the `TerminateProcess()` system call.

THREADLIMIT

Syntax

THREADLIMIT=*integer*

Description

Limits the number of concurrent threads that can be part of a job. Exceeding the limit causes the job to terminate. The system sends the following signals in sequence to all processes belongs to the job: SIGINT, SIGTERM, and SIGKILL.

By default, jobs submitted to the queue without a job-level thread limit are killed when the thread limit is reached. Application-level limits override any default limit specified in the queue.

The limit must be a positive integer.

Default

Unlimited

USE_PAM_CREDS

Syntax

USE_PAM_CREDS=y | n

Description

If USE_PAM_CREDS=y, applies PAM limits to an application when its job is dispatched to a Linux host using PAM. PAM limits are system resource limits defined in `limits.conf`.

When USE_PAM_CREDS is enabled, PAM limits override others.

If the execution host does not have PAM configured and this parameter is enabled, the job fails.

For parallel jobs, only takes effect on the first execution host.

Overrides MEMLIMIT_TYPE=Process.

Overridden (for CPU limit only) by LSB_JOB_CPULIMIT=y.

Overridden (for memory limits only) by LSB_JOB_MEMLIMIT=y.

Default

n

lsb.events

The LSF batch event log file lsb.events is used to display LSF batch event history and for mbatchd failure recovery.

Whenever a host, job, or queue changes status, a record is appended to the event log file. The file is located in LSB_SHAREDIR/*cluster_name*/logdir, where LSB_SHAREDIR must be defined in lsf.conf(5) and *cluster_name* is the name of the LSF cluster, as returned by lsid. See mbatchd(8) for the description of LSB_SHAREDIR.

The bhist command searches the most current lsb.events file for its output.

lsb.events structure

The event log file is an ASCII file with one record per line. For the lsb.events file, the first line has the format # *history_seek_position*>, which indicates the file position of the first history event after log switch. For the lsb.events.# file, the first line has the format # *timestamp_most_recent_event*, which gives the timestamp of the most recent event in the file.

Limiting the size of lsb.events

Use MAX_JOB_NUM in lsb.params to set the maximum number of finished jobs whose events are to be stored in the lsb.events log file.

Once the limit is reached, mbatchd starts a new event log file. The old event log file is saved as lsb.events.*n*, with subsequent sequence number suffixes incremented by 1 each time a new log file is started. Event logging continues in the new lsb.events file.

Records and fields

The fields of a record are separated by blanks. The first string of an event record indicates its type. The following types of events are recorded:

- JOB_NEW
- JOB_FORWARD
- JOB_ACCEPT
- JOB_START
- JOB_START_ACCEPT
- JOB_STATUS
- JOB_SWITCH
- JOB_MOVE
- QUEUE_CTRL
- HOST_CTRL
- MBD_START
- MBD_DIE
- UNFULFILL
- LOAD_INDEX
- JOB_SIGACT
- MIG
- JOB_MODIFY2
- JOB_SIGNAL
- JOB_EXECUTE

- JOB_REQUEUE
- JOB_CLEAN
- JOB_EXCEPTION
- JOB_EXT_MSG
- JOB_ATTA_DATA
- JOB_CHUNK
- SBD_UNREPORTED_STATUS
- PRE_EXEC_START
- JOB_FORCE
- GRP_ADD
- GRP_MOD
- LOG_SWITCH
- JOB_RESIZE_NOTIFY_START
- JOB_RESIZE_NOTIFY_ACCEPT
- JOB_RESIZE_NOTIFY_DONE
- JOB_RESIZE_RELEASE
- JOB_RESIZE_CANCEL

JOB_NEW

A new job has been submitted. The fields in order of occurrence are:

Version number (%s)

The version number

Event time (%d)

The time of the event

jobId (%d)

Job ID

userId (%d)

UNIX user ID of the submitter

options (%d)

Bit flags for job processing

numProcessors (%d)

Number of processors requested for execution

submitTime (%d)

Job submission time

beginTime (%d)

Start time – the job should be started on or after this time

termTime (%d)

Termination deadline – the job should be terminated by this time (%d)

sigValue (%d)

Signal value

chkpntPeriod (%d)

Checkpointing period

restartPid (%d)

Restart process ID

userName (%s)

User name

rLimits

Soft CPU time limit (%d), see getrlimit(2)

rLimits

Soft file size limit (%d), see getrlimit(2)

rLimits

Soft data segment size limit (%d), see getrlimit(2)

rLimits

Soft stack segment size limit (%d), see getrlimit(2)

rLimits

Soft core file size limit (%d), see getrlimit(2)

rLimits

Soft memory size limit (%d), see getrlimit(2)

rLimits

Reserved (%d)

rLimits

Reserved (%d)

rLimits

Reserved (%d)

rLimits

Soft run time limit (%d), see getrlimit(2)

rLimits

Reserved (%d)

hostSpec (%s)

Model or host name for normalizing CPU time and run time

hostFactor (%f)

CPU factor of the above host

umask (%d)

> File creation mask for this job

queue (%s)

> Name of job queue to which the job was submitted

resReq (%s)

> Resource requirements

fromHost (%s)

> Submission host name

cwd (%s)

> Current working directory (up to 4094 characters for UNIX or 255 characters for Windows)

chkpntDir (%s)

> Checkpoint directory

inFile (%s)

> Input file name (up to 4094 characters for UNIX or 255 characters for Windows)

outFile (%s)

> Output file name (up to 4094 characters for UNIX or 255 characters for Windows)

errFile (%s)

> Error output file name (up to 4094 characters for UNIX or 255 characters for Windows)

subHomeDir (%s)

> Submitter's home directory

jobFile (%s)

> Job file name

numAskedHosts (%d)

> Number of candidate host names

askedHosts (%s)

> List of names of candidate hosts for job dispatching

dependCond (%s)

> Job dependency condition

preExecCmd (%s)

> Job pre-execution command

timeEvent (%d)

> Time Event, for job dependency condition; specifies when time event ended

jobName (%s)

> Job name (up to 4094 characters for UNIX or 255 characters for Windows)

command (%s)

> Job command (up to 4094 characters for UNIX or 255 characters for Windows)

nxf (%d)

> Number of files to transfer (%d)

xf (%s)

> List of file transfer specifications

mailUser (%s)

> Mail user name

projectName (%s)

> Project name

niosPort (%d)

> Callback port if batch interactive job

maxNumProcessors (%d)

> Maximum number of processors

schedHostType (%s)

> Execution host type

loginShell (%s)

> Login shell

userGroup (%s)

> User group

exceptList (%s)

> Exception handlers for the job

options2 (%d)

> Bit flags for job processing

idx (%d)

> Job array index

inFileSpool (%s)

> Spool input file (up to 4094 characters for UNIX or 255 characters for Windows)

commandSpool (%s)

> Spool command file (up to 4094 characters for UNIX or 255 characters for Windows)

jobSpoolDir (%s)

Job spool directory (up to 4094 characters for UNIX or 255 characters for Windows)

userPriority (%d)

User priority

rsvId %s

Advance reservation ID; for example, "user2#0"

jobGroup (%s)

The job group under which the job runs

extsched (%s)

External scheduling options

warningAction (%s)

Job warning action

warningTimePeriod (%d)

Job warning time period in seconds

sla (%s)

SLA service class name under which the job runs

SLArunLimit (%d)

Absolute run time limit of the job for SLA service classes

licenseProject (%s)

LSF License Scheduler project name

options3 (%d)

Bit flags for job processing

app (%s)

Application profile name

postExecCmd (%s)

Post-execution command to run on the execution host after the job finishes

runtimeEstimation (%d)

Estimated run time for the job

requeueEValues (%s)

Job exit values for automatic job requeue

resizeNotifyCmd (%s)

Resize notification command to run on the first execution host to inform job of a resize event.

JOB_FORWARD

A job has been forwarded to a remote cluster (Platform MultiCluster only).

If LSF_HPC_EXTENSIONS="SHORT_EVENTFILE" is specified in lsf.conf, older daemons and commands (pre-LSF Version 6.0) cannot recognize the lsb.events file format.

The fields in order of occurrence are:

Version number (%s)

The version number

Event time (%d)

The time of the event

jobId (%d)

Job ID

numReserHosts (%d)

Number of reserved hosts in the remote cluster

If LSF_HPC_EXTENSIONS="SHORT_EVENTFILE" is specified in lsf.conf, the value of this field is the number of .hosts listed in the reserHosts field.

cluster (%s)

Remote cluster name

reserHosts (%s)

List of names of the reserved hosts in the remote cluster

If LSF_HPC_EXTENSIONS="SHORT_EVENTFILE" is specified in lsf.conf, the value of this field is logged in a shortened format.

idx (%d)

Job array index

JOB_ACCEPT

A job from a remote cluster has been accepted by this cluster. The fields in order of occurrence are:

Version number (%s)

The version number

Event time (%d)

The time of the event

jobId (%d)

Job ID at the accepting cluster

remoteJid (%d)

Job ID at the submission cluster

cluster (%s)

> Job submission cluster name

idx (%d)

> Job array index

JOB_START

> A job has been dispatched.
>
> If LSF_HPC_EXTENSIONS="SHORT_EVENTFILE" is specified in lsf.conf, older daemons and commands (pre-LSF Version 6.0) cannot recognize the lsb.events file format.
>
> The fields in order of occurrence are:

Version number (%s)

> The version number

Event time (%d)

> The time of the event

jobId (%d)

> Job ID

jStatus (%d)

> Job status, (4, indicating the RUN status of the job)

jobPid (%d)

> Job process ID

jobPGid (%d)

> Job process group ID

hostFactor (%f)

> CPU factor of the first execution host

numExHosts (%d)

> Number of processors used for execution
>
> If LSF_HPC_EXTENSIONS="SHORT_EVENTFILE" is specified in lsf.conf, the value of this field is the number of .hosts listed in the execHosts field.

execHosts (%s)

> List of execution host names
>
> If LSF_HPC_EXTENSIONS="SHORT_EVENTFILE" is specified in lsf.conf, the value of this field is logged in a shortened format.

queuePreCmd (%s)

> Pre-execution command

queuePostCmd (%s)

Post-execution command

jFlags (%d)

Job processing flags

userGroup (%s)

User group name

idx (%d)

Job array index

additionalInfo (%s)

Placement information of HPC jobs

jFlags2 (%d)

JOB_START_ACCEPT

A job has started on the execution host(s). The fields in order of occurrence are:

Version number (%s)

The version number

Event time (%d)

The time of the event

jobId (%d)

Job ID

jobPid (%d)

Job process ID

jobPGid (%d)

Job process group ID

idx (%d)

Job array index

JOB_STATUS

The status of a job changed after dispatch. The fields in order of occurrence are:

Version number (%s)

The version number

Event time (%d)

The time of the event

jobId (%d)

Job ID

jStatus (%d)

New status, see <lsf/lsbatch.h>

reason (%d)

Pending or suspended reason code, see <lsf/lsbatch.h>

subreasons (%d)

Pending or suspended subreason code, see <lsf/lsbatch.h>

cpuTime (%f)

CPU time consumed so far

endTime (%d)

Job completion time

ru (%d)

Resource usage flag

lsfRusage (%s)

Resource usage statistics, see <lsf/lsf.h>

exitStatus (%d)

Exit status of the job, see <lsf/lsbatch.h>

idx (%d)

Job array index

exitInfo (%d)

Job termination reason, see <lsf/lsbatch.h>

duration4PreemptBackfill

How long a backfilled job can run; used for preemption backfill jobs

JOB_SWITCH

A job switched from one queue to another (bswitch). The fields in order of occurrence are:

Version number (%s)

The version number

Event time (%d)

The time of the event

userId (%d)

UNIX user ID of the user invoking the command

jobId (%d)

Job ID

queue (%s)

Target queue name

idx (%d)

Job array index

userName (%s)

Name of the job submitter

JOB_MOVE

A job moved toward the top or bottom of its queue (bbot or btop). The fields in order of occurrence are:

Version number (%s)

The version number

Event time (%d)

The time of the event

userId (%d)

UNIX user ID of the user invoking the command

jobId (%d)

Job ID

position (%d)

Position number

base (%d)

Operation code, (TO_TOP or TO_BOTTOM), see <lsf/lsbatch.h>

idx (%d)

Job array index

userName (%s)

Name of the job submitter

QUEUE_CTRL

A job queue has been altered. The fields in order of occurrence are:

Version number (%s)

The version number

Event time (%d)

The time of the event

opCode (%d)

Operation code), see <lsf/lsbatch.h>

queue (%s)

Queue name

userId (%d)

UNIX user ID of the user invoking the command

userName (%s)

Name of the user

ctrlComments (%s)

Administrator comment text from the -C option of badmin queue control commands qclose, qopen, qact, and qinact

HOST_CTRL

A batch server host changed status. The fields in order of occurrence are:

Version number (%s)

The version number

Event time (%d)

The time of the event

opCode (%d)

Operation code, see <lsf/lsbatch.h>

host (%s)

Host name

userId (%d)

UNIX user ID of the user invoking the command

userName (%s)

Name of the user

ctrlComments (%s)

Administrator comment text from the -C option of badmin host control commands hclose and hopen

MBD_START

The mbatchd has started. The fields in order of occurrence are:

Version number (%s)

The version number

Event time (%d)

The time of the event

master (%s)

Master host name

cluster (%s)

> cluster name

numHosts (%d)

> Number of hosts in the cluster

numQueues (%d)

> Number of queues in the cluster

MBD_DIE

> The mbatchd died. The fields in order of occurrence are:

Version number (%s)

> The version number

Event time (%d)

> The time of the event

master (%s)

> Master host name

numRemoveJobs (%d)

> Number of finished jobs that have been removed from the system and logged in the
> current event file

exitCode (%d)

> Exit code from mbatchd

ctrlComments (%s)

> Administrator comment text from the -C option of `badmin mbdrestart`

UNFULFILL

> Actions that were not taken because the mbatchd was unable to contact the sbatchd on the
> job execution host. The fields in order of occurrence are:

Version number (%s)

> The version number

Event time (%d)

> The time of the event

jobId (%d)

> Job ID

notSwitched (%d)

> Not switched: the mbatchd has switched the job to a new queue, but the sbatchd has
> not been informed of the switch

sig (%d)

Signal: this signal has not been sent to the job

sig1 (%d)

Checkpoint signal: the job has not been sent this signal to checkpoint itself

sig1Flags (%d)

Checkpoint flags, see `<lsf/lsbatch.h>`

chkPeriod (%d)

New checkpoint period for job

notModified (%s)

If set to true, then parameters for the job cannot be modified.

idx (%d)

Job array index

LOAD_INDEX

mbatchd restarted with these load index names (see `lsf.cluster`(5)). The fields in order of occurrence are:

Version number (%s)

The version number

Event time (%d)

The time of the event

nIdx (%d)

Number of index names

name (%s)

List of index names

JOB_SIGACT

An action on a job has been taken. The fields in order of occurrence are:

Version number (%s)

The version number

Event time (%d)

The time of the event

jobId (%d)

Job ID

period (%d)

Action period

pid (%d)

Process ID of the child sbatchd that initiated the action

jstatus (%d)

Job status

reasons (%d)

Job pending reasons

flags (%d)

Action flags, see `<lsf/lsbatch.h>`

actStatus (%d)

Action status:

1: Action started

2: One action preempted other actions

3: Action succeeded

4: Action Failed

signalSymbol (%s)

Action name, accompanied by actFlags

idx (%d)

Job array index

MIG

A job has been migrated (bmig). The fields in order of occurrence are:

Version number (%s)

The version number

Event time (%d)

The time of the event

jobId (%d)

Job ID

numAskedHosts (%d)

Number of candidate hosts for migration

askedHosts (%s)

List of names of candidate hosts

userId (%d)

UNIX user ID of the user invoking the command

idx (%d)

Job array index

userName (%s)

>   Name of the job submitter

JOB_MODIFY2

>   This is created when the mbatchd modifies a previously submitted job with bmod.

Version number (%s)

>   The version number

Event time (%d)

>   The time of the event

jobIdStr (%s)

>   Job ID

options (%d)

>   Bit flags for job modification options processing

options2 (%d)

>   Bit flags for job modification options processing

delOptions (%d)

>   Delete options for the options field

delOptions2 (%d)

>   Delete options for the options2 field

userId (%d)

>   UNIX user ID of the submitter

userName (%s)

>   User name

submitTime (%d)

>   Job submission time

umask (%d)

>   File creation mask for this job

numProcessors (%d)

>   Number of processors requested for execution. The value 2147483646 means the
>   number of processors is undefined.

beginTime (%d)

>   Start time – the job should be started on or after this time

termTime (%d)

>   Termination deadline – the job should be terminated by this time

sigValue (%d)

> Signal value

restartPid (%d)

> Restart process ID for the original job

jobName (%s)

> Job name (up to 4094 characters for UNIX or 255 characters for Windows)

queue (%s)

> Name of job queue to which the job was submitted

numAskedHosts (%d)

> Number of candidate host names

askedHosts (%s)

> List of names of candidate hosts for job dispatching; blank if the last field value is 0. If there is more than one host name, then each additional host name will be returned in its own field

resReq (%s)

> Resource requirements

rLimits

> Soft CPU time limit (%d), see `getrlimit`(2)

rLimits

> Soft file size limit (%d), see `getrlimit`(2)

rLimits

> Soft data segment size limit (%d), see `getrlimit`2)

rLimits

> Soft stack segment size limit (%d), see `getrlimit`(2)

rLimits

> Soft core file size limit (%d), see `getrlimit`(2)

rLimits

> Soft memory size limit (%d), see `getrlimit`(2)

rLimits

> Reserved (%d)

rLimits

> Reserved (%d)

rLimits

> Reserved (%d)

rLimits

>Soft run time limit (%d), see `getrlimit(2)`

rLimits

>Reserved (%d)

hostSpec (%s)

>Model or host name for normalizing CPU time and run time

dependCond (%s)

>Job dependency condition

timeEvent (%d)

>Time Event, for job dependency condition; specifies when time event ended

subHomeDir (%s)

>Submitter's home directory

inFile (%s)

>Input file name (up to 4094 characters for UNIX or 255 characters for Windows)

outFile (%s)

>Output file name (up to 4094 characters for UNIX or 255 characters for Windows)

errFile (%s)

>Error output file name (up to 4094 characters for UNIX or 255 characters for Windows)

command (%s)

>Job command (up to 4094 characters for UNIX or 255 characters for Windows)

inFileSpool (%s)

>Spool input file (up to 4094 characters for UNIX or 255 characters for Windows)

commandSpool (%s)

>Spool command file (up to 4094 characters for UNIX or 255 characters for Windows)

chkpntPeriod (%d)

>Checkpointing period

chkpntDir (%s)

>Checkpoint directory

nxf (%d)

>Number of files to transfer

xf (%s)

>List of file transfer specifications

jobFile (%s)

> Job file name

fromHost (%s)

> Submission host name

cwd (%s)

> Current working directory (up to 4094 characters for UNIX or 255 characters for Windows)

preExecCmd (%s)

> Job pre-execution command

mailUser (%s)

> Mail user name

projectName (%s)

> Project name

niosPort (%d)

> Callback port if batch interactive job

maxNumProcessors (%d)

> Maximum number of processors. The value 2147483646 means the maximum number of processors is undefined.

loginShell (%s)

> Login shell

schedHostType (%s)

> Execution host type

userGroup (%s)

> User group

exceptList (%s)

> Exception handlers for the job

userPriority (%d)

> User priority

rsvId %s

> Advance reservation ID; for example, "user2#0"

extsched (%s)

> External scheduling options

warningTimePeriod (%d)

> Job warning time period in seconds

warningAction (%s)

> Job warning action

jobGroup (%s)

> The job group to which the job is attached

sla (%s)

> SLA service class name that the job is to be attached to

licenseProject (%s)

> LSF License Scheduler project name

options3 (%d)

> Bit flags for job processing

delOption3 (%d)

> Delete options for the options3 field

app (%s)

> Application profile name

apsString (%s)

> Absolute priority scheduling (APS) value set by administrator

postExecCmd (%s)

> Post-execution command to run on the execution host after the job finishes

runtimeEstimation (%d)

> Estimated run time for the job

requeueEValues (%s)

> Job exit values for automatic job requeue

resizeNotifyCmd (%s)

> Resize notification command to run on the first execution host to inform job of a resize event.

JOB_SIGNAL

> This is created when a job is signaled with bkill or deleted with bdel . The fields are in the order they appended:

Version number (%s)

> The version number

Event time (%d)

> The time of the event

jobId (%d)

Job ID

userId (%d)

UNIX user ID of the user invoking the command

runCount (%d)

Number of runs

signalSymbol (%s)

Signal name

idx (%d)

Job array index

userName (%s)

Name of the job submitter

JOB_EXECUTE

This is created when a job is actually running on an execution host. The fields in order of occurrence are:

Version number (%s)

The version number

Event time (%d)

The time of the event

jobId (%d)

Job ID

execUid (%d)

Mapped UNIX user ID on execution host

jobPGid (%d)

Job process group ID

execCwd (%s)

Current working directory job used on execution host (up to 4094 characters for UNIX or 255 characters for Windows)

execHome (%s)

Home directory job used on execution host

execUsername (%s)

Mapped user name on execution host

jobPid (%d)

Job process ID

idx (%d)

>Job array index

additionalInfo (%s)

>Placement information of HPC jobs

SLAscaledRunLimit (%d)

>Run time limit for the job scaled by the execution host

execRusage

>An internal field used by LSF.

Position

>An internal field used by LSF.

duration4PreemptBackfill

>How long a backfilled job can run; used for preemption backfill jobs

JOB_REQUEUE

>This is created when a job ended and requeued by mbatchd. The fields in order of occurrence are:

Version number (%s)

>The version number

Event time (%d)

>The time of the event

jobId (%d)

>Job ID

idx (%d)

>Job array index

JOB_CLEAN

>This is created when a job is removed from the mbatchd memory. The fields in order of occurrence are:

Version number (%s)

>The version number

Event time (%d)

>The time of the event

jobId (%d)

>Job ID

idx (%d)

Job array index

JOB_EXCEPTION

This is created when an exception condition is detected for a job. The fields in order of occurrence are:

Version number (%s)

The version number

Event time (%d)

The time of the event

jobId (%d)

Job ID

exceptMask (%d)

Exception Id

0x01: missched

0x02: overrun

0x04: underrun

0x08: abend

0x10: cantrun

0x20: hostfail

0x40: startfail

0x100:runtime_est_exceeded

actMask (%d)

Action Id

0x01: kill

0x02: alarm

0x04: rerun

0x08: setexcept

timeEvent (%d)

Time Event, for `missched` exception specifies when time event ended.

exceptInfo (%d)

Except Info, pending reason for `missched` or `cantrun` exception, the exit code of the job for the `abend` exception, otherwise 0.

idx (%d)

Job array index

JOB_EXT_MSG

An external message has been sent to a job. The fields in order of occurrence are:

Version number (%s)

The version number

Event time (%d)

The time of the event

jobId (%d)

Job ID

idx (%d)

Job array index

msgIdx (%d)

Index in the list

userId (%d)

Unique user ID of the user invoking the command

dataSize (%ld)

Size of the data if it has any, otherwise 0

postTime (%ld)

Message sending time

dataStatus (%d)

Status of the attached data

desc (%s)

Text description of the message

userName (%s)

Name of the author of the message

JOB_ATTA_DATA

An update on the data status of a message for a job has been sent. The fields in order of occurrence are:

Version number (%s)

The version number

Event time (%d)

The time of the event

jobId (%d)

Job ID

idx (%d)

>>> Job array index

msgIdx (%d)

>>> Index in the list

dataSize (%ld)

>>> Size of the data if is has any, otherwise 0

dataStatus (%d)

>>> Status of the attached data

fileName (%s)

>>> File name of the attached data

JOB_CHUNK

>>> This is created when a job is inserted into a chunk.

>>> If LSF_HPC_EXTENSIONS="SHORT_EVENTFILE" is specified in lsf.conf, older daemons and commands (pre-LSF Version 6.0) cannot recognize the lsb.events file format.

>>> The fields in order of occurrence are:

Version number (%s)

>>> The version number

Event time (%d)

>>> The time of the event

membSize (%ld)

>>> Size of array membJobId

membJobId (%ld)

>>> Job IDs of jobs in the chunk

numExHosts (%ld)

>>> Number of execution hosts

>>> If LSF_HPC_EXTENSIONS="SHORT_EVENTFILE" is specified in lsf.conf, the value of this field is the number of .hosts listed in the execHosts field.

execHosts (%s)

>>> Execution host name array

>>> If LSF_HPC_EXTENSIONS="SHORT_EVENTFILE" is specified in lsf.conf, the value of this field is logged in a shortened format.

SBD_UNREPORTED_STATUS

>>> This is created when an unreported status change occurs. The fields in order of occurrence are:

Version number (%s)

>   The version number

Event time (%d)

>   The time of the event

jobId (%d)

>   Job ID

actPid (%d)

>   Acting processing ID

jobPid (%d)

>   Job process ID

jobPGid (%d)

>   Job process group ID

newStatus (%d)

>   New status of the job

reason (%d)

>   Pending or suspending reason code, see `<lsf/lsbatch.h>`

suspreason (%d)

>   Pending or suspending subreason code, see `<lsf/lsbatch.h>`

lsfRusage

>   The following fields contain resource usage information for the job (see `getrusage`
>   (2)). If the value of some field is unavailable (due to job exit or the difference among
>   the operating systems), -1 will be logged. Times are measured in seconds, and sizes are
>   measured in KB.

ru_utime (%f)

>   User time used

ru_stime (%f)

>   System time used

ru_maxrss (%f)

>   Maximum shared text size

ru_ixrss (%f)

>   Integral of the shared text size over time (in KB seconds)

ru_ismrss (%f)

>   Integral of the shared memory size over time (valid only on Ultrix)

ru_idrss (%f)

Integral of the unshared data size over time

ru_isrss (%f)

Integral of the unshared stack size over time

ru_minflt (%f)

Number of page reclaims

ru_majflt (%f)

Number of page faults

ru_nswap (%f)

Number of times the process was swapped out

ru_inblock (%f)

Number of block input operations

ru_oublock (%f)

Number of block output operations

ru_ioch (%f)

Number of characters read and written (valid only on HP-UX)

ru_msgsnd (%f)

Number of System V IPC messages sent

ru_msgrcv (%f)

Number of messages received

ru_nsignals (%f)

Number of signals received

ru_nvcsw (%f)

Number of voluntary context switches

ru_nivcsw (%f)

Number of involuntary context switches

ru_exutime (%f)

Exact user time used (valid only on ConvexOS)

exitStatus (%d)

Exit status of the job, see <lsf/lsbatch.h>

execCwd (%s)

Current working directory job used on execution host (up to 4094 characters for UNIX or 255 characters for Windows)

execHome (%s)

Home directory job used on execution host

execUsername (%s)

Mapped user name on execution host

msgId (%d)

ID of the message

actStatus (%d)

Action status

1: Action started

2: One action preempted other actions

3: Action succeeded

4: Action Failed

sigValue (%d)

Signal value

seq (%d)

Sequence status of the job

idx (%d)

Job array index

jRusage

The following fields contain resource usage information for the job. If the value of some field is unavailable (due to job exit or the difference among the operating systems), -1 will be logged. Times are measured in seconds, and sizes are measured in KB.

mem (%d)

Total resident memory usage in KB of all currently running processes in a given process group

swap (%d)

Totaly virtual memory usage in KB of all currently running processes in given process groups

utime (%d)

Cumulative total user time in seconds

stime (%d)

Cumulative total system time in seconds

npids (%d)

Number of currently active process in given process groups. This entry has four sub-fields:

pid (%d)

> Process ID of the child sbatchd that initiated the action

ppid (%d)

> Parent process ID

pgid (%d)

> Process group ID

jobId (%d)

> Process Job ID

npgids (%d)

> Number of currently active process groups

exitInfo (%d)

> Job termination reason, see <lsf/lsbatch.h>

PRE_EXEC_START

> A pre-execution command has been started.
>
> The fields in order of occurrence are:

Version number (%s)

> The version number

Event time (%d)

> The time of the event

jobId (%d)

> Job ID

jStatus (%d)

> Job status, (4, indicating the RUN status of the job)

jobPid (%d)

> Job process ID

jobPGid (%d)

> Job process group ID

hostFactor (%f)

> CPU factor of the first execution host

numExHosts (%d)

> Number of processors used for execution

execHosts (%s)

> List of execution host names

queuePreCmd (%s)

> Pre-execution command

queuePostCmd (%s)

> Post-execution command

jFlags (%d)

> Job processing flags

userGroup (%s)

> User group name

idx (%d)

> Job array index

additionalInfo (%s)

> Placement information of HPC jobs

JOB_FORCE

> A job has been forced to run with `brun`.

Version number (%s)

> The version number

Event time (%d)

> The time of the event

jobId (%d)

> Job ID

userId (%d)

> UNIX user ID of the user invoking the command

idx (%d)

> Job array index

options (%d)

> Bit flags for job processing

numExecHosts (%ld)

> Number of execution hosts
>
> If LSF_HPC_EXTENSIONS="SHORT_EVENTFILE" is specified in `lsf.conf`, the value of this field is the number of .hosts listed in the execHosts field.

execHosts (%s)

> Execution host name array

If LSF_HPC_EXTENSIONS="SHORT_EVENTFILE" is specified in lsf.conf, the value of this field is logged in a shortened format.

userName (%s)

Name of the user

queue (%s)

Name of queue if a remote brun job ran; otherwise, this field is empty

GRP_ADD

This is created when a job group is added. The fields in order of occurrence are:

Version number (%s)

The version number

Event time (%d)

The time of the event

userId (%d)

UNIX user ID of the job group owner

submitTime (%d)

Job submission time

userName (%s)

User name of the job group owner

depCond (%s)

Job dependency condition

timeEvent (%d)

Time Event, for job dependency condition; specifies when time event ended

groupSpec (%s)

Job group name

delOptions (%d)

Delete options for the options field

delOptions2 (%d)

Delete options for the options2 field

sla (%s)

SLA service class name that the job group is to be attached to

maxJLimit (%d)

Job group limit set by bgadd -L

groupType (%d)

Job group creation method:

- 0x01 - job group was created explicitly
- 0x02 - job group was created implicitly

GRP_MOD

This is created when a job group is modified. The fields in order of occurrence are:

Version number (%s)

The version number

Event time (%d)

The time of the event

userId (%d)

UNIX user ID of the job group owner

submitTime (%d)

Job submission time

userName (%s)

User name of the job group owner

depCond (%s)

Job dependency condition

timeEvent (%d)

Time Event, for job dependency condition; specifies when time event ended

groupSpec (%s)

Job group name

delOptions (%d)

Delete options for the options field

delOptions2 (%d)

Delete options for the options2 field

sla (%s)

SLA service class name that the job group is to be attached to

maxJLimit (%d)

Job group limit set by bgmod -L

LOG_SWITCH

This is created when switching the event file lsb.events. The fields in order of occurrence are:

Version number (%s)

The version number

Event time (%d)

The time of the event

jobId (%d)

Job ID

JOB_RESIZE_NOTIFY_START

LSF logs this event when a resize (shrink or grow) request has been sent to the first execution host. The fields in order of occurrence are:

Version number (%s)

The version number.

Event time (%d)

The time of the event.

jobId (%d)

The job ID.

idx (%d)

Job array index.

notifyId (%d)

Identifier or handle for notification.

numResizeHosts (%d)

Number of processors used for execution. If LSF_HPC_EXTENSIONS="SHORT_EVENTFILE" is specified in `lsf.conf`, the value of this field is the number of hosts listed in short format.

resizeHosts (%s)

List of execution host names. If LSF_HPC_EXTENSIONS="SHORT_EVENTFILE" is specified in lsf.conf, the value of this field is logged in a shortened format.

JOB_RESIZE_NOTIFY_ACCEPT

LSF logs this event when a resize request has been accepted from the first execution host of a job. The fields in order of occurrence are:

Version number (%s)

The version number.

Event time (%d)

The time of the event.

jobId (%d)

The job ID.

idx (%d)

>Job array index.

notifyId (%d)

>Identifier or handle for notification.

resizeNotifyCmdPid (%d)

>Resize notification executable process ID. If no resize notification executable is defined, this field will be set to 0.

resizeNotifyCmdPGid (%d)

>Resize notification executable process group ID. If no resize notification executable is defined, this field will be set to 0.

status (%d)

>Status field used to indicate possible errors. 0 Success, 1 failure.

JOB_RESIZE_NOTIFY_DONE

>LSF logs this event when the resize notification command completes. The fields in order of occurrence are:

Version number (%s)

>The version number.

Event time (%d)

>The time of the event.

jobId (%d)

>The job ID.

idx (%d)

>Job array index.

notifyId (%d)

>Identifier or handle for notification.

status (%d)

>Resize notification exit value. (0, success, 1, failure, 2 failure but cancel request.)

JOB_RESIZE_RELEASE

>LSF logs this event when receiving resource release request from client. The fields in order of occurrence are:

Version number (%s)

>The version number.

Event time (%d)

>The time of the event.

jobId (%d)

> The job ID.

idx (%d)

> Job array index.

reqid (%d)

> Request Identifier or handle.

options (%d)

> Release options.

userId (%d)

> UNIX user ID of the user invoking the command.

userName (%s)

> User name of the submitter.

resizeNotifyCmd (%s)

> Resize notification command to run on the first execution host to inform job of a resize event.

numResizeHosts (%d)

> Number of processors used for execution during resize. If LSF_HPC_EXTENSIONS="SHORT_EVENTFILE" is specified in lsf.conf, the value of this field is the number of hosts listed in short format.

resizeHosts (%s)

> List of execution host names during resize. If LSF_HPC_EXTENSIONS="SHORT_EVENTFILE" is specified in lsf.conf, the value of this field is logged in a shortened format.

JOB_RESIZE_CANCEL

> LSF logs this event when receiving cancel request from client. The fields in order of occurrence are:

Version number (%s)

> The version number.

Event time (%d)

> The time of the event.

jobId (%d)

> The job ID.

idx (%d)

> Job array index.

userId (%d)

UNIX user ID of the user invoking the command.

userName (%s)

User name of the submitter.

lsb.hosts

The lsb.hosts file contains host-related configuration information for the server hosts in the cluster. It is also used to define host groups, host partitions, and compute units.

This file is optional. All sections are optional.

By default, this file is installed in LSB_CONFDIR/*cluster_name*/configdir.

Changing lsb.hosts configuration

After making any changes to lsb.hosts, run badmin reconfig to reconfigure mbatchd.

Host section

Description

Optional. Defines the hosts, host types, and host models used as server hosts, and contains per-host configuration information. If this section is not configured, LSF uses all hosts in the cluster (the hosts listed in lsf.cluster.*cluster_name*) as server hosts.

Each host, host model or host type can be configured to:

- Limit the maximum number of jobs run in total
- Limit the maximum number of jobs run by each user
- Run jobs only under specific load conditions
- Run jobs only under specific time windows

The entries in a line for a host override the entries in a line for its model or type.

When you modify the cluster by adding or removing hosts, no changes are made to lsb.hosts. This does not affect the default configuration, but if hosts, host models, or host types are specified in this file, you should check this file whenever you make changes to the cluster and update it manually if necessary.

Host section structure

The first line consists of keywords identifying the load indices that you wish to configure on a per-host basis. The keyword HOST_NAME must be used; the others are optional. Load indices not listed on the keyword line do not affect scheduling decisions.

Each subsequent line describes the configuration information for one host, host model or host type. Each line must contain one entry for each keyword. Use empty parentheses ( ) or a dash (-) to specify the default value for an entry.

HOST_NAME

Required. Specify the name, model, or type of a host, or the keyword default.

host name

The name of a host defined in lsf.cluster.*cluster_name*.

host model

A host model defined in lsf.shared.

host type

A host type defined in lsf.shared.

default

The reserved host name default indicates all hosts in the cluster not otherwise referenced in the section (by name or by listing its model or type).

CHKPNT

Description

If C, checkpoint copy is enabled. With checkpoint copy, all opened files are automatically copied to the checkpoint directory by the operating system when a process is checkpointed.

Example

```
HOST_NAME   CHKPNT hostA            C
```

Compatibility

Checkpoint copy is only supported on Cray systems.

Default

No checkpoint copy

DISPATCH_WINDOW

Description

The time windows in which jobs from this host, host model, or host type are dispatched. Once dispatched, jobs are no longer affected by the dispatch window.

Default

Not defined (always open)

EXIT_RATE

Description

Specifies a threshold for exited jobs. If the job exit rate is exceeded for 5 minutes or the period specified by JOB_EXIT_RATE_DURATION in lsb.params, LSF invokes LSF_SERVERDIR/eadmin to trigger a host exception.

EXIT_RATE for a specific host overrides a default GLOBAL_EXIT_RATE specified in lsb.params.

Example

The following Host section defines a job exit rate of 20 jobs for all hosts, and an exit rate of 10 jobs on host A.

```
Begin Host
HOST_NAME     MXJ       EXIT_RATE  # Keywords
Default       !         20
hostA         !         10
End Host
```

Default

Not defined

JL/U

Description

Per-user job slot limit for the host. Maximum number of job slots that each user can use on this host.

Example

```
HOST_NAME  JL/U hostA        2
```

Default

Unlimited

MIG

Syntax

MIG=_minutes_

Description

Enables automatic job migration and specifies the migration threshold for checkpointable or rerunnable jobs, in minutes.

LSF automatically migrates jobs that have been in the SSUSP state for more than the specified number of minutes. Specify a value of 0 to migrate jobs immediately upon suspension. The migration threshold applies to all jobs running on the host.

Job-level command line migration threshold overrides threshold configuration in application profile and queue. Application profile configuration overrides queue level configuration. When a host migration threshold is specified, and is lower than the value for the job, the queue, or the application, the host value is used.

Does not affect MultiCluster jobs that are forwarded to a remote cluster.

Default

Not defined. LSF does not migrate checkpointable or rerunnable jobs automatically.

MXJ

Description

The number of job slots on the host.

With MultiCluster resource leasing model, this is the number of job slots on the host that are available to the local cluster.

Use "!" to make the number of job slots equal to the number of CPUs on a host.

For the reserved host name default, "!" makes the number of job slots equal to the number of CPUs on all hosts in the cluster not otherwise referenced in the section.

By default, the number of running and suspended jobs on a host cannot exceed the number of job slots. If preemptive scheduling is used, the suspended jobs are not counted as using a job slot.

On multiprocessor hosts, to fully use the CPU resource, make the number of job slots equal to or greater than the number of processors.

Default

Unlimited

load_index

Syntax

```
load_index loadSched[/loadStop]
```

Specify `io`, `it`, `ls`, `mem`, `pg`, `r15s`, `r1m`, `r15m`, `swp`, `tmp`, `ut`, or a non-shared custom external load index as a column. Specify multiple columns to configure thresholds for multiple load indices.

Description

Scheduling and suspending thresholds for dynamic load indices supported by LIM, including external load indices.

Each load index column must contain either the default entry or two numbers separated by a slash '/', with no white space. The first number is the scheduling threshold for the load index; the second number is the suspending threshold.

Queue-level scheduling and suspending thresholds are defined in `lsb.queues`. If both files specify thresholds for an index, those that apply are the most restrictive ones.

Example

```
HOST_NAME      mem      swp hostA          100/10   200/30
```

This example translates into a `loadSched` condition of

```
mem>=100 && swp>=200
```

and a `loadStop` condition of

```
mem < 10 || swp < 30
```

Default

Not defined

Example of a Host section

```
Begin Host
HOST_NAME    MXJ    JL/U  r1m        pg         DISPATCH_WINDOW
hostA        1      -     0.6/1.6    10/20      (5:19:00-1:8:30 20:00-8:30)
SUNSOL       1      -     0.5/2.5 -             23:00-8:00
default      2      1     0.6/1.6    20/40            ()
End Host
```

SUNSOL is a host type defined in lsf.shared. This example Host section configures one host and one host type explicitly and configures default values for all other load-sharing hosts.

HostA runs one batch job at a time. A job will only be started on hostA if the r1m index is below 0.6 and the pg index is below 10; the running job is stopped if the r1m index goes above 1.6 or the pg index goes above 20. HostA only accepts batch jobs from 19:00 on Friday evening until 8:30 Monday morning and overnight from 20:00 to 8:30 on all other days.

For hosts of type SUNSOL, the pg index does not have host-specific thresholds and such hosts are only available overnight from 23:00 to 8:00.

The entry with host name default applies to each of the other hosts in the cluster. Each host can run up to two jobs at the same time, with at most one job from each user. These hosts are available to run jobs at all times. Jobs may be started if the r1m index is below 0.6 and the pg index is below 20, and a job from the lowest priority queue is suspended if r1m goes above 1.6 or pg goes above 40.

HostGroup section

Description

Optional. Defines host groups.

The name of the host group can then be used in other host group, host partition, and queue definitions, as well as on the command line. Specifying the name of a host group has exactly the same effect as listing the names of all the hosts in the group.

Structure

Host groups are specified in the same format as user groups in lsb.users.

The first line consists of two mandatory keywords, GROUP_NAME and GROUP_MEMBER, as well as an optional keywords, CONDENSE and GROUP_ADMIN. Subsequent lines name a group and list its membership.

The sum of all host groups, compute groups, and host partitions cannot be more than 1024.

GROUP_NAME

Description

An alphanumeric string representing the name of the host group.

You cannot use the reserved name all, and group names must not conflict with host names.

CONDENSE

Description

Optional. Defines condensed host groups.

Condensed host groups are displayed in a condensed output format for the bhosts and bjobs commands.

If you configure a host to belong to more than one condensed host group, bjobs can display any of the host groups as execution host name.

Valid Values

Y or N.

Default

N (the specified host group is not condensed)

GROUP_MEMBER

Description

A space-delimited list of host names or previously defined host group names, enclosed in one pair of parentheses.

You cannot use more than one pair of parentheses to define the list.

The names of hosts and host groups can appear on multiple lines because hosts can belong to multiple groups. The reserved name all specifies all hosts in the cluster. An exclamation mark (!) indicates an externally-defined host group, which the egroup executable retrieves.

Pattern definition

You can use string literals and special characters when defining host group members. Each entry cannot contain any spaces, as the list itself is space delimited.

When a leased-in host joins the cluster, the host name is in the form of *host@cluster*. For these hosts, only the host part of the host name is subject to pattern definitions.

You can use the following special characters to specify host group members:

- Use a tilde (~) to exclude specified hosts or host groups from the list.
- Use an asterisk (*) as a wildcard character to represent any number of characters.
- Use square brackets with a hyphen ([*integer1 - integer2*]) to define a range of non-negative integers at the end of a host name. The first integer must be less than the second integer.
- Use square brackets with commas ([*integer1*, *integer2*...]) to define individual non-negative integers at the end of a host name.
- Use square brackets with commas and hyphens (for example, [*integer1 - integer2*, *integer3*, *integer4 - integer5*]) to define different ranges of non-negative integers at the end of a host name.

Restrictions

- You cannot use more than one set of square brackets in a single host group definition.

- The following example is *not* correct:

```
...  (hostA[1-10]B[1-20] hostC[101-120])
```

- The following example is correct:

```
...  (hostA[1-20] hostC[101-120])
```

- You cannot define subgroups that contain wildcards and special characters.

GROUP_ADMIN

Description

Host group administrators have the ability to open or close the member hosts for the group they are administering.

the GROUP_ADMIN field is a space-delimited list of user names or previously defined user group names, enclosed in one pair of parentheses.

You cannot use more than one pair of parentheses to define the list.

The names of users and user groups can appear on multiple lines because users can belong to and administer multiple groups.

When host group administrators (who are not also cluster administrators) open or close a host, they must specify a comment with the -C option.

Valid values

Any existing user or user group can be specified. A user group that specifies an external list is also allowed; however, in this location, you use the user group name that has been defined with (!) rather than (!) itself.

Restrictions

- You cannot specify any wildcards or special characters (for example: *, !, $, #, &, ~).
- You cannot specify an external group (egroup).
- You cannot use the keyword ALL and you cannot administer any group that has ALL as its members.
- User names and user group names cannot have spaces.

Example HostGroup sections

Example 1

```
Begin HostGroup
GROUP_NAME   GROUP_MEMBER  GROUP_ADMIN
groupA       (hostA hostD)  (user1 user10)
groupB       (hostF groupA hostK)  ()
groupC       (!)  ()
End HostGroup
```

This example defines three host groups:

- groupA includes hostA and hostD and can be administered by user1 and user10.
- groupB includes hostF and hostK, along with all hosts in groupA. It has no administrators (only the cluster administrator can control the member hosts).

- The group membership of `groupC` is defined externally and retrieved by the `egroup` executable.

Example 2

```
Begin HostGroup
GROUP_NAME     GROUP_MEMBER GROUP_ADMIN
groupA         (all) ()
groupB         (groupA ~hostA ~hostB) (user11 user14)
groupC         (hostX hostY hostZ) ()
groupD         (groupC ~hostX) usergroupB
groupE         (all ~groupC ~hostB) ()
groupF         (hostF groupC hostK) ()
End HostGroup
```

This example defines the following host groups:

- `groupA` contains all hosts in the cluster and is administered by the cluster administrator.
- `groupB` contains all the hosts in the cluster except for `hostA` and `hostB` and is administered by user11 and user14.
- `groupC` contains only `hostX`, `hostY`, and `hostZ` and is administered by the cluster administrator.
- `groupD` contains the hosts in `groupC` except for `hostX`. Note that `hostX` must be a member of host group `groupC` to be excluded from `groupD`. `usergroupB` is the administrator for `groupD`.
- `groupE` contains all hosts in the cluster excluding the hosts in `groupC` and `hostB` and is administered by the cluster administrator.
- `groupF` contains `hostF`, `hostK`, and the 3 hosts in `groupC` and is administered by the cluster administrator.

Example 3

```
Begin HostGroup
GROUP_NAME     CONDENSE     GROUP_MEMBER GROUP_ADMIN
groupA         N            (all) ()
groupB         N            (hostA, hostB) (usergroupC user1)
groupC         Y            (all)()
End HostGroup
```

This example defines the following host groups:

- `groupA` shows uncondensed output and contains all hosts in the cluster and is administered by the cluster administrator.
- `groupB` shows uncondensed output, and contains `hostA` and `hostB`. It is administered by all members of usergroupC and user1.
- `groupC` shows condensed output and contains all hosts in the cluster and is administered by the cluster administrator.

Example 4

```
Begin HostGroup

GROUP_NAME CONDENSE GROUP_MEMBER GROUP_ADMIN

groupA          Y (host*)  (user7)

groupB          N (*A)  ()

groupC          N (hostB* ~hostB[1-50])  ()

groupD          Y (hostC[1-50] hostC[101-150])  (usergroupJ)

groupE          N (hostC[51-100] hostC[151-200])  ()

groupF          Y (hostD[1,3] hostD[5-10])  ()

groupG          N (hostD[11-50] ~hostD[15,20,25] hostD2)  ()

End HostGroup
```

This example defines the following host groups:

- groupA shows condensed output, and contains all hosts starting with the string host. It is administered by user7.
- groupB shows uncondensed output, and contains all hosts ending with the string A, such as hostA and is administered by the cluster administrator.
- groupC shows uncondensed output, and contains all hosts starting with the string hostB except for the hosts from hostB1 to hostB50 and is administered by the cluster administrator.
- groupD shows condensed output, and contains all hosts from hostC1 to hostC50 and all hosts from hostC101 to hostC150 and is administered by the the members of usergroupJ.
- groupE shows uncondensed output, and contains all hosts from hostC51 to hostC100 and all hosts from hostC151 to hostC200 and is administered by the cluster administrator.
- groupF shows condensed output, and contains hostD1, hostD3, and all hosts from hostD5 to hostD10 and is administered by the cluster administrator.
- groupG shows uncondensed output, and contains all hosts from hostD11 to hostD50 except for hostD15, hostD20, and hostD25. groupG also includes hostD2. It is administered by the cluster administrator.

HostPartition section

Description

Optional. Used with host partition user-based fairshare scheduling. Defines a host partition, which defines a user-based fairshare policy at the host level.

Configure multiple sections to define multiple partitions.

The members of a host partition form a host group with the same name as the host partition.

> **Restriction:**
>
> You cannot use host partitions and host preference simultaneously.

Limitations on queue configuration

- If you configure a host partition, you cannot configure fairshare at the queue level.
- If a queue uses a host that belongs to a host partition, it should not use any hosts that don't belong to that partition. All the hosts in the queue should belong to the same partition. Otherwise, you might notice unpredictable scheduling behavior:
    - Jobs in the queue sometimes may be dispatched to the host partition even though hosts not belonging to any host partition have a lighter load.
    - If some hosts belong to one host partition and some hosts belong to another, only the priorities of one host partition are used when dispatching a parallel job to hosts from more than one host partition.

Shared resources and host partitions

- If a resource is shared among hosts included in host partitions and hosts that are not included in any host partition, jobs in queues that use the host partitions will always get the shared resource first, regardless of queue priority.
- If a resource is shared among host partitions, jobs in queues that use the host partitions listed first in the HostPartition section of lsb.hosts will always have priority to get the shared resource first. To allocate shared resources among host partitions, LSF considers host partitions in the order they are listed in lsb.hosts.

Structure

Each host partition always consists of 3 lines, defining the name of the partition, the hosts included in the partition, and the user share assignments.

HPART_NAME

Syntax

HPART_NAME=*partition_name*

Description

Specifies the name of the partition. The name must be 59 characters or less.

HOSTS

Syntax

HOSTS=[[~]*host_name* | [~]*host_group* | all]...

Description

Specifies the hosts in the partition, in a space-separated list.

A host cannot belong to multiple partitions.

A host group cannot be empty.

Hosts that are not included in any host partition are controlled by the FCFS scheduling policy instead of the fairshare scheduling policy.

Optionally, use the reserved host name all to configure a single partition that applies to all hosts in a cluster.

Optionally, use the not operator (~) to exclude hosts or host groups from the list of hosts in the host partition.

Examples

HOSTS=all ~hostK ~hostM

The partition includes all the hosts in the cluster, except for host K and host M.

```
HOSTS=groupA ~hostL
```

The partition includes all the hosts in host group groupA except for host L.

USER_SHARES

Syntax

USER_SHARES=[*user*, *number_shares*]...

Description

Specifies user share assignments

- Specify at least one user share assignment.
- Enclose each user share assignment in square brackets, as shown.
- Separate a list of multiple share assignments with a space between the square brackets.
- *user*—Specify users who are also configured to use the host partition. You can assign the shares:

  - To a single user (specify *user_name*). To specify a Windows user account, include the domain name in uppercase letters (*DOMAIN_NAME\user_name*).
  - To users in a group, individually (specify *group_name@*) or collectively (specify *group_name*). To specify a Windows user group, include the domain name in uppercase letters (*DOMAIN_NAME\group_name*).
  - To users not included in any other share assignment, individually (specify the keyword default) or collectively (specify the keyword others).

By default, when resources are assigned collectively to a group, the group members compete for the resources according to FCFS scheduling. You can use hierarchical fairshare to further divide the shares among the group members.

When resources are assigned to members of a group individually, the share assignment is recursive. Members of the group and of all subgroups always compete for the resources according to FCFS scheduling, regardless of hierarchical fairshare policies.

- *number_shares*

  - Specify a positive integer representing the number of shares of the cluster resources assigned to the user.
  - The number of shares assigned to each user is only meaningful when you compare it to the shares assigned to other users or to the total number of shares. The total number of shares is just the sum of all the shares assigned in each share assignment.

Example of a HostPartition section

```
Begin HostPartition
```
```
HPART_NAME = Partition1 HOSTS = hostA hostB USER_SHARES = [groupA@, 3]
[groupB, 7] [default, 1]
```
```
End HostPartition
```

ComputeUnit section

Description

Optional. Defines compute units.

Once defined, the compute unit can be used in other compute unit and queue definitions, as well as in the command line. Specifying the name of a compute unit has the same effect as listing the names of all the hosts in the compute unit.

Compute units are similar to host groups, with the added feature of granularity allowing the construction of structures that mimic the network architecture. Job scheduling using compute unit resource requirements effectively spreads jobs over the cluster based on the configured compute units.

To enforce consistency, compute unit configuration has the following requirements:

- Hosts and host groups appear in the finest granularity compute unit type, and nowhere else.
- Hosts appear in only one compute unit of the finest granularity.
- All compute units of the same type have the same type of compute units (or hosts) as members.

Structure

Compute units are specified in the same format as host groups in lsb.hosts.

The first line consists of three mandatory keywords, NAME, MEMBER, and TYPE, as well as an optional keywords CONDENSE and ADMIN. Subsequent lines name a compute unit and list its membership.

The sum of all host groups, compute groups, and host partitions cannot be more than 1024.

NAME

Description

An alphanumeric string representing the name of the compute unit.

You cannot use the reserved names all, allremote, others, and default. Compute unit names must not conflict with host names, host partitions, or host group names.

CONDENSE

Description

Optional. Defines condensed compute units.

Condensed compute units are displayed in a condensed output format for the bhosts and bjobs commands. The condensed compute unit format includes the slot usage for each compute unit.

Valid Values

Y or N.

Default

N (the specified host group is not condensed)

MEMBER
Description

A space-delimited list of host names or previously defined compute unit names, enclosed in one pair of parentheses.

You cannot use more than one pair of parentheses to define the list.

The names of hosts and host groups can appear only once, and only in a compute unit type of the finest granularity.

An exclamation mark (!) indicates an externally-defined host group, which the egroup executable retrieves.

Pattern definition

You can use string literals and special characters when defining compute unit members. Each entry cannot contain any spaces, as the list itself is space delimited.

You can use the following special characters to specify host and host group compute unit members:

- Use a tilde (~) to exclude specified hosts or host groups from the list.
- Use an asterisk (*) as a wildcard character to represent any number of characters.
- Use square brackets with a hyphen ([*integer1 - integer2*]) to define a range of non-negative integers at the end of a host name. The first integer must be less than the second integer.
- Use square brackets with commas ([*integer1*, *integer2...*]) to define individual non-negative integers at the end of a host name.
- Use square brackets with commas and hyphens (for example, [*integer1 - integer2*, *integer3*, *integer4 - integer5*]) to define different ranges of non-negative integers at the end of a host name.

Restrictions

- You cannot use more than one set of square brackets in a single compute unit definition.

  - The following example is *not* correct:
    ```
    ... (encl A[1-10] B[1-20] encl C[101-120])
    ```
  - The following example is correct:
    ```
    ... (encl A[1-20] encl C[101-120])
    ```
- Compute unit names cannot be used in compute units of the finest granularity.
- You cannot include host or host group names except in compute units of the finest granularity.
- You must not skip levels of granularity. For example:

If `lsb.params` contains `COMPUTE_UNIT_TYPES=enclosure rack cabinet` then a compute unit of type `cabinet` can contain compute units of type `rack`, but not of type `enclosure`.

* The keywords all, allremote, all@cluster, other and default cannot be used when defining compute units.

TYPE

Description

The type of the compute unit, as defined in the COMPUTE_UNIT_TYPES parameter of `lsb.params`.

ADMIN

Description

Host group administrators have the ability to open or close the member hosts for the compute unit they are administering.

the `ADMIN` field is a space-delimited list of user names or previously defined user group names, enclosed in one pair of parentheses.

You cannot use more than one pair of parentheses to define the list.

The names of users and user groups can appear on multiple lines because users can belong to and administer multiple compute units.

When host group administrators (who are not also cluster administrators) open or close a host, they must specify a comment with the -C option.

Valid values

Any existing user or user group can be specified. A user group that specifies an external list is also allowed; however, in this location, you use the user group name that has been defined with (!) rather than (!) itself.

Restrictions

* You cannot specify any wildcards or special characters (for example: *, !, $, #, &, ~).
* You cannot specify an external group (egroup).
* You cannot use the keyword ALL and you cannot administer any group that has ALL as its members.
* User names and user group names cannot have spaces.

Example ComputeUnit sections

Example 1

(For the `lsb.params` entry
`COMPUTE_UNIT_TYPES=enclosure rack cabinet`

)

```
Begin ComputeUnit
```

NAME	MEMBER	TYPE
encl1	(host1 host2)	enclosure
encl2	(host3 host4)	enclosure
encl3	(host5 host6)	enclosure
encl4	(host7 host8)	enclosure
rack1	(encl1 encl2)	rack
rack2	(encl3 encl4)	rack
cbnt1	(rack1 rack2)	cabinet

```
End ComputeUnit
```

This example defines seven compute units:

- encl1, encl2, encl3 and encl4 are the finest granularity, and each contain two hosts.
- rack1 is of coarser granularity and contains two levels. At the enclosure level rack1 contains encl1 and encl2. At the lowest level rack1 contains host1, host2, host3, and host4.
- rack2 has the same structure as rack1, and contains encl3 and encl4.
- cbnt1 contains two racks (rack1 and rack2), four enclosures (encl1, encl2, encl3, and encl4) and all eight hosts. Compute unit cbnt1 is the coarsest granularity in this example.

Example 2

(For the lsb.params entry COMPUTE_UNIT_TYPES=enclosure rack cabinet)

```
Begin ComputeUnit
```

NAME	CONDENSE	MEMBER	TYPE	ADMIN
encl1	Y	(hg123 ~hostA ~hostB)	enclosure	(user11 user14)
encl2	Y	(hg456)	enclosure	()
encl3	N	(hostA hostB)	enclosure	usergroupB
encl4	N	(hgroupX ~hostB)	enclosure	()
encl5	Y	(hostC* ~hostC[101-150])	enclosure	usergroupJ
encl6	N	(hostC[101-150])	enclosure	()
rack1	Y	(encl1 encl2 encl3)	rack	()
rack2	N	(encl4 encl5)	rack	usergroupJ
rack3	N	(encl6)	rack	()
cbnt1	Y	(rack1 rack2)	cabinet	()
cbnt2	N	(rack3)	cabinet	user14

```
End ComputeUnit
```

This example defines 11 compute units:

- All six enclosures (finest granularity) contain only hosts and host groups. All three racks contain only enclosures. Both cabinets (coarsest granularity) contain only racks.
- encl1 contains all the hosts in host group hg123 except for hostA and hostB and is administered by user11 and user14. Note that hostA and hostB must be members of host group hg123 to be excluded from encl1. encl1 shows condensed output.

- encl 2 contains host group hg456 and is administered by the cluster administrator. encl 2 shows condensed output.
- encl 3 contains host A and host B. usergroupB is the administrator for encl 3. encl 3 shows uncondensed output.
- encl 4 contains host group hgroupX except for host B. Since each host can appear in only one enclosure and host B is already in encl 3, it cannot be in encl 4. encl 4 is administered by the cluster administrator. encl 4 shows uncondensed output.
- encl 5 contains all hosts starting with the string host C except for hosts host C101 to host C150, and is administered by usergroupJ. encl 5 shows condensed output.
- rack1 contains encl 1, encl 2, and encl 3. rack1 shows condensed output.
- rack2 contains encl 4, and encl 5. rack2 shows uncondensed output.
- rack3 contains encl 6. rack3 shows uncondensed output.
- cbnt1 contains rack1 and rack2. cbnt 1 shows condensed output.
- cbnt 2 contains rack3. Even though rack3 only contains encl6, cbnt 3 cannot contain encl 6 directly because that would mean skipping the level associated with compute unit type rack. cbnt 2 shows uncondensed output.

Automatic time-based configuration

Variable configuration is used to automatically change LSF configuration based on time windows. You define automatic configuration changes in lsb.hosts by using if-else constructs and time expressions. After you change the files, reconfigure the cluster with the badmin reconfig command.

The expressions are evaluated by LSF every 10 minutes based on mbatchd start time. When an expression evaluates true, LSF dynamically changes the configuration based on the associated configuration statements. Reconfiguration is done in real time without restarting mbatchd, providing continuous system availability.

Example

In the following example, the #if, #else, #endif are not interpreted as comments by LSF but as if-else constructs.

```
Begin Host

HOST_NAME    r15s    r1m    pg

host1         3/5     3/5    12/20

#if time(5:16:30-1:8:30 20:00-8:30)

host2         3/5     3/5    12/20

#else

0host2        2/3     2/3    10/12

#endif

host3         3/5     3/5    12/20

End Host
```

lsb.modules

The `lsb.modules` file contains configuration information for LSF scheduler and resource broker modules. The file contains only one section, named PluginModule.

This file is optional. If no scheduler or resource broker modules are configured, LSF uses the default scheduler plugin modules named schmod_default and schmod_fcfs.

The `lsb.modules` file is stored in the directory `LSB_CONFDIR`/*cluster_name*/`configdir`, where LSB_CONFDIR is defined in `lsf.conf`.

Changing lsb.modules configuration

After making any changes to `lsb.modules`, run `badmin reconfig` to reconfigure mbatchd.

PluginModule section

Description

Defines the plugin modules for the LSF scheduler and LSF resource broker. If this section is not configured, LSF uses the default scheduler plugin modules named schmod_default and schmod_fcfs, which enable the LSF default scheduling features.

Example PluginModule section

The following PluginModule section enables all scheduling policies provided by LSF:

```
Begin PluginModule
```

SCH_PLUGIN	RB_PLUGIN	SCH_DISABLE_PHASES
schmod_default	()	()
schmod_fairshare	()	()
schmod_fcfs	()	()
schmod_limit	()	()
schmod_parallel	()	()
schmod_reserve	()	()
schmod_preemption	()	()
schmod_advrsv	()	()
schmod_mc	()	()
schmod_jobweight	()	()
schmod_cpuset	()	()
schmod_pset	()	()
schmod_ps	()	()
schmod_aps	()	()

```
End PluginModule
```

PluginModule section structure

The first line consists of the following keywords:

- SCH_PLUGIN

- • RB_PLUGIN
- • SCH_DISABLE_PHASES

They identify the scheduler plugins, resource broker plugins, and the scheduler phase to be disabled for the plugins that you wish to configure.

Each subsequent line describes the configuration information for one scheduler plugin module, resource broker plugin module, and scheduler phase, if any, to be disabled for the plugin. Each line must contain one entry for each keyword. Use empty parentheses ( ) or a dash (-) to specify the default value for an entry.

SCH_PLUGIN

Description

Required. The SCH_PLUGIN column specifies the shared module name for the LSF scheduler plugin. Each plugin requires a corresponding license. Scheduler plugins are called in the order they are listed in the PluginModule section.

By default, all shared modules for scheduler plugins are located in LSF_LIBDIR. On UNIX, you can also specify a full path to the name of the scheduler plugin.

The following modules are supplied with LSF:

schmod_default

Enables the default LSF scheduler features.

Licensed by: LSF_Manager

schmod_fcfs

Enables the first-come, first-served (FCFS) scheduler features. schmod_fcfs can appear anywhere in the SCH_PLUGIN list. By default, if schmod_fcfs is not configured in lsb.modules, it is loaded automatically along with schmod_default.

Source code (sch.mod.fcfs.c) for the schmod_fcfs scheduler plugin module is installed in the directory

```
LSF_TOP/7.0/misc/examples/external_plugin/
```

Use the LSF scheduler plugin SDK to modify the FCFS scheduler module code to suit the job scheduling requirements of your site.

See *Platform LSF Programmer's Guide* for more detailed information about writing, building, and configuring your own custom scheduler plugins.

schmod_fairshare

Enables the LSF fairshare scheduling features.

schmod_limit

Enables the LSF resource allocation limit features.

Licensed by: LSF_Manager

schmod_parallel

Enables scheduling of parallel jobs submitted with bsub -n.

schmod_reserve

Enables the LSF resource reservation features.

To enable processor reservation, backfill, and memory reservation for parallel jobs, you must configure both schmod_parallel and schmod_reserve in lsb.modules. If only schmod_reserve is configured, backfill and memory reservation are enabled only for sequential jobs, and processor reservation is not enabled.

schmod_preemption

Enables the LSF preemption scheduler features.

schmod_advrsv

Handles jobs that use advance reservations (brsvadd, brsvs, brsvdel, bsub -U)

schmod_cpuset

Handles jobs that use IRIX cpusets (bsub -ext[sched] "CPUSET[*cpuset_options*]")

The schmod_cpuset plugin name must be configured after the standard LSF plugin names in the PluginModule list.

schmod_mc

Enables MultiCluster job forwarding

Licensed by: LSF_MultiCluster

schmod_ps

Enables resource ownership functionality of EGO-enabled SLA scheduling policies

schmod_pset

Enables scheduling policies required for jobs that use HP-UX processor sets (pset) allocations (bsub -ext[sched] "PSET[*topology*]")

The schmod_pset plugin name must be configured after the standard LSF plugin names in the PluginModule list.

schmod_aps

Enables absolute priority scheduling (APS) policies configured by APS_PRIORITY in lsb.queues.

The schmod_aps plugin name must be configured after the schmod_fairshare plugin name in the PluginModule list, so that the APS value can override the fairshare job ordering decision.

Licensed by: LSF_HPC

schmod_jobweight

An optional scheduler plugin module to enable Cross-Queue Job Weight scheduling policies. The schmod_jobweight plugin must be listed before schmod_cpuset and schmod_rms, and after all other scheduler plugin modules.

You should not use job weight scheduling together with fairshare scheduling or job preemption. To avoid scheduling conflicts, you should comment out schmod_fairshare and schmod_preemption in lsb.modules.

Scheduler plugin SDK

Use the LSF scheduler plugin SDK to write customized scheduler modules that give you more flexibility and control over job scheduling. Enable your custom scheduling policies by configuring your modules under SCH_PLUGIN in the PluginModules section of lsb.modules.

The directory

```
LSF_TOP/7.0/misc/examples/external_plugin/
```

contains sample plugin code. See *Platform LSF Programmer's Guide* for more detailed information about writing, building, and configuring your own custom scheduler plugins.

RB_PLUGIN

Description

RB_PLUGIN specifies the shared module name for resource broker plugins. Resource broker plugins collect and update job resource accounting information, and provide it to the scheduler.

Normally, for each scheduler plugin module, there is a corresponding resource broker plugin module to support it. However, the resource broker also supports multiple plugin modules for one scheduler plugin module.

For example, a fairshare policy may need more than one resource broker plugin module to support it if the policy has multiple configurations.

A scheduler plugin can have one, multiple, or none RB plugins corresponding to it.

Example

NAME	RB_PLUGIN
schmod_default	()
schmod_fairshare	(rb_fairshare)

Default

Undefined

SCH_DISABLE_PHASES

Description

SCH_DISABLE_PHASES specifies which scheduler phases, if any, to be disabled for the plugin. LSF scheduling has four phases:

1. Preprocessing — the scheduler checks the readiness of the job for scheduling and prepares a list of ready resource seekers. It also checks the start time of a job, and evaluates any job dependencies.
2. Match/limit — the scheduler evaluates the job resource requirements and prepares candidate hosts for jobs by matching jobs with resources. It also applies resource allocation limits. Jobs with all required resources matched go on to order/allocation phase. Not all

jobs are mapped to all potential available resources. Jobs without any matching resources will not go through the Order/Allocation Phase but can go through the Post-processing phase, where preemption may be applied to get resources the job needs to run.

3. Order/allocation — the scheduler sorts jobs with matched resources and allocates resources for each job, assigning job slot, memory, and other resources to the job. It also checks if the allocation satisfies all constraints defined in configuration, such as queue slot limit, deadline for the job, etc.

   1. In the order phase, the scheduler applies policies such as FCFS, Fairshare and Host-partition and consider job priorities within user groups and share groups. By default, job priority within a pool of jobs from the same user is based on how long the job has been pending.

   2. For resource intensive jobs (jobs requiring a lot of CPUs or a large amount of memory), resource reservation is performed so that these jobs are not starved.

   3. When all the currently available resources are allocated, jobs go on to post-processing.

4. Post-processing — the scheduler prepares jobs from the order/allocation phase for dispatch and applies preemption or backfill policies to obtain resources for the jobs that have completed pre-processing or match/limit phases, but did not have resources available to enter the next scheduling phase.

Each scheduler plugin module invokes one or more scheduler phase. The processing for a give phase can be disabled or skipped if:

The plugin module does not need to do any processing for that phase or the processing has already been done by a previous plugin module in the list.

The scheduler will not invoke phases marked by SCH_DISABLE_PHASES when scheduling jobs.

None of the plugins provided by LSF should require phases to be disabled, but your own custom plugin modules using the scheduler SDK may need to disable one or more scheduler phases.

Example

In the following configuration, the `schmod_custom` plugin module disables the order allocation (3) and post-processing (4) phases:

NAME	SCH_DISABLE_PHASES
schmod_default	()
schmod_custom	(3, 4)

Default

Undefined

lsb.params

The lsb.params file defines general parameters used by the LSF system. This file contains only one section, named Parameters. mbatchd uses lsb.params for initialization. The file is optional. If not present, the LSF-defined defaults are assumed.

Some of the parameters that can be defined in lsb.params control timing within the system. The default settings provide good throughput for long-running batch jobs while adding a minimum of processing overhead in the batch daemons.

This file is installed by default in LSB_CONFDIR/*cluster_name*/configdir.

Changing lsb.params configuration

After making any changes to lsb.params, run badmin reconfig to reconfigure mbatchd.

Parameters section

This section and all the keywords in this section are optional. If keywords are not present, the default values are assumed.

Parameters set at installation

The following parameter values are set at installation for the purpose of testing a new cluster:

```
Begin Parameters
DEFAULT_QUEUE  = normal      #default job queue name
MBD_SLEEP_TIME = 20          #mbatchd scheduling interval (60 secs is default)
SBD_SLEEP_TIME = 15          #sbatchd scheduling interval (30 secs is default)
JOB_ACCEPT_INTERVAL = 1      #interval for any host to accept a job
                             #(default is 1 (one-fold of MBD_SLEEP_TIME))
End Parameters
```

With this configuration, jobs submitted to the LSF system will be started on server hosts quickly. If this configuration is not suitable for your production use, you should either remove the parameters to take the default values, or adjust them as needed.

For example, to avoid having jobs start when host load is high, increase JOB_ACCEPT_INTERVAL so that the job scheduling interval is longer to give hosts more time to adjust load indices after accepting jobs.

In production use, you should define DEFAULT_QUEUE to the normal queue, MBD_SLEEP_TIME to 60 seconds (the default), and SBD_SLEEP_TIME to 30 seconds (the default).

ABS_RUNLIMIT

Syntax

ABS_RUNLIMIT=y | Y

Description

If set, absolute (wall-clock) run time is used instead of normalized run time for all jobs submitted with the following values:

- Run time limit specified by the -W option of bsub
- RUNLIMIT queue-level parameter in lsb.queues
- RUNLIMIT application-level parameter in lsb.applications
- RUNTIME parameter in lsb.applications

The run time estimates and limits are not normalized by the host CPU factor.

Default

N (run limit and run time estimate are normalized)

ACCT_ARCHIVE_AGE

Syntax

ACCT_ARCHIVE_AGE=*days*

Description

Enables automatic archiving of LSF accounting log files, and specifies the archive interval. LSF archives the current log file if the length of time from its creation date exceeds the specified number of days.

See also

- ACCT_ARCHIVE_SIZE also enables automatic archiving
- ACCT_ARCHIVE_TIME also enables automatic archiving
- MAX_ACCT_ARCHIVE_FILE enables automatic deletion of the archives

Default

-1 (Not defined; no limit to the age of lsb.acct)

ACCT_ARCHIVE_SIZE

Syntax

ACCT_ARCHIVE_SIZE=*kilobytes*

Description

Enables automatic archiving of LSF accounting log files, and specifies the archive threshold. LSF archives the current log file if its size exceeds the specified number of kilobytes.

See also

- ACCT_ARCHIVE_SIZE also enables automatic archiving
- ACCT_ARCHIVE_TIME also enables automatic archiving
- MAX_ACCT_ARCHIVE_FILE enables automatic deletion of the archives

Default

-1 (Not defined; no limit to the size of lsb.acct)

ACCT_ARCHIVE_TIME

Syntax

ACCT_ARCHIVE_TIME=*hh:mm*

Description

Enables automatic archiving of LSF accounting log file lsb.acct, and specifies the time of day to archive the current log file.

See also

- ACCT_ARCHIVE_SIZE also enables automatic archiving
- ACCT_ARCHIVE_TIME also enables automatic archiving
- MAX_ACCT_ARCHIVE_FILE enables automatic deletion of the archives

Default

Not defined (no time set for archiving lsb.acct)

CHUNK_JOB_DURATION

Syntax

CHUNK_JOB_DURATION=*minutes*

Description

Specifies a CPU limit, run limit, or estimated run time for jobs submitted to a chunk job queue to be chunked.

When CHUNK_JOB_DURATION is set, the CPU limit or run limit set at the queue level (CPULIMIT or RUNLMIT), application level (CPULIMIT or RUNLIMIT), or job level (-c or -W bsub options), or the run time estimate set at the application level (RUNTIME) must be less than or equal to CHUNK_JOB_DURATION for jobs to be chunked.

If CHUNK_JOB_DURATION is set, jobs are *not* chunked if:

- No CPU limit, run time limit, or run time estimate is specified at any level, or
- A CPU limit, run time limit, or run time estimate is greater than the value of CHUNK_JOB_DURATION.

The value of CHUNK_JOB_DURATION is displayed by bparams -l.

Examples

- CHUNK_JOB_DURATION is not defined:
  - Jobs with no CPU limit, run limit, or run time estimate are chunked
  - Jobs with a CPU limit, run limit, or run time estimate less than or equal to 30 are chunked

- Jobs with a CPU limit, run limit, or run time estimate greater than 30 are *not* chunked
- CHUNK_JOB_DURATION=90:

  - Jobs with no CPU limit, run limit, or run time estimate are *not* chunked
  - Jobs with a CPU limit, run limit, or run time estimate less than or equal to 90 are chunked
  - Jobs with a CPU limit, run limit, or run time estimate greater than 90 are *not* chunked

Default

-1 (Not defined.)

CLEAN_PERIOD

Syntax

CLEAN_PERIOD=*seconds*

Description

For non-repetitive jobs, the amount of time that job records for jobs that have finished or have been killed are kept in mbatchd core memory after they have finished.

Users can still see all jobs after they have finished using the bjobs command.

For jobs that finished more than CLEAN_PERIOD seconds ago, use the bhist command.

Default

3600 (1 hour)

COMMITTED_RUN_TIME_FACTOR

Syntax

COMMITTED_RUN_TIME_FACTOR=*number*

Description

Used only with fairshare scheduling. Committed run time weighting factor.

In the calculation of a user's dynamic priority, this factor determines the relative importance of the committed run time in the calculation. If the -W option of bsub is not specified at job submission and a RUNLIMIT has not been set for the queue, the committed run time is not considered.

Valid Values

Any positive number between 0.0 and 1.0

Default

0.0

COMPUTE_UNIT_TYPES

Syntax

COMPUTE_UNIT_TYPES=_type1 type2..._

Description

Used to define valid compute unit types for topological resource requirement allocation.

The order in which compute unit types appear specifies the containment relationship between types. Finer grained compute unit types appear first, followed by the coarser grained type that contains them, and so on.

At most one compute unit type in the list can be followed by an exclamation mark designating it as the default compute unit type. If no exclamation mark appears, the first compute unit type in the list is taken as the default type.

Valid Values

Any space-separated list of alphanumeric strings.

Default

Not defined

Example

COMPUTE_UNIT_TYPES=cell enclosure! rack

Specifies three compute unit types, with the default type enclosure. Compute units of type rack contain type enclosure, and of type enclosure contain type cell.

CONDENSE_PENDING_REASONS

Syntax

CONDENSE_PENDING_REASONS=ALL | **PARTIAL** |**N**

Description

Set to **ALL**, condenses all host-based pending reasons into one generic pending reason. This is equivalent to setting **CONDENSED_PENDING_REASON=Y**.

Set to **PARTIAL**, condenses all host-based pending reasons except shared resource pending reasons into one generic pending reason.

If enabled, you can request a full pending reason list by running the following command:
badmin diagnose _jobId_

> **Tip:**
>
> You must be LSF administrator or a queue administrator to run this command.

Examples

- CONDENSE_PENDING_REASONS=ALL If a job has no other pending reason, bjobs -p or bjobs -l displays the following:

  ```
  Individual host based reasons
  ```

- CONDENSE_PENDING_REASONS=N The pending reasons are not suppressed. Host-based pending reasons are displayed.

Default

N

CPU_TIME_FACTOR

Syntax

CPU_TIME_FACTOR=_number_

Description

Used only with fairshare scheduling. CPU time weighting factor.

In the calculation of a user's dynamic share priority, this factor determines the relative importance of the cumulative CPU time used by a user's jobs.

Default

0.7

DEFAULT_APPLICATION

Syntax

DEFAULT_APPLICATION=_application_profile_name_

Description

The name of the default application profile. The application profile must already be defined in lsb.applications.

When you submit a job to LSF without explicitly specifying an application profile, LSF associates the job with the specified application profile.

Default

Not defined. When a user submits a job without explicitly specifying an application profile, and no default application profile is defined by this parameter, LSF does not associate the job with any application profile.

DEFAULT_HOST_SPEC

Syntax

DEFAULT_HOST_SPEC=_host_name_ | _host_model_

Description

The default CPU time normalization host for the cluster.

The CPU factor of the specified host or host model will be used to normalize the CPU time limit of all jobs in the cluster, unless the CPU time normalization host is specified at the queue or job level.

Default

Not defined

DEFAULT_JOBGROUP

Syntax

DEFAULT_JOBGROUP=*job_group_name*

Description

The name of the default job group.

When you submit a job to LSF without explicitly specifying a job group, LSF associates the job with the specified job group. The LSB_DEFAULT_JOBGROUP environment variable overrrides the setting of DEFAULT_JOBGROUP. The bsub -g *job_group_name* option overrides both LSB_DEFAULT_JOBGROUP and DEFAULT_JOBGROUP.

Default job group specification supports macro substitution for project name (%p) and user name (%u). When you specify bsub -P *project_name*, the value of %p is the specified project name. If you do not specify a project name at job submission, %p is the project name defined by setting the environment variable LSB_DEFAULTPROJECT, or the project name specified by DEFAULT_PROJECT in lsb.params. the default project name is default.

For example, a default job group name specified by DEFAULT_JOBGROUP=/canada/%p/%u is expanded to the value for the LSF project name and the user name of the job submission user (for example, /canada/projects/user1).

Job group names must follow this format:

- Job group names must start with a slash character (/). For example, DEFAULT_JOBGROUP=/A/B/C is correct, but DEFAULT_JOBGROUP=A/B/C is not correct.
- Job group names cannot end with a slash character (/). For example, DEFAULT_JOBGROUP=/A/ is not correct.
- Job group names cannot contain more than one slash character (/) in a row. For example, job group names like DEFAULT_JOBGROUP=/A//B or DEFAULT_JOBGROUP=A////B are not correct.
- Job group names cannot contain spaces. For example, DEFAULT_JOBGROUP=/A/B C/D is not correct.
- Project names and user names used for macro substitution with %p and %u cannot start or end with slash character (/).
- Project names and user names used for macro substitution with %p and %u cannot contain spaces or more than one slash character (/) in a row.
- Project names or user names containing slash character (/) will create separate job groups. For example, if the project name is canada/projects, DEFAULT_JOBGROUP=/%p results in a job group hierarchy /canada/projects.

Example

```
DEFAULT_JOBGROUP=/canada/projects
```

Default

Not defined. When a user submits a job without explicitly specifying job group name, and the LSB_DEFAULT_JOBGROUP environment variable is not defined, LSF does not associate the job with any job group.

DEFAULT_PROJECT

Syntax

DEFAULT_PROJECT=*project_name*

Description

The name of the default project. Specify any string.

When you submit a job without specifying any project name, and the environment variable LSB_DEFAULTPROJECT is not set, LSF automatically assigns the job to this project.

Default

```
default
```

DEFAULT_QUEUE

Syntax

DEFAULT_QUEUE=*queue_name* ...

Description

Space-separated list of candidate default queues (candidates must already be defined in lsb.queues).

When you submit a job to LSF without explicitly specifying a queue, and the environment variable LSB_DEFAULTQUEUE is not set, LSF puts the job in the first queue in this list that satisfies the job's specifications subject to other restrictions, such as requested hosts, queue status, etc.

Default

This parameter is set at installation to DEFAULT_QUEUE=normal .

When a user submits a job to LSF without explicitly specifying a queue, and there are no candidate default queues defined (by this parameter or by the user's environment variable LSB_DEFAULTQUEUE), LSF automatically creates a new queue named default, using the default configuration, and submits the job to that queue.

DEFAULT_SLA_VELOCITY

Syntax

DEFAULT_SLA_VELOCITY=*num_slots*

Description

For EGO-enabled SLA scheduling, the number of slots that the SLA should request for parallel jobs running in the SLA.

By default, an EGO-enabled SLA requests slots from EGO based on the number of jobs the SLA needs to run. If the jobs themselves require more than one slot, they will remain pending. To avoid this for parallel jobs, set DEFAULT_SLA_VELOCITY to the total number of slots that are expected to be used by parallel jobs.

Default

1

DETECT_IDLE_JOB_AFTER

Syntax

DETECT_IDLE_JOB_AFTER=*time_minutes*

Description

The minimum job run time before mbatchd reports that the job is idle.

Default

20 (mbatchd checks if the job is idle after 20 minutes of run time)

DISABLE_UACCT_MAP

Syntax

DISABLE_UACCT_MAP=y | Y

Description

Specify y or Y to disable user-level account mapping.

Default

N

EADMIN_TRIGGER_DURATION

Syntax

EADMIN_TRIGGER_DURATION=*minutes*

Description

Defines how often LSF_SERVERDIR/eadmin is invoked once a job exception is detected. Used in conjunction with job exception handling parameters JOB_IDLE, JOB_OVERRUN, and JOB_UNDERRUN in lsb.queues.

Tip:

Tune EADMIN_TRIGGER_DURATION carefully. Shorter values
may raise false alarms, longer values may not trigger exceptions
frequently enough.

Example

```
EADMIN_TRIGGER_DURATION=5
```

Default

1 minute

ENABLE_DEFAULT_EGO_SLA

Syntax

ENABLE_DEFAULT_EGO_SLA=*service_class_name* | *consumer_name*

Description

The name of the default service class or EGO consumer name for EGO-enabled SLA
scheduling. If the specified SLA does not exist in lsb.serviceclasses, LSF creates one with
the specified consumer name, velocity of 1, priority of 1, and a time window that is always
open.

If the name of the default SLA is not configured in lsb.servicesclasses, it must be the
name of a valid EGO consumer.

ENABLE_DEFAULT_EGO_SLA is required to turn on EGO-enabled SLA scheduling. All
LSF resource management is delegated to Platform EGO, and all LSF hosts are under EGO
control. When all jobs running in the default SLA finish, all allocated hosts are released to
EGO after the default idle timeout of 120 seconds (configurable by MAX_HOST_IDLE_TIME
in lsb.servicecelasses).

When you submit a job to LSF without explicitly using the -sla option to specify a service class
name, LSF puts the job in the default service class specified by *service_class_name*.

Default

Not defined. When a user submits a job to LSF without explicitly specifying a service class,
and there is no default service class defined by this parameter, LSF does not attach the job to
any service class.

ENABLE_EVENT_STREAM

Syntax

ENABLE_EVENT_STREAM=Y | **N**

Description

Used only with event streaming for system performance analysis tools, such as the Platform
LSF reporting feature.

Default

N (event streaming is not enabled)

ENABLE_EXIT_RATE_PER_SLOT

Syntax

ENABLE_EXIT_RATE_PER_SLOT=Y | **N**

Description

Scales the actual exit rate thresholds on a host according to the number of slots on the host. For example, if EXIT_RATE=2 in lsb.hosts or GLOBAL_EXIT_RATE=2 in lsb.params, and the host has 2 job slots, the job exit rate threshold will be 4.

Default

N

ENABLE_HIST_RUN_TIME

Syntax

ENABLE_HIST_RUN_TIME=y | **Y**

Description

Used only with fairshare scheduling. If set, enables the use of historical run time in the calculation of fairshare scheduling priority.

Default

N

ENABLE_HOST_INTERSECTION

Syntax

ENABLE_HOST_INTERSECTION=Y | N

Description

When enabled, allows job submission to any host that belongs to the intersection created when considering the queue the job was submitted to, any advance reservation hosts, or any hosts specified by bsub -m at the time of submission.

When disabled job submission with hosts specified can be accepted only if specified hosts are a subset of hosts defined in the queue.

The following commands are affected by ENABLE_HOST_INTERSECTION:

* bsub
* bmod
* bmig
* brestart

- `bswitch`

If no hosts exist in the intersection, the job is rejected.

Default

N

ENABLE_USER_RESUME

Syntax

ENABLE_USER_RESUME=Y | N

Description

Defines job resume permissions.

When this parameter is defined:

- If the value is Y, users can resume their own jobs that have been suspended by the administrator.
- If the value is N, jobs that are suspended by the administrator can only be resumed by the administrator or root; users do not have permission to resume a job suspended by another user or the administrator. Administrators can resume jobs suspended by users or administrators.

Default

N (users cannot resume jobs suspended by administrator)

ENFORCE_ONE_UG_LIMITS

Syntax

ENFORCE_ONE_UG_LIMITS=Y | N

Upon job submission with the -G option and when user groups have overlapping members, defines whether only the specified user group's limits (or those of any parent group) are enforced or whether the most restrictive user group limits of any overlapping user/user group are enforced.

- If the value is Y, only the limits defined for the user group that you specify with -G during job submission apply to the job, even if there are overlapping members of groups.

  If you have nested user groups, the limits of a user's group parent also apply.

  View existing limits by running blimits.
- If the value is N and the user group has members that overlap with other user groups, the strictest possible limits (that you can view by running blimits) defined for any of the member user groups are enforced for the job.

Default

N

EVENT_STREAM_FILE

Syntax

EVENT_STREAM_FILE=*file_path*

Description

Determines the path to the event data stream file used by system performance analysis tools such as Platform LSF Reporting.

Default

LSF_TOP/work/*cluster_name*/logdir/stream/lsb.stream

EVENT_UPDATE_INTERVAL

Syntax

EVENT_UPDATE_INTERVAL=*seconds*

Description

Used with duplicate logging of event and accounting log files. LSB_LOCALDIR in lsf.conf must also be specified. Specifies how often to back up the data and synchronize the directories (LSB_SHAREDIR and LSB_LOCALDIR).

If you do not define this parameter, the directories are synchronized when data is logged to the files, or when mbatchd is started on the first LSF master host. If you define this parameter, mbatchd synchronizes the directories only at the specified time intervals.

Use this parameter if NFS traffic is too high and you want to reduce network traffic.

Valid values

1 to INFINIT_INT

INFINIT_INT is defined in lsf.h

Recommended values

Between 10 and 30 seconds, or longer depending on the amount of network traffic.

> **Note:**
>
> Avoid setting the value to exactly 30 seconds, because this will trigger the default behavior and cause mbatchd to synchronize the data every time an event is logged.

Default

-1 (Not defined.)

See also

LSB_LOCALDIR in lsf.conf

EXIT_RATE_TYPE

Syntax

EXIT_RATE_TYPE=[**JOBEXIT** | **JOBEXIT_NONLSF**] [**JOBINIT**] [**HPCINIT**]

Description

When host exception handling is configured (EXIT_RATE in `lsb.hosts` or GLOBAL_EXIT_RATE in `lsb.params`), specifies the type of job exit to be handled.

JOBEXIT

Job exited after it was dispatched and started running.

JOBEXIT_NONLSF

Job exited with exit reasons related to LSF and not related to a host problem (for example, user action or LSF policy). These jobs are not counted in the exit rate calculation for the host.

JOBINIT

Job exited during initialization because of an execution environment problem. The job did not actually start running.

HPCINIT

Job exited during initialization of a Platform LSF HPC because of an execution environment problem. The job did not actually start running.

Default

JOBEXIT_NONLSF

FAIRSHARE_ADJUSTMENT_FACTOR

Syntax

FAIRSHARE_ADJUSTMENT_FACTOR=*number*

Description

Used only with fairshare scheduling. Fairshare adjustment plugin weighting factor.

In the calculation of a user's dynamic share priority, this factor determines the relative importance of the user-defined adjustment made in the fairshare plugin (libfairshareadjust.*).

A positive float number both enables the fairshare plugin and acts as a weighting factor.

Default

0 (user-defined adjustment made in the fairshare plugin not used)

GLOBAL_EXIT_RATE

Syntax

GLOBAL_EXIT_RATE=*number*

Description

Specifies a cluster-wide threshold for exited jobs. If EXIT_RATE is not specified for the host in lsb.hosts, GLOBAL_EXIT_RATE defines a default exit rate for all hosts in the cluster. Host-level EXIT_RATE overrides the GLOBAL_EXIT_RATE value.

If the global job exit rate is exceeded for 5 minutes or the period specified by JOB_EXIT_RATE_DURATION, LSF invokes LSF_SERVERDIR/eadmin to trigger a host exception.

Example

GLOBAL_EXIT_RATE=10 defines a job exit rate of 10 jobs for all hosts.

Default

2147483647 (Unlimited threshold.)

HIST_HOURS

Syntax

HIST_HOURS=*hours*

Description

Used only with fairshare scheduling. Determines a rate of decay for cumulative CPU time and historical run time.

To calculate dynamic user priority, LSF scales the actual CPU time using a decay factor, so that 1 hour of recently-used time is equivalent to 0.1 hours after the specified number of hours has elapsed.

To calculate dynamic user priority with historical run time, LSF scales the accumulated run time of finished jobs using the same decay factor, so that 1 hour of recently-used time is equivalent to 0.1 hours after the specified number of hours has elapsed.

When HIST_HOURS=0, CPU time accumulated by running jobs is not decayed.

Default

5

JOB_ACCEPT_INTERVAL

Syntax

JOB_ACCEPT_INTERVAL=*integer*

Description

The number you specify is multiplied by the value of lsb. params MBD_SLEEP_TIME (60 seconds by default). The result of the calculation is the number of seconds to wait after dispatching a job to a host, before dispatching a second job to the same host.

If 0 (zero), a host may accept more than one job. By default, there is no limit to the total number of jobs that can run on a host, so if this parameter is set to 0, a very large number of jobs might be dispatched to a host all at once. This can overload your system to the point that it will be unable to create any more processes. It is not recommended to set this parameter to 0.

JOB_ACCEPT_INTERVAL set at the queue level (lsb. queues) overrides JOB_ACCEPT_INTERVAL set at the cluster level (lsb. params).

Default

1

JOB_ATTA_DIR

Syntax

JOB_ATTA_DIR=*directory*

Description

The shared directory in which mbatchd saves the attached data of messages posted with the bpost command.

Use JOB_ATTA_DIR if you use bpost and bread to transfer large data files between jobs and want to avoid using space in LSB_SHAREDDIR. By default, the bread command reads attachment data from the JOB_ATTA_DIR directory.

JOB_ATTA_DIR should be shared by all hosts in the cluster, so that any potential LSF master host can reach it. Like LSB_SHAREDIR, the directory should be owned and writable by the primary LSF administrator. The directory must have at least 1 MB of free space.

The attached data will be stored under the directory in the format:

JOB_ATTA_DIR/*timestamp.jobid*.msgs/msg$msgindex

On UNIX, specify an absolute path. For example:

JOB_ATTA_DIR=/opt/share/lsf_work

On Windows, specify a UNC path or a path with a drive letter. For example:

JOB_ATTA_DIR=\\HostA\temp\lsf_work

or

JOB_ATTA_DIR=D:\temp\lsf_work

After adding JOB_ATTA_DIR to lsb. params, use badmin reconfig to reconfigure your cluster.

Valid values

JOB_ATTA_DIR can be any valid UNIX or Windows path up to a maximum length of 256 characters.

Default

Not defined

If JOB_ATTA_DIR is not specified, job message attachments are saved in LSB_SHAREDI R/ i nf o/.

JOB_DEP_LAST_SUB

Description

Used only with job dependency scheduling.

If set to 1, whenever dependency conditions use a job name that belongs to multiple jobs, LSF evaluates only the most recently submitted job.

Otherwise, all the jobs with the specified name must satisfy the dependency condition.

Default

0

JOB_EXIT_RATE_DURATION

Description

Defines how long LSF waits before checking the job exit rate for a host. Used in conjunction with EXIT_RATE in l sb. hosts for LSF host exception handling.

If the job exit rate is exceeded for the period specified by JOB_EXIT_RATE_DURATION, LSF invokes LSF_SERVERDI R/eadmi n to trigger a host exception.

Tuning

Tip:

Tune JOB_EXIT_RATE_DURATION carefully. Shorter values may raise false alarms, longer values may not trigger exceptions frequently enough.

Example

JOB_EXI T_RATE_DURATI ON=10

Default

5 minutes

JOB_GROUP_CLEAN

Syntax

JOB_GROUP_CLEAN=Y | N

Description

If JOB_GROUP_CLEAN = Y, implicitly created job groups that are empty and have no limits assigned to them are automatically deleted.

Default

N (Implicitly created job groups are not automatically deleted unless they are deleted manually with bgdel .)

JOB_INCLUDE_POSTPROC

Syntax

JOB_INCLUDE_POSTPROC=Y | N

Description

Specifies whether LSF includes the post-execution processing of the job as part of the job. When set to Y:

- Prevents a new job from starting on a host until post-execution processing is finished on that host
- Includes the CPU and run times of post-execution processing with the job CPU and run times
- sbatchd sends both job finish status (DONE or EXIT) and post-execution processing status (POST_DONE or POST_ERR) to mbatchd at the same time

In MultiCluster job forwarding model, the JOB_INCLUDE_POSTPROC value in the receiving cluster applies to the job.

MultiCluster job lease model, the JOB_INCLUDE_POSTPROC value applies to jobs running on remote leased hosts as if they were running on local hosts.

The variable LSB_JOB_INCLUDE_POSTPROC in the user environment overrides the value of JOB_INCLUDE_POSTPROC in an application profile in lsb. applications. JOB_INCLUDE_POSTPROC in an application profile in lsb. applications overrides the value of JOB_INCLUDE_POSTPROC in lsb. params.

For SGI cpusets, if JOB_INCLUDE_POSTPROC=Y, LSF does not release the cpuset until post-execution processing has finished, even though post-execution processes are not attached to the cpuset.

Default

N (Post-execution processing is not included as part of the job, and a new job can start on the execution host before post-execution processing finishes.)

JOB_POSITION_CONTROL_BY_ADMIN

Syntax

JOB_POSITION_CONTROL_BY_ADMIN=Y | **N**

Description

Allows LSF administrators to control whether users can use btop and bbot to move jobs to the top and bottom of queues. When JOB_POSITION_CONTROL_BY_ADMIN=Y, only the

LSF administrator (including any queue administrators) can use bbot and btop to move jobs within a queue.

Default

N

See also

bbot, btop

JOB_POSTPROC_TIMEOUT

Syntax

JOB_POSTPROC_TIMEOUT=*minutes*

Description

Specifies a timeout in minutes for job post-execution processing. The specified timeout must be greater than zero.

If post-execution processing takes longer than the timeout, sbatchd reports that post-execution has failed (POST_ERR status), and kills the entire process group of the job's post-execution processes on UNIX and Linux. On Windows, only the parent process of the post-execution command is killed when the timeout expires. The child processes of the post-execution command are not killed.

If JOB_INCLUDE_POSTPROC=Y, and sbatchd kills the post-execution processes because the timeout has been reached, the CPU time of the post-execution processing is set to 0, and the job's CPU time does not include the CPU time of post-execution processing.

JOB_POSTPROC_TIMEOUT defined in an application profile in lsb.applications overrides the value in lsb.params. JOB_POSTPROC_TIMEOUT cannot be defined in user environment.

In MultiCluster job forwarding model, the JOB_POSTPROC_TIMEOUT value in the receiving cluster applies to the job.

MultiCluster job lease model, the JOB_POSTPROC_TIMEOUT value applies to jobs running on remote leased hosts as if they were running on local hosts.

Default

2147483647 (Unlimited; post-execution processing does not time out.)

JOB_PRIORITY_OVER_TIME

Syntax

JOB_PRIORITY_OVER_TIME=*increment*/*interval*

Description

JOB_PRIORITY_OVER_TIME enables automatic job priority escalation when MAX_USER_PRIORITY is also defined.

Valid Values

increment

Specifies the value used to increase job priority every *interval* minutes. Valid values are positive integers.

interval

Specifies the frequency, in minutes, to *increment* job priority. Valid values are positive integers.

Default

-1 (Not defined.)

Example

JOB_PRIORITY_OVER_TIME=3/20

Specifies that every 20 minute *interval increment* to job priority of pending jobs by 3.

See also

MAX_USER_PRIORITY

JOB_RUNLIMIT_RATIO

Syntax

JOB_RUNLIMIT_RATIO=*integer* | **0**

Description

Specifies a ratio between a job run limit and the runtime estimate specified by bsub -We or bmod -We, -We+, -Wep. The ratio does not apply to the RUNTIME parameter in lsb.applications.

This ratio can be set to 0 and no restrictions are applied to the runtime estimate.

JOB_RUNLIMIT_RATIO prevents abuse of the runtime estimate. The value of this parameter is the ratio of run limit divided by the runtime estimate.

By default, the ratio value is 0. Only administrators can set or change this ratio. If the ratio changes, it only applies to newly submitted jobs. The changed value does not retroactively reapply to already submitted jobs.

If the ratio value is greater than 0:

- If the users specifiy a runtime estimate only (bsub -We), the job-level run limit will automatically be set to *runtime_ratio * runtime_estimate*. Jobs running longer than this run limit are killed by LSF. If the job-level run limit is greater than the hard run limit in the queue, the job is rejected.
- If the users specify a runtime estimate (-We) and job run limit (-W) at job submission, and the run limit is greater than *runtime_ratio * runtime_estimate*, the job is rejected.
- If the users modify the run limit to be greater than *runtime_ratio*, they must increase the runtime estimate first (bmod -We). Then they can increase the default run limit.
- LSF remembers the run limit is set with bsub -W or convert from *runtime_ratio * runtime_estimate*. When users modify the run limit with bmod -Wn, the run limit is

automatically be set to *runtime_ratio* * *runtime_estimate* If the run limit is set from *runtime_ratio*, LSF rejects the run limit modification.

- If users modify the runtime estimate with bmod -We and the run limit is set by the user, the run limit is MIN(*new_estimate* * *new_ratio*, *run_limit*). If the run limit is set by *runtime_ratio*, the run limit is set to *new_estimate* * *new_ratio*.
- If users modify the runtime estimate by using bmod -Wen and the run limit is set by the user, it is not changed. If the run limit is set by *runtime_ratio*, it is set to unlimited.

In MultiCluster job forwarding model, JOB_RUNLIMIT_RATIO valuese in both the sending and receiving clusters apply to the job. The run limit in the receiving cluster cannot be greater than the value of *runtime* * JOB_RUNLIMIT_RATIO in the receiving cluster. Some examples:

- Run limit (for example with bsub -We) is 10, JOB_RUNLIMIT_RATIO=5 in the sending cluster, JOB_RUNLIMIT_RATIO=0 in the receiving cluster—run limit=50, and the job will run
- Run limit (for example with bsub -We) is 10, JOB_RUNLIMIT_RATIO=5 in the sending cluster, JOB_RUNLIMIT_RATIO=3 in the receiving cluster—run limit=50, and the job will pend
- Run limit (for example with bsub -We) is 10, JOB_RUNLIMIT_RATIO=5 in the sending cluster, JOB_RUNLIMIT_RATIO=6 in the receiving cluster—run limit=50, and the job will run
- Run limit (for example with bsub -We) is 10, JOB_RUNLIMIT_RATIO=0 in the sending cluster, JOB_RUNLIMIT_RATIO=5 in the receiving cluster—run limit=50, and the job will run

MultiCluster job lease model, the JOB_RUNLIMIT_RATIO value applies to jobs running on remote leased hosts as if they were running on local hosts.

Default

0

JOB_SCHEDULING_INTERVAL

Syntax

JOB_SCHEDULING_INTERVAL=*milliseconds*

Description

Time interval at which mbatchd sends jobs for scheduling to the scheduling daemon mbschd along with any collected load information.

If set to 0, there is no interval between job scheduling sessions.

Valid Value

Number of seconds greater than or equal to zero (0).

Default

5000 milliseconds

JOB_SPOOL_DIR

Syntax

JOB_SPOOL_DIR=_dir_

Description

Specifies the directory for buffering batch standard output and standard error for a job.

When JOB_SPOOL_DIR is defined, the standard output and standard error for the job is buffered in the specified directory.

Files are copied from the submission host to a temporary file in the directory specified by the JOB_SPOOL_DIR on the execution host. LSF removes these files when the job completes.

If JOB_SPOOL_DIR is not accessible or does not exist, files are spooled to the default job output directory $HOME/.lsbatch.

For bsub -is and bsub -Zs, JOB_SPOOL_DIR must be readable and writable by the job submission user, and it must be shared by the master host and the submission host. If the specified directory is not accessible or does not exist, and JOB_SPOOL_DIR is specified, bsub -is cannot write to the default directory LSB_SHAREDIR/_cluster_name_/lsf_indir, and bsub -Zs cannot write to the default directory LSB_SHAREDIR/_cluster_name_/lsf_cmddir, and the job will fail.

As LSF runs jobs, it creates temporary directories and files under JOB_SPOOL_DIR. By default, LSF removes these directories and files after the job is finished. See bsub for information about job submission options that specify the disposition of these files.

On UNIX, specify an absolute path. For example:

```
JOB_SPOOL_DIR=/home/share/lsf_spool
```

On Windows, specify a UNC path or a path with a drive letter. For example:

```
JOB_SPOOL_DIR=\\HostA\share\spooldir
```

or

```
JOB_SPOOL_DIR=D:\share\spooldir
```

In a mixed UNIX/Windows cluster, specify one path for the UNIX platform and one for the Windows platform. Separate the two paths by a pipe character (|):

```
JOB_SPOOL_DIR=/usr/share/lsf_spool | \\HostA\share\spooldir
```

Valid value

JOB_SPOOL_DIR can be any valid path.

The entire path including JOB_SPOOL_DIR can up to 4094 characters on UNIX and Linux or up to 255 characters for Windows. This maximum path length includes:

- All directory and file paths attached to the JOB_SPOOL_DIR path
- Temporary directories and files that the LSF system creates as jobs run.

The path you specify for JOB_SPOOL_DIR should be as short as possible to avoid exceeding this limit.

Default

Not defined

Batch job output (standard output and standard error) is sent to the .lsbatch directory on the execution host:

- On UNIX: $HOME/.lsbatch
- On Windows: %windir%\lsbtmp*user_id*\.lsbatch

If %HOME% is specified in the user environment, uses that directory instead of %windir% for spooled output.

JOB_TERMINATE_INTERVAL

Syntax

JOB_TERMINATE_INTERVAL=*seconds*

Description

UNIX only.

Specifies the time interval in seconds between sending SIGINT, SIGTERM, and SIGKILL when terminating a job. When a job is terminated, the job is sent SIGINT, SIGTERM, and SIGKILL in sequence with a sleep time of JOB_TERMINATE_INTERVAL between sending the signals. This allows the job to clean up if necessary.

Default

10 (seconds)

LOCAL_MAX_PREEXEC_RETRY

Syntax

LOCAL_MAX_PREEXEC_RETRY=*integer*

Description

The maximum number of times to attempt the pre-execution command of a job on the local cluster.

Valid values

0 < LOCAL_MAX_PREEXEC_RETRY < INFINIT_INT

INFINIT_INT is defined in lsf.h.

Default

2147483647 (Unlimited number of preexec retry times.)

LSB_STOP_ASKING_LICENSES_TO_LS

Syntax

LSB_STOP_ASKING_LICENSES_TO_LS=n | N | y | Y

Description

Used in conjunction with LSF License Scheduler, optimizes the licenses available to pending jobs. Setting this parameter to y stops mbatchd from requesting a license from the License Scheduler daemon when a job cannot run—even with a license—because of other pending reasons. For example, if jobA pends because of a license shortage, and, while jobA waits for a license, other jobs fill all available slots, mbatchd stops requesting a license for jobA. This prevents jobA from obtaining a license that it cannot use, which makes more licenses available for other jobs. Once the slots required by jobA become available, mbatchd requests a license for jobA. If no license is available, jobA pends until it gets a license.

Default

N. A job that was initially pending due to a license requirement and is now pending for another reason continues to request a license.

LSB_SYNC_HOST_STAT_LIM

Syntax

LSB_SYNC_HOST_STAT_LIM=y | Y

Description

Improves the speed with which mbatchd obtains host status, and therefore the speed with which LSF reschedules rerunnable jobs: the sooner LSF knows that a host has become unavailable, the sooner LSF reschedules any rerunnable jobs executing on that host. Useful for a large cluster.

When you define this parameter, mbatchd periodically obtains the host status from the master LIM, and then verifies the status by polling each sbatchd at an interval defined by the parameters MBD_SLEEP_TIME and LSB_MAX_PROBE_SBD.

Default

N. mbatchd obtains and reports host status, without contacting the master LIM, by polling each sbatchd at an interval defined by the parameters MBD_SLEEP_TIME and LSB_MAX_PROBE_SBD.

See also

MBD_SLEEP_TIME

LSB_MAX_PROBE_SBD in lsf.conf

MAX_ACCT_ARCHIVE_FILE

Syntax

MAX_ACCT_ARCHIVE_FILE=*integer*

Description

Enables automatic deletion of archived LSF accounting log files and specifies the archive limit.

Compatibility

ACCT_ARCHIVE_SIZE or ACCT_ARCHIVE_AGE should also be defined.

Example

```
MAX_ACCT_ARCHIVE_FILE=10
```

LSF maintains the current lsb.acct and up to 10 archives. Every time the old lsb.acct.9 becomes lsb.acct.10, the old lsb.acct.10 gets deleted.

See also

- ACCT_ARCHIVE_AGE also enables automatic archiving
- ACCT_ARCHIVE_SIZE also enables automatic archiving
- ACCT_ARCHIVE_TIME also enables automatic archiving

Default

-1 (Not defined. No deletion of lsb.acct.*n* files).

MAX_CONCURRENT_JOB_QUERY

Syntax

MAX_CONCURRENT_JOB_QUERY=*integer*

Description

Defines how many concurrent job queries mbatchd can handle.

- If mbatchd is using multithreading, a dedicated query port is defined by the parameter LSB_QUERY_PORT in lsf.conf. When mbatchd has a dedicated query port, the value of MAX_CONCURRENT_JOB_QUERY sets the maximum number of job queries for each child mbatchd that is forked by mbatchd. This means that the total number of job queries can be more than the number specified by MAX_CONCURRENT_JOB_QUERY.
- If mbatchd is not using multithreading, the value of MAX_CONCURRENT_JOB_QUERY sets the maximum total number of job queries.

Valid values

1-100

Default

2147483647 (Unlimited concurrent job queries.)

See also

LSB_QUERY_PORT in lsf.conf

MAX_EVENT_STREAM_FILE_NUMBER

Syntax

MAX_EVENT_STREAM_FILE_NUMBER=*integer*

Description

Determines the maximum number of different `lsb.stream.utc` files that `mbatchd` uses. If the number of `lsb.stream.utc` files reaches this number, `mbatchd` logs and error message to the `mbd.log` file and stops writing events to the `lsb.stream` file.

Default

10

MAX_EVENT_STREAM_SIZE

Syntax

MAX_EVENT_STREAM_SIZE=*integer*

Description

Determines the maximum size in MB of the `lsb.stream` file used by system performance analysis tools such as Platform LSF Reporting. For LSF Reporting, the recommended size is 2000 MB.

When the MAX_EVENT_STREAM_SIZE size is reached, LSF logs a special event EVENT_END_OF_STREAM, closes the stream and moves it to `lsb.stream.0` and a new stream is opened.

All applications that read the file once the event EVENT_END_OF_STREAM is logged should close the file and reopen it.

Recommended value

2000 MB

Default

1024 MB

MAX_INFO_DIRS

Syntax

MAX_INFO_DIRS=*num_subdirs*

Description

The number of subdirectories under the LSB_SHAREDIR/*cluster_name*/`logdir`/`info` directory.

When MAX_INFO_DIRS is enabled, `mbatchd` creates the specified number of subdirectories in the `info` directory. These subdirectories are given an integer as its name, starting with 0 for the first subdirectory. `mbatchd` writes the job files of all new submitted jobs into these subdirectories using the following formula to choose the subdirectory in which to store the job file:

```
subdirectory = jobID % MAX_INFO_DIRS
```

This formula ensures an even distribution of job files across the subdirectories.

> **Important:**
>
> If you are using local duplicate event logging, you must run
> badmin mbdrestart after changing MAX_INFO_DIRS for the
> changes to take effect.

Valid values

0-1024

Default

0 (no subdirectories under the info directory; mbatchd writes all jobfiles to the info
directory)

Example

MAX_INFO_DIRS=10

mbatchd creates ten subdirectories from LSB_SHAREDIR/*cluster_name*/logdir/info/0 to
LSB_SHAREDIR/*cluster_name*/logdir/info/9.

MAX_JOB_ARRAY_SIZE

Syntax

MAX_JOB_ARRAY_SIZE=*integer*

Description

Specifies the maximum number of jobs in a job array that can be created by a user for a single
job submission. The maximum number of jobs in a job array cannot exceed this value.

A large job array allows a user to submit a large number of jobs to the system with a single job
submission.

Valid values

Specify a positive integer between 1 and 2147483646

Default

1000

MAX_JOB_ATTA_SIZE

Syntax

MAX_JOB_ATTA_SIZE=*integer* | **0**

Specify any number less than 20000.

Description

Maximum attached data size, in KB, that can be transferred to a job.

Maximum size for data attached to a job with the bpost command. Useful if you use bpost and bread to transfer large data files between jobs and you want to limit the usage in the current working directory.

0 indicates that jobs cannot accept attached data files.

Default

2147483647 (Unlimited; LSF does not set a maximum size of job attachments.)

MAX_JOB_MSG_NUM

Syntax

MAX_JOB_MSG_NUM=*integer* | **0**

Description

Maximum number of message slots for each job. Maximum number of messages that can be posted to a job with the bpost command.

0 indicates that jobs cannot accept external messages.

Default

128

MAX_JOB_NUM

Syntax

MAX_JOB_NUM=*integer*

Description

The maximum number of finished jobs whose events are to be stored in the lsb.events log file.

Once the limit is reached, mbatchd starts a new event log file. The old event log file is saved as lsb.events.*n*, with subsequent sequence number suffixes incremented by 1 each time a new log file is started. Event logging continues in the new lsb.events file.

Default

1000

MAX_JOB_PREEMPT

Syntax

MAX_JOB_PREEMPT=*integer*

Description

The maximum number of times a job can be preempted. Applies to queue-level jobs only.

Valid values

0 < MAX_JOB_PREEMPT < INFINIT_INT

INFINIT_INT is defined in lsf.h.

Default

2147483647 (Unlimited number of preemption times.)

MAX_JOB_REQUEUE

Syntax

MAX_JOB_REQUEUE=*integer*

Description

The maximum number of times to requeue a job automatically.

Valid values

0 < MAX_JOB_REQUEUE < INFINIT_INT

INFINIT_INT is defined in lsf.h.

Default

2147483647 (Unlimited number of requeue times.)

MAX_JOBID

Syntax

MAX_JOBID=*integer*

Description

The job ID limit. The job ID limit is the highest job ID that LSF will ever assign, and also the maximum number of jobs in the system.

By default, LSF assigns job IDs up to 6 digits. This means that no more than 999999 jobs can be in the system at once.

Specify any integer from 999999 to 2147483646 (for practical purposes, you can use any 10-digit integer less than this value).

You cannot lower the job ID limit, but you can raise it to 10 digits. This allows longer term job accounting and analysis, and means you can have more jobs in the system, and the job ID numbers will roll over less often.

LSF assigns job IDs in sequence. When the job ID limit is reached, the count rolls over, so the next job submitted gets job ID "1". If the original job 1 remains in the system, LSF skips that number and assigns job ID "2", or the next available job ID. If you have so many jobs in the system that the low job IDs are still in use when the maximum job ID is assigned, jobs with sequential numbers could have totally different submission times.

Example

MAX_JOBID=125000000

Default

999999

MAX_JOBINFO_QUERY_PERIOD

Syntax

MAX_JOBINFO_QUERY_PERIOD=*integer*

Description

Maximum time for job information query commands (for example, with bj obs) to wait.

When the time arrives, the query command processes exit, and all associated threads are terminated.

If the parameter is not defined, query command processes will wait for all threads to finish.

Specify a multiple of MBD_REFRESH_TIME.

Valid values

Any positive integer greater than or equal to one (1)

Default

2147483647 (Unlimited wait time.)

See also

LSB_BLOCK_JOBINFO_TIMEOUT in lsf.conf

MAX_PEND_JOBS

Syntax

MAX_PEND_JOBS=*integer*

Description

The maximum number of pending jobs in the system.

This is the hard system-wide pending job threshold. No user or user group can exceed this limit unless the job is forwarded from a remote cluster.

If the user or user group submitting the job has reached the pending job threshold as specified by MAX_PEND_JOBS, LSF will reject any further job submission requests sent by that user or user group. The system will continue to send the job submission requests with the interval specified by SUB_TRY_INTERVAL in lsb.params until it has made a number of attempts equal to the LSB_NTRIES environment variable. If LSB_NTRIES is not defined and LSF rejects the job submission request, the system will continue to send the job submission requests indefinitely as the default behavior.

Default

2147483647 (Unlimited number of pending jobs.)

See also

SUB_TRY_INTERVAL

MAX_PREEXEC_RETRY

Syntax

MAX_PREEXEC_RETRY=*integer*

Description

MultiCluster job forwarding model only. The maximum number of times to attempt the pre-execution command of a job from a remote cluster.

If the job's pre-execution command fails all attempts, the job is returned to the submission cluster.

Valid values

0 < MAX_PREEXEC_RETRY < INFINIT_INT

INFINIT_INT is defined in lsf.h.

Default

5

MAX_SBD_FAIL

Syntax

MAX_SBD_FAIL=*integer*

Description

The maximum number of retries for reaching a non-responding slave batch daemon, sbatchd.

The interval between retries is defined by MBD_SLEEP_TIME. If mbatchd fails to reach a host and has retried MAX_SBD_FAIL times, the host is considered unreachable.

If you define LSB_SYNC_HOST_STAT_LIM=Y, mbatchd obtains the host status from the master LIM before it polls sbatchd. When the master LIM reports that a host is unavailable (LIM is down) or unreachable (sbatchd is down) MAX_SBD_FAIL number of times, mbatchd reports the host status as unavailable or unreachable.

When a host becomes unavailable, mbatchd assumes that all jobs running on that host have exited and that all rerunnable jobs (jobs submitted with the bsub -r option) are scheduled to be rerun on another host.

Default

3

MAX_USER_PRIORITY

Syntax

MAX_USER_PRIORITY=*integer*

Description

Enables user-assigned job priority and specifies the maximum job priority a user can assign to a job.

LSF and queue administrators can assign a job priority higher than the specified value for jobs they own.

Compatibility

User-assigned job priority changes the behavior of btop and bbot.

Example

MAX_USER_PRIORITY=100

Specifies that 100 is the maximum job priority that can be specified by a user.

Default

-1 (Not defined.)

See also

- bsub, bmod, btop, bbot
- JOB_PRIORITY_OVER_TIME

MBD_EGO_CONNECT_TIMEOUT

Syntax

MBD_EGO_CONNECT_TIMEOUT=*seconds*

Description

For EGO-enabled SLA scheduling, timeout parameter for network I/O connection with EGO vemkd.

Default

0 seconds

MBD_EGO_READ_TIMEOUT

Syntax

MBD_EGO_READ_TIMEOUT=*seconds*

Description

For EGO-enabled SLA scheduling, timeout parameter for network I/O read from EGO vemkd after connection with EGO.

Default

0 seconds

MBD_EGO_TIME2LIVE

Syntax

MBD_EGO_TIME2LIVE=*minutes*

Description

For EGO-enabled SLA scheduling, specifies how long EGO should keep information about host allocations in case mbatchd restarts,

Default

0 minutes

MBD_QUERY_CPUS

Syntax

MBD_QUERY_CPUS=*cpu_list*

cpu_list defines the list of master host CPUS on which the mbatchd child query processes can run. Format the list as a white-space delimited list of CPU numbers.

For example, if you specify
MBD_QUERY_CPUS=1 2 3

the mbatchd child query processes will run only on CPU numbers 1, 2, and 3 on the master host.

Description

This parameter allows you to specify the master host CPUs on which mbatchd child query processes can run (hard CPU affinity). This improves mbatchd scheduling and dispatch performance by binding query processes to specific CPUs so that higher priority mbatchd processes can run more efficiently.

When you define this parameter, LSF runs mbatchd child query processes *only* on the specified CPUs. The operating system can assign other processes to run on the same CPU; however, if utilization of the bound CPU is lower than utilization of the unbound CPUs.

Important

1. You can specify CPU affinity only for master hosts that use one of the following operating systems:

    *   Linux 2.6 or higher
    *   Solaris 8 or higher

2. If failover to a master host candidate occurs, LSF maintains the hard CPU affinity, provided that the master host candidate has the same CPU configuration as the original master host. If the configuration differs, LSF ignores the CPU list and reverts to default behavior.

Related parameters

To improve scheduling and dispatch performance of all LSF daemons, you should use MBD_QUERY_CPUS together with EGO_DAEMONS_CPUS (in ego.conf), which controls LIM CPU allocation, and LSF_DAEMONS_CPUS, which binds mbatchd and mbschd daemon processes to specific CPUs so that higher priority daemon processes can run more efficiently. To get best performance, CPU allocation for all four daemons should be assigned their own CPUs. For example, on a 4 CPU SMP host, the following configuration will give the best performance:

```
EGO_DAEMONS_CPUS=0 LSF_DAEMONS_CPUS=1:2 MBD_QUERY_CPUS=3
```

Default

Not defined

See also

LSF_DAEMONS_CPUS in lsf.conf

MBD_REFRESH_TIME

Syntax

MBD_REFRESH_TIME=_seconds_ [_min_refresh_time_]

where _min_refresh_time_ defines the minimum time (in seconds) that the child mbatchd will stay to handle queries. The valid range is 0 - 300.

Description

Time interval, in seconds, when mbatchd will fork a new child mbatchd to service query requests to keep information sent back to clients updated. A child mbatchd processes query requests creating threads.

MBD_REFRESH_TIME applies only to UNIX platforms that support thread programming.

To enable MBD_REFRESH_TIME you must specify LSB_QUERY_PORT in lsf.conf. The child mbatchd listens to the port number specified by LSB_QUERY_PORT and creates threads to service requests until the job changes status, a new job is submitted, or MBD_REFRESH_TIME has expired.

- If MBD_REFRESH_TIME is < _min_refresh_time_, the child mbatchd exits at MBD_REFRESH_TIME even if the job changes status or a new job is submitted before MBD_REFRESH_TIME expires.
- If MBD_REFRESH_TIME > _min_refresh_time_ the child mbatchd exits at _min_refresh_time_ if a job changes status or a new job is submitted before the _min_refresh_time_, or a job exits after the _min_refresh_time_ when a job changes status or a new job is submitted.
- If MBD_REFRESH_TIME > _min_refresh_time_ and no job changes status or a new job is submitted, the child mbatchd exits at MBD_REFRESH_TIME

The value of this parameter must be between 0 and 300. Any values specified out of this range are ignored, and the system default value is applied.

The bj obs command may not display up-to-date information if two consecutive query commands are issued before a child mbatchd expires because child mbatchd job information is not updated. If you use the bj obs command and do not get up-to-date information, you may need to decrease the value of this parameter. Note, however, that the lower the value of this parameter, the more you negatively affect performance.

The number of concurrent requests is limited by the number of concurrent threads that a process can have. This number varies by platform:

- Sun Solaris, 2500 threads per process
- AIX, 512 threads per process
- Digital, 256 threads per process
- HP-UX, 64 threads per process

Default

5 seconds if *min_refresh_time* is not defined or if MBD_REFRESH_TIME is defined value is less than 5; 300 seconds if the defined value is more than 300

min_refresh_time default is 10 seconds

See also

LSB_QUERY_PORT in lsf.conf

MBD_SLEEP_TIME

Syntax

MBD_SLEEP_TIME=*seconds*

Description

Used in conjunction with the parameters SLOT_RESERVE, MAX_SBD_FAIL, and JOB_ACCEPT_INTERVAL.

Amount of time in seconds used for calculating parameter values.

Default

If not defined, 60 seconds. MBD_SLEEP_TIME is set at installation to 20 seconds.

MBD_USE_EGO_MXJ

Syntax

MBD_USE_EGO_MXJ=Y | N

Description

By default, when EGO-enabled SLA scheduling is configured, EGO allocates an entire host to LSF, which uses its own MXJ definition to determine how many slots are available on the host. LSF gets its host allocation from EGO, and runs as many jobs as the LSF configured MXJ for that host dictates.

MBD_USE_EGO_MXJ forces LSF to use the job slot maximum configured in the EGO consumer. This allows partial sharing of hosts (for example, a large SMP computer) among

different consumers or workload managers. When MBD_USE_EGO_MXJ is set, LSF schedules jobs based on the number of slots allocated from EGO. For example, if host A has 4 processors, but EGO allocates 2 slots to an EGO-enabled SLA consumer. LSF can schedule a maximum of 2 jobs from that SLA on host A.

Important:

If you set MBD_USE_EGO_MXJ=Y, you can configure only one service class, including the default SLA.

Default

N (mbatcthd uses the LSF MXJ)

MC_PENDING_REASON_PKG_SIZE

Syntax

MC_PENDING_REASON_PKG_SIZE=*kilobytes* | **0**

Description

MultiCluster job forwarding model only. Pending reason update package size, in KB. Defines the maximum amount of pending reason data this cluster will send to submission clusters in one cycle.

Specify the keyword 0 (zero) to disable the limit and allow any amount of data in one package.

Default

512

MC_PENDING_REASON_UPDATE_INTERVAL

Syntax

MC_PENDING_REASON_UPDATE_INTERVAL=*seconds* | 0

Description

MultiCluster job forwarding model only. Pending reason update interval, in seconds. Defines how often this cluster will update submission clusters about the status of pending MultiCluster jobs.

Specify the keyword 0 (zero) to disable pending reason updating between clusters.

Default

300

MC_RECLAIM_DELAY

Syntax

MC_RECLAIM_DELAY=*minutes*

Description

MultiCluster resource leasing model only. The reclaim interval (how often to reconfigure shared leases) in minutes.

Shared leases are defined by Type=shared in the lsb.resources HostExport section.

Default

10 (minutes)

MC_RUSAGE_UPDATE_INTERVAL

Syntax

MC_RUSAGE_UPDATE_INTERVAL=*seconds*

Description

MultiCluster only. Enables resource use updating for MultiCluster jobs running on hosts in the cluster and specifies how often to send updated information to the submission or consumer cluster.

Default

300

MIN_SWITCH_PERIOD

Syntax

MIN_SWITCH_PERIOD=*seconds*

Description

The minimum period in seconds between event log switches.

Works together with MAX_JOB_NUM to control how frequently mbatchd switches the file. mbatchd checks if MAX_JOB_NUM has been reached every MIN_SWITCH_PERIOD seconds. If mbatchd finds that MAX_JOB_NUM has been reached, it switches the events file.

To significantly improve the performance of mbatchd for large clusters, set this parameter to a value equal to or greater than 600. This causes mbatchd to fork a child process that handles event switching, thereby reducing the load on mbatchd. mbatchd terminates the child process and appends delta events to new events after the MIN_SWITCH_PERIOD has elapsed.

Default

0

No minimum period. Log switch frequency is not restricted.

See also

MAX_PEND_JOBS

NEWJOB_REFRESH

Syntax

NEWJOB_REFRESH=Y | N

Description

Enables a child `mbatchd` to get up to date information about new jobs from the parent `mbatchd`. When set to Y, job queries with `bjobs` display new jobs submitted after the child `mbatchd` was created.

If you have enabled multithreaded mbatchd support, the `bjobs` command may not display up-to-date information if two consecutive query commands are issued before a child `mbatchd` expires because child `mbatchd` job information is not updated. Use NEWJOB_REFRESH=Y to enable the parent `mbatchd` to push new job information to a child `mbatchd`

When NEWJOB_REFRESH=Y, as users submit new jobs, the parent `mbatchd` pushes the new job event to the child `mbatchd`. The parent `mbatchd` transfers the following kinds of new jobs to the child `mbatchd`:

- Newly submitted jobs
- Restarted jobs
- Remote lease model jobs from the submission cluster
- Remote forwarded jobs from the submission cluster

When NEWJOB_REFRESH=Y, you should set MBD_REFRESH_TIME to a value greater than 10 seconds.

Required parameters

LSB_QUERY_PORT must be enabled in `lsf.conf`.

Restrictions

The parent `mbatchd` only pushes the new job event to a child `mbatchd`. The child `mbatchd` is not aware of status changes of existing jobs. The child `mbatchd` will not reflect the results of job control commands (`bmod`, `bmig`, `bswitch`, `btop`, `bbot`, `brequeue`, `bstop`, `bresume`, and so on) invoked after the child `mbatchd` is created.

Default

N (Not defined. New jobs are not pushed to the child `mbatchd`.)

See also

MBD_REFRESH_TIME

NO_PREEMPT_FINISH_TIME

Syntax

NO_PREEMPT_FINISH_TIME=*minutes* | *percentage*

Description

Prevents preemption of jobs that will finish within the specified number of minutes or the specified percentage of the estimated run time or run limit.

Specifies that jobs due to finish within the specified number of minutes or percentage of job duration should not be preempted, where *minutes* is wall-clock time, not normalized time. Percentage must be greater than 0 or less than 100% (between 1% and 99%).

For example, if the job run limit is 60 minutes and NO_PREEMPT_FINISH_TIME=10%, the job cannot be preempted after it running 54 minutes or longer.

If you specify percentage for NO_PREEMPT_RUN_TIME, requires a run time (bsub -We or RUNTIME in lsb.applications), or run limit to be specified for the job (bsub -W, or RUNLIMIT in lsb.queues, or RUNLIMIT in lsb.applications)

Default

-1 (Not defined.)

NO_PREEMPT_RUN_TIME

Syntax

NO_PREEMPT_RUN_TIME=*minutes* | *percentage*

Description

Prevents preemption of jobs that have been running for the specified number of minutes or the specified percentage of the estimated run time or run limit.

Specifies that jobs that have been running for the specified number of minutes or longer should not be preempted, where *minutes* is wall-clock time, not normalized time. Percentage must be greater than 0 or less than 100% (between 1% and 99%).

For example, if the job run limit is 60 minutes and NO_PREEMPT_RUN_TIME=50%, the job cannot be preempted after it running 30 minutes or longer.

If you specify percentage for NO_PREEMPT_RUN_TIME, requires a run time (bsub -We or RUNTIME in lsb.applications), or run limit to be specified for the job (bsub -W, or RUNLIMIT in lsb.queues, or RUNLIMIT in lsb.applications)

Default

-1 (Not defined.)

NQS_QUEUES_FLAGS

Syntax

NQS_QUEUES_FLAGS=integer

Description

For Cray NQS compatibility only. Used by LSF to get the NQS queue information.

If the NQS version on a Cray is NQS 1.1, 80.42 or NQS 71.3, this parameter does not need to be defined.

For other versions of NQS on Cray, define both NQS_QUEUES_FLAGS and NQS_REQUESTS_FLAGS.

To determine the value of this parameter, run the NQS qstat command. The value of Npk_int [1] in the output is the value you need for this parameter. Refer to the NQS chapter in *Administering Platform LSF* for more details.

Default

2147483647 (Not defined.)

NQS_REQUESTS_FLAGS

Syntax

NQS_REQUESTS_FLAGS=*integer*

Description

For Cray NQS compatibility only.

If the NQS version on a Cray is NQS 80.42 or NQS 71.3, this parameter does not need to be defined.

If the version is NQS 1.1 on a Cray, set this parameter to 251918848. This is the is the qstat flag that LSF uses to retrieve requests on Cray in long format.

For other versions of NQS on a Cray, run the NQS qstat command. The value of Npk_int [1] in the output is the value you need for this parameter. Refer to the NQS chapter in *Administering Platform LSF* for more details.

Default

2147483647 (Not defined.)

PARALLEL_SCHED_BY_SLOT

Syntax

PARALLEL_SCHED_BY_SLOT=y | **Y**

Description

If defined, LSF schedules jobs based on the number of slots assigned to the hosts instead of the number of CPUs. These slots can be defined by host in lsb.hosts or by slot limit in lsb.resources.

All slot-related messages still show the word "processors", but actually refer to "slots" instead. Similarly, all scheduling activities also use slots instead of processors.

Default

N (Disabled.)

See also

- JL/U and MXJ in lsb.hosts

• SLOTS and SLOTS_PER_PROCESSOR in `lsb.resources`

PEND_REASON_MAX_JOBS

Syntax

PEND_REASON_MAX_JOBS=*integer*

Description

Number of jobs for each user per queue for which pending reasons are calculated by the scheduling daemon mbschd. Pending reasons are calculated at a time period set by PEND_REASON_UPDATE_INTERVAL.

Default

20 jobs

PEND_REASON_UPDATE_INTERVAL

Syntax

PEND_REASON_UPDATE_INTERVAL=*seconds*

Description

Time interval that defines how often pending reasons are calculated by the scheduling daemon mbschd.

Default

30 seconds

PG_SUSP_IT

Syntax

PG_SUSP_IT=*seconds*

Description

The time interval that a host should be interactively idle (it > 0) before jobs suspended because of a threshold on the pg load index can be resumed.

This parameter is used to prevent the case in which a batch job is suspended and resumed too often as it raises the paging rate while running and lowers it while suspended. If you are not concerned with the interference with interactive jobs caused by paging, the value of this parameter may be set to 0.

Default

180 seconds

PREEMPT_FOR

Syntax

PREEMPT_FOR=[**GROUP_JLP**] [**GROUP_MAX**] [**HOST_JLU**] [**LEAST_RUN_TIME**] [**MINI_JOB**] [**USER_JLP**]

Description

If preemptive scheduling is enabled, this parameter is used to disregard suspended jobs when determining if a job slot limit is exceeded, to preempt jobs with the shortest running time, and to optimize preemption of parallel jobs.

Specify one or more of the following keywords. Use spaces to separate multiple keywords.

GROUP_JLP

Counts only running jobs when evaluating if a user group is approaching its per-processor job slot limit (SLOTS_PER_PROCESSOR, USERS, and PER_HOST=all in the lsb.resources file). Suspended jobs are ignored when this keyword is used.

GROUP_MAX

Counts only running jobs when evaluating if a user group is approaching its total job slot limit (SLOTS, PER_USER=all, and HOSTS in the lsb.resources file). Suspended jobs are ignored when this keyword is used. When preemptive scheduling is enabled, suspended jobs never count against the total job slot limit for individual users.

HOST_JLU

Counts only running jobs when evaluating if a user or user group is approaching its per-host job slot limit (SLOTS and USERS in the lsb.resources file). Suspended jobs are ignored when this keyword is used.

LEAST_RUN_TIME

Preempts the job that has been running for the shortest time. Run time is wall-clock time, not normalized run time.

MINI_JOB

Optimizes the preemption of parallel jobs by preempting only enough parallel jobs to start the high-priority parallel job.

OPTIMAL_MINI_JOB

Optimizes preemption of parallel jobs by preempting only low-priority parallel jobs using the least number of slots to allow the high-priority parallel job to start.

User limits and user group limits can interfere with preemption optimization of OPTIMAL_MINI_JOB. You should not configure OPTIMAL_MINI_JOB if you have user or user group limits configured.

You should configure PARALLEL_SCHED_BY_SLOT=Y when using OPTIMAL_MINI_JOB.

USER_JLP

Counts only running jobs when evaluating if a user is approaching their per-processor job slot limit (SLOTS_PER_PROCESSOR, USERS, and PER_HOST=all in the `lsb.resources` file). Suspended jobs are ignored when this keyword is used. Ignores suspended jobs when calculating the user-processor job slot limit for individual users. When preemptive scheduling is enabled, suspended jobs never count against the total job slot limit for individual users.

Default

0 (The parameter is not defined.)

If preemptive scheduling is enabled, more lower-priority parallel jobs may be preempted than necessary to start a high-priority parallel job. Both running and suspended jobs are counted when calculating the number of job slots in use, except for the following limits:

- The total job slot limit for hosts, specified at the host level
- Total job slot limit for individual users, specified at the user level—by default, suspended jobs still count against the limit for user groups

PREEMPT_JOBTYPE

Syntax

PREEMPT_JOBTYPE=[**EXCLUSIVE**] [**BACKFILL**]

Description

If preemptive scheduling is enabled, this parameter enables preemption of exclusive and backfill jobs.

Specify one or both of the following keywords. Separate keywords with a space.

EXCLUSIVE

Enables preemption of and preemption by exclusive jobs. LSB_DISABLE_LIMLOCK_EXCL=Y in `lsf.conf` must also be defined.

BACKFILL

Enables preemption of backfill jobs. Jobs from higher priority queues can preempt jobs from backfill queues that are either backfilling reserved job slots or running as normal jobs.

Default

Not defined. Exclusive and backfill jobs are only preempted if the exclusive low priority job is running on a different host than the one used by the preemptive high priority job.

PREEMPTABLE_RESOURCES

Syntax

PREEMPTABLE_RESOURCES=*resource_name* ...

Description

Enables license preemption when preemptive scheduling is enabled (has no effect if PREEMPTIVE is not also specified) and specifies the licenses that will be preemption resources. Specify shared numeric resources, static or decreasing, that LSF is configured to release (RELEASE=Y in lsf.shared, which is the default).

You must also configure LSF preemption actions to make the preempted application releases its licenses. To kill preempted jobs instead of suspending them, set TERMINATE_WHEN=PREEMPT in lsb.queues, or set JOB_CONTROLS in lsb.queues and specify brequeue as the SUSPEND action.

Default

Not defined (if preemptive scheduling is configured, LSF preempts on job slots only)

PREEMPTION_WAIT_TIME

Syntax

PREEMPTION_WAIT_TIME=_seconds_

Description

Platform LSF License Scheduler only. You must also specify PREEMPTABLE_RESOURCES in lsb.params).

The amount of time LSF waits, after preempting jobs, for preemption resources to become available. Specify at least 300 seconds.

If LSF does not get the resources after this time, LSF might preempt more jobs.

Default

300 (seconds)

PRIVILEGED_USER_FORCE_BKILL

Syntax

PRIVILEGED_USER_FORCE_BKILL=y | Y

Description

If Y, only root or the LSF administrator can successfully run bkill -r. For any other users, -r is ignored. If not defined, any user can run bkill -r.

Default

Not defined.

REMOTE_MAX_PREEXEC_RETRY

Syntax

REMOTE_MAX_PREEXEC_RETRY=_integer_

Description

The maximum number of times to attempt the pre-execution command of a job on the remote cluster.

Valid values

0 < REMOTE_MAX_PREEXEC_RETRY < INFINIT_INT

INFINIT_INT is defined in lsf.h.

Default

5

RESOURCE_RESERVE_PER_SLOT

Syntax

RESOURCE_RESERVE_PER_SLOT=y | Y

Description

If Y, mbatchd reserves resources based on job slots instead of per-host.

By default, mbatchd only reserves static resources for parallel jobs on a per-host basis. For example, by default, the command:

```
bsub -n 4 -R "rusage[mem=500]" -q reservation my_job
```

requires the job to reserve 500 MB on each host where the job runs.

Some parallel jobs need to reserve resources based on job slots, rather than by host. In this example, if per-slot reservation is enabled by RESOURCE_RESERVE_PER_SLOT, the job my_job must reserve 500 MB of memory for each job slot (4*500=2 GB) on the host in order to run.

If RESOURCE_RESERVE_PER_SLOT is set, the following command reserves the resource static_resource on all 4 job slots instead of only 1 on the host where the job runs:

```
bsub -n 4 -R "static_resource > 0 rusage[static_resource=1]" myjob
```

Default

N (Not defined; reserve resources per-host.)

RUN_JOB_FACTOR

Syntax

RUN_JOB_FACTOR=_number_

Description

Used only with fairshare scheduling. Job slots weighting factor.

In the calculation of a user's dynamic share priority, this factor determines the relative importance of the number of job slots reserved and in use by a user.

Default

3.0

RUN_TIME_FACTOR

Syntax

RUN_TIME_FACTOR=*number*

Description

Used only with fairshare scheduling. Run time weighting factor.

In the calculation of a user's dynamic share priority, this factor determines the relative importance of the total run time of a user's running jobs.

Default

0.7

SBD_SLEEP_TIME

Syntax

SBD_SLEEP_TIME=*seconds*

Description

The interval at which LSF checks the load conditions of each host, to decide whether jobs on the host must be suspended or resumed.

The job-level resource usage information is updated at a maximum frequency of every SBD_SLEEP_TIME seconds.

The update is done only if the value for the CPU time, resident memory usage, or virtual memory usage has changed by more than 10 percent from the previous update or if a new process or process group has been created.

Default

SBD_SLEEP_TIME is set at installation to 15 seconds. If not defined, 30 seconds.

SCHED_METRIC_ENABLE

Syntax

SCHED_METRIC_ENABLE=Y | N

Description

Enable scheduler performance metric collection.

Use badmin perfmon stop and badmin perfmon start to dynamically control performance metric collection.

The update is done only if the value for the CPU time, resident memory usage, or virtual memory usage has changed by more than 10 percent from the previous update or if a new process or process group has been created.

Default

N

SCHED_METRIC_SAMPLE_PERIOD

Syntax

SCHED_METRIC_SAMPLE_PERIOD=*seconds*

Description

Set a default performance metric sampling period in seconds.

Cannot be less than 60 seconds.

Use badmin perfmon setperiod to dynamically change performance metric sampling period.

Default

60 seconds

SLA_TIMER

Syntax

SLA_TIMER=*seconds*

Description

For EGO-enabled SLA scheduling. Controls how often each service class is evaluated and a network message is sent to EGO communicating host demand.

Valid values

Positive integer between 2 and 21474847

Default

0 (Not defined.)

SUB_TRY_INTERVAL

Syntax

SUB_TRY_INTERVAL=*integer*

Description

The number of seconds for the requesting client to wait before resubmitting a job. This is sent by mbatchd to the client.

Default

60 seconds

See also

MAX_PEND_JOBS

SYSTEM_MAPPING_ACCOUNT

Syntax

SYSTEM_MAPPING_ACCOUNT=*user_account*

Description

Enables Windows workgroup account mapping, which allows LSF administrators to map all Windows workgroup users to a single Windows system account, eliminating the need to create multiple users and passwords in LSF. Users can submit and run jobs using their local user names and passwords, and LSF runs the jobs using the mapped system account name and password. With Windows workgroup account mapping, all users have the same permissions because all users map to the same system account.

To specify the user account, include the domain name in uppercase letters (*DOMAIN_NAME \user_name*).

Define this parameter for LSF Windows Workgroup installations only.

Default

Not defined

USE_SUSP_SLOTS

Syntax

USE_SUSP_SLOTS=Y | N

Description

If USE_SUSP_SLOTS=Y, allows jobs from a low priority queue to use slots held by suspended jobs in a high priority queue, which has a preemption relation with the low priority queue.

Set USE_SUSP_SLOTS=N to prevent low priority jobs from using slots held by suspended jobs in a high priority queue, which has a preemption relation with the low priority queue.

Default

Y

Automatic time-based configuration

Variable configuration is used to automatically change LSF configuration based on time windows. You define automatic configuration changes in lsb. params by using if-else constructs and time expressions. After you change the files, reconfigure the cluster with the badmin reconfig command.

The expressions are evaluated by LSF every 10 minutes based on mbatchd start time. When an expression evaluates true, LSF dynamically changes the configuration based on the associated configuration statements. Reconfiguration is done in real time without restarting mbatchd, providing continuous system availability.

Example

```
# if 18:30-19:30 is your short job express period, but
# you want all jobs going to the short queue by default
# and be subject to the thresholds of that queue

# for all other hours, normal is the default queue

#if time(18:30-19:30)
DEFAULT_QUEUE=short
#else
DEFAULT_QUEUE=normal
#endif
```

lsb.queues

The lsb.queues file defines batch queues. Numerous controls are available at the queue level to allow cluster administrators to customize site policies.

This file is optional; if no queues are configured, LSF creates a queue named default, with all parameters set to default values.

This file is installed by default in LSB_CONFDIR/*cluster_name*/configdir.

Changing lsb.queues configuration

After making any changes to lsb.queues, run badmin reconfig to reconfigure mbatchd.

Some parameters such as run window and run time limit do not take effect immediately for running jobs unless you run mbatchd restart or sbatchd restart on the job execution host.

lsb.queues structure

Each queue definition begins with the line Begin Queue and ends with the line End Queue. The queue name must be specified; all other parameters are optional.

Parameters

- ADMINISTRATORS
- APS_PRIORITY
- BACKFILL
- CHKPNT
- CHUNK_JOB_SIZE
- CORELIMIT
- CPULIMIT
- DATALIMIT
- DEFAULT_EXTSCHED
- DEFAULT_HOST_SPEC
- DESCRIPTION
- DISPATCH_ORDER
- DISPATCH_WINDOW
- EXCLUSIVE
- FAIRSHARE
- FAIRSHARE_QUEUES
- FILELIMIT
- HJOB_LIMIT
- HOSTS
- IGNORE_DEADLINE
- IMPT_JOBBKLG
- INTERACTIVE
- INTERRUPTIBLE_BACKFILL
- JOB_ACCEPT_INTERVAL
- JOB_ACTION_WARNING_TIME
- JOB_CONTROLS

- JOB_IDLE
- JOB_OVERRUN
- JOB_STARTER
- JOB_UNDERRUN
- JOB_WARNING_ACTION
- load_index
- LOCAL_MAX_PREEXEC_RETRY
- MANDATORY_EXTSCHED
- MAX_JOB_PREEMPT
- MAX_JOB_REQUEUE
- MAX_PREEXEC_RETRY
- MAX_RSCHED_TIME
- MEMLIMIT
- MIG
- NEW_JOB_SCHED_DELAY
- NICE
- NQS_QUEUES
- PJOB_LIMIT
- POST_EXEC
- PRE_EXEC
- PREEMPTION
- PRIORITY
- PROCESSLIMIT
- PROCLIMIT
- QJOB_LIMIT
- QUEUE_GROUP
- QUEUE_NAME
- RCVJOBS_FROM
- REMOTE_MAX_PREEXEC_RETRY
- REQUEUE_EXIT_VALUES
- RERUNNABLE
- RESOURCE_RESERVE
- RES_REQ
- RESUME_COND
- RUN_WINDOW
- RUNLIMIT
- SLOT_POOL
- SLOT_RESERVE
- SLOT_SHARE
- SNDJOBS_TO
- STACKLIMIT
- STOP_COND
- SWAPLIMIT
- THREADLIMIT
- UJOB_LIMIT
- USE_PAM_CREDS

- USERS

ADMINISTRATORS

Syntax

ADMINISTRATORS=*user_name* | *user_group* ...

Description

List of queue administrators. To specify a Windows user account or user group, include the domain name in uppercase letters (*DOMAIN_NAME\user_name* or *DOMAIN_NAME \user_group*).

Queue administrators can perform operations on any user's job in the queue, as well as on the queue itself.

Default

Not defined. You must be a cluster administrator to operate on this queue.

APS_PRIORITY

Syntax

APS_PRIORITY=WEIGHT[[*factor, value*] [*subfactor, value*]...]...**] LIMIT[**[*factor, value*] [*subfactor, value*]...]...**] GRACE_PERIOD[**[*factor, value*] [*subfactor, value*]...]...**]**

Description

Specifies calculation factors for absolute priority scheduling (APS). Pending jobs in the queue are ordered according to the calculated APS value.

If weight of a subfactor is defined, but the weight of parent factor is not defined, the parent factor weight is set as 1.

The WEIGHT and LIMIT factors are floating-point values. Specify a *value* for GRACE_PERIOD in seconds (*value*s), minutes (*value*m), or hours (*value*h).

The default unit for grace period is hours.

For example, the following sets a grace period of 10 hours for the MEM factor, 10 minutes for the JPRIORITY factor, 10 seconds for the QPRIORITY factor, and 10 hours (default) for the RSRC factor:

```
GRACE_PERIOD[[MEM, 10h] [JPRIORITY,  10m] [QPRIORITY, 10s] [RSRC,  10]]
```

You cannot specify zero (0) for the WEIGHT, LIMIT, and GRACE_PERIOD of any factor or subfactor.

APS queues cannot configure cross-queue fairshare (FAIRSHARE_QUEUES). The QUEUE_GROUP parameter replaces FAIRSHARE_QUEUES, which is obsolete in LSF 7.0.

Suspended (bstop) jobs and migrated jobs (bmig) are always scheduled before pending jobs. For migrated jobs, LSF keeps the existing job priority information.

If LSB_REQUEUE_TO_BOTTOM and LSB_MIG2PEND are configured in lsf.conf, the migrated jobs keep their APS information. When LSB_REQUEUE_TO_BOTTOM and LSB_MIG2PEND are configured, the migrated jobs need to compete with other pending jobs

based on the APS value. If you want to reset the APS value, the you should use `brequeue`, not `bmig`.

Default

Not defined

BACKFILL

Syntax

BACKFILL=Y | **N**

Description

If Y, enables backfill scheduling for the queue.

A possible conflict exists if BACKFILL and PREEMPTION are specified together. If PREEMPT_JOBTYPE = BACKFILL is set in the lsb.params file, a backfill queue can be preemptable. Otherwise a backfill queue cannot be preemptable. If BACKFILL is enabled do not also specify PREEMPTION = PREEMPTABLE.

BACKFILL is required for interruptible backfill queues (INTERRUPTIBLE_BACKFILL=*seconds*).

Default

Not defined. No backfilling.

CHKPNT

Syntax

CHKPNT=*chkpnt_dir* [*chkpnt_period*]

Description

Enables automatic checkpointing for the queue. All jobs submitted to the queue are checkpointable.

The checkpoint directory is the directory where the checkpoint files are created. Specify an absolute path or a path relative to CWD, do not use environment variables.

Specify the optional checkpoint period in minutes.

Only running members of a chunk job can be checkpointed.

If checkpoint-related configuration is specified in both the queue and an application profile, the application profile setting overrides queue level configuration.

If checkpoint-related configuration is specified in the queue, application profile, and at job level:

- Application-level and job-level parameters are merged. If the same parameter is defined at both job-level and in the application profile, the job-level value overrides the application profile value.
- The merged result of job-level and application profile settings override queue-level configuration.

To enable checkpointing of MultiCluster jobs, define a checkpoint directory in both the send-jobs and receive-jobs queues (CHKPNT in lsb. queues), or in an application profile (CHKPNT_DIR, CHKPNT_PERIOD, CHKPNT_INITPERIOD, CHKPNT_METHOD in lsb. applications) of both submission cluster and execution cluster. LSF uses the directory specified in the execution cluster.

To make a MultiCluster job checkpointable, both submission and execution queues must enable checkpointing, and the application profile or queue setting on the execution cluster determines the checkpoint directory. Checkpointing is not supported if a job runs on a leased host.

The file path of the checkpoint directory can contain up to 4000 characters for UNIX and Linux, or up to 255 characters for Windows, including the directory and file name.

Default

Not defined

CHUNK_JOB_SIZE

Syntax

CHUNK_JOB_SIZE=_integer_

Description

Chunk jobs only. Enables job chunking and specifies the maximum number of jobs allowed to be dispatched together in a chunk. Specify a positive integer greater than 1.

The ideal candidates for job chunking are jobs that have the same host and resource requirements and typically take 1 to 2 minutes to run.

Job chunking can have the following advantages:

- Reduces communication between sbatchd and mbatchd and reduces scheduling overhead in mbschd.
- Increases job throughput in mbatchd and CPU utilization on the execution hosts.

However, throughput can deteriorate if the chunk job size is too big. Performance may decrease on queues with CHUNK_JOB_SIZE greater than 30. You should evaluate the chunk job size on your own systems for best performance.

With MultiCluster job forwarding model, this parameter does not affect MultiCluster jobs that are forwarded to a remote cluster.

Compatibility

This parameter is ignored in the following kinds of queues and applications:

- Interactive (INTERACTIVE=ONLY parameter)
- CPU limit greater than 30 minutes (CPULIMIT parameter)
- Run limit greater than 30 minutes (RUNLIMIT parameter)
- Runtime estimate greater than 30 minutes (RUNTIME parameter in lsb. applications only)

If CHUNK_JOB_DURATION is set in lsb. params, chunk jobs are accepted regardless of the value of CPULIMIT, RUNLIMIT or RUNTIME.

Example

The following configures a queue named chunk, which dispatches up to 4 jobs in a chunk:

```
Begin Queue
QUEUE_NAME      = chunk
PRIORITY        = 50
CHUNK_JOB_SIZE = 4
End Queue
```

Default

Not defined

CORELIMIT

Syntax

CORELIMIT=*integer*

Description

The per-process (hard) core file size limit (in KB) for all of the processes belonging to a job from this queue (see getrlimit(2)).

Default

Unlimited

CPULIMIT

Syntax

CPULIMIT=[*default_limit***]** *maximum_limit*

where *default_limit* and *maximum_limit* are:

[*hour*:]*minute*[/*host_name* | /*host_model*]

Description

Maximum normalized CPU time and optionally, the default normalized CPU time allowed for all processes of a job running in this queue. The name of a host or host model specifies the CPU time normalization host to use.

Limits the total CPU time the job can use. This parameter is useful for preventing runaway jobs or jobs that use up too many resources.

When the total CPU time for the whole job has reached the limit, a SIGXCPU signal is sent to all processes belonging to the job. If the job has no signal handler for SIGXCPU, the job is killed immediately. If the SIGXCPU signal is handled, blocked, or ignored by the application, then after the grace period expires, LSF sends SIGINT, SIGTERM, and SIGKILL to the job to kill it.

If a job dynamically spawns processes, the CPU time used by these processes is accumulated over the life of the job.

Processes that exist for fewer than 30 seconds may be ignored.

By default, if a default CPU limit is specified, jobs submitted to the queue without a job-level CPU limit are killed when the default CPU limit is reached.

If you specify only one limit, it is the maximum, or hard, CPU limit. If you specify two limits, the first one is the default, or soft, CPU limit, and the second one is the maximum CPU limit. The number of minutes may be greater than 59. Therefore, three and a half hours can be specified either as 3:30 or 210.

If no host or host model is given with the CPU time, LSF uses the default CPU time normalization host defined at the queue level (DEFAULT_HOST_SPEC in lsb. queues) if it has been configured, otherwise uses the default CPU time normalization host defined at the cluster level (DEFAULT_HOST_SPEC in lsb. params) if it has been configured, otherwise uses the host with the largest CPU factor (the fastest host in the cluster).

On Windows, a job that runs under a CPU time limit may exceed that limit by up to SBD_SLEEP_TIME. This is because sbatchd periodically checks if the limit has been exceeded.

On UNIX systems, the CPU limit can be enforced by the operating system at the process level.

You can define whether the CPU limit is a per-process limit enforced by the OS or a per-job limit enforced by LSF with LSB_JOB_CPULIMIT in lsf. conf.

Jobs submitted to a chunk job queue are not chunked if CPULIMIT is greater than 30 minutes.

Default

Unlimited

DATALIMIT

Syntax

DATALIMIT=[*default_limit*] *maximum_limit*

Description

The per-process data segment size limit (in KB) for all of the processes belonging to a job from this queue (see getrlimit(2)).

By default, if a default data limit is specified, jobs submitted to the queue without a job-level data limit are killed when the default data limit is reached.

If you specify only one limit, it is the maximum, or hard, data limit. If you specify two limits, the first one is the default, or soft, data limit, and the second one is the maximum data limit

Default

Unlimited

DEFAULT_EXTSCHED

Syntax

DEFAULT_EXTSCHED=*external_scheduler_options*

Description

Specifies default external scheduling options for the queue.

-extsched options on the bsub command are merged with DEFAULT_EXTSCHED options, and -extsched options override any conflicting queue-level options set by DEFAULT_EXTSCHED.

Default

Not defined

DEFAULT_HOST_SPEC

Syntax

DEFAULT_HOST_SPEC=*host_name* | *host_model*

Description

The default CPU time normalization host for the queue.

The CPU factor of the specified host or host model is used to normalize the CPU time limit of all jobs in the queue, unless the CPU time normalization host is specified at the job level.

Default

Not defined. The queue uses the DEFAULT_HOST_SPEC defined in lsb.params. If DEFAULT_HOST_SPEC is not defined in either file, LSF uses the fastest host in the cluster.

DESCRIPTION

Syntax

DESCRIPTION=*text*

Description

Description of the job queue displayed by bqueues -l.

This description should clearly describe the service features of this queue, to help users select the proper queue for each job.

The text can include any characters, including white space. The text can be extended to multiple lines by ending the preceding line with a backslash (\). The maximum length for the text is 512 characters.

DISPATCH_ORDER

Syntax

DISPATCH_ORDER=QUEUE

Description

Defines an *ordered* cross-queue fairshare set. DISPATCH_ORDER indicates that jobs are dispatched according to the order of queue priorities first, then user fairshare priority.

By default, a user has the same priority across the master and slave queues. If the same user submits several jobs to these queues, user priority is calculated by taking into account all the jobs the user has submitted across the master-slave set.

If DISPATCH_ORDER=QUEUE is set in the master queue, jobs are dispatched according to queue priorities first, then user priority. Jobs from users with lower fairshare priorities who have pending jobs in higher priority queues are dispatched before jobs in lower priority queues. This avoids having users with higher fairshare priority getting jobs dispatched from low-priority queues.

Jobs in queues having the same priority are dispatched according to user priority.

Queues that are not part of the cross-queue fairshare can have any priority; they are not limited to fall outside of the priority range of cross-queue fairshare queues.

Default

Not defined

DISPATCH_WINDOW

Syntax

DISPATCH_WINDOW=*time_window* ...

Description

The time windows in which jobs from this queue are dispatched. Once dispatched, jobs are no longer affected by the dispatch window.

Default

Not defined. Dispatch window is always open.

EXCLUSIVE

Syntax

EXCLUSIVE=Y | **N** | **CU**[*cu_type*]

Description

If Y, specifies an exclusive queue.

If CU, CU[], or CU[*cu_type*], specifies an exclusive queue as well as a queue exclusive to compute units of type *cu_type* (as defined in lsb.params). If no type is specified, the default compute unit type is used.

Jobs submitted to an exclusive queue with bsub -x are only dispatched to a host that has no other LSF jobs running. Jobs submitted to a compute unit exclusive queue with bsub -R "cu [excl]" only run on a compute unit that has no other jobs running.

For hosts shared under the MultiCluster resource leasing model, jobs are not dispatched to a host that has LSF jobs running, even if the jobs are from another cluster.

Default

N

FAIRSHARE

Syntax

FAIRSHARE=USER_SHARES[[*user*, *number_shares*] ...]

- Specify at least one user share assignment.
- Enclose the list in square brackets, as shown.
- Enclose each user share assignment in square brackets, as shown.
- *user:* Specify users who are also configured to use queue. You can assign the shares to:

  - A single user (specify *user_name*). To specify a Windows user account, include the domain name in uppercase letters (*DOMAIN_NAME\user_name*).
  - Users in a group, individually (specify *group_name@*) or collectively (specify *group_name*). To specify a Windows user group, include the domain name in uppercase letters (*DOMAIN_NAME\group_name).*
  - Users not included in any other share assignment, individually (specify the keyword default) or collectively (specify the keyword others)

    - By default, when resources are assigned collectively to a group, the group members compete for the resources on a first-come, first-served (FCFS) basis. You can use hierarchical fairshare to further divide the shares among the group members.
    - When resources are assigned to members of a group individually, the share assignment is recursive. Members of the group and of all subgroups always compete for the resources according to FCFS scheduling, regardless of hierarchical fairshare policies.
- *number_shares*

  - Specify a positive integer representing the number of shares of the cluster resources assigned to the user.
  - The number of shares assigned to each user is only meaningful when you compare it to the shares assigned to other users or to the total number of shares. The total number of shares is just the sum of all the shares assigned in each share assignment.

Description

Enables queue-level user-based fairshare and specifies share assignments. Only users with share assignments can submit jobs to the queue.

Compatibility

Do not configure hosts in a cluster to use fairshare at both queue and host levels. However, you can configure user-based fairshare and queue-based fairshare together.

Default

Not defined. No fairshare.

FAIRSHARE_QUEUES

Syntax

FAIRSHARE_QUEUES=*queue_name*[*queue_name* ...]

Description

Defines cross-queue fairshare. When this parameter is defined:

- The queue in which this parameter is defined becomes the "*master queue*".
- Queues listed with this parameter are "*slave queues*" and inherit the fairshare policy of the master queue.
- A user has the same priority across the master and slave queues. If the same user submits several jobs to these queues, user priority is calculated by taking into account all the jobs the user has submitted across the master-slave set.

Notes

- By default, the PRIORITY range defined for queues in cross-queue fairshare cannot be used with any other queues. For example, you have 4 queues: queue1, queue2, queue3, queue4. You configure cross-queue fairshare for queue1, queue2, queue3 and assign priorities of 30, 40, 50 respectively.
- By default, the priority of queue4 (which is not part of the cross-queue fairshare) cannot fall between the priority range of the cross-queue fairshare queues (30-50). It can be any number up to 29 or higher than 50. It does not matter if queue4 is a fairshare queue or FCFS queue. If DISPATCH_ORDER=QUEUE is set in the master queue, the priority of queue4 (which is not part of the cross-queue fairshare) can be any number, including a priority falling between the priority range of the cross-queue fairshare queues (30-50).
- FAIRSHARE must be defined in the master queue. If it is also defined in the queues listed in FAIRSHARE_QUEUES, it is ignored.
- Cross-queue fairshare can be defined more than once within lsb.queues. You can define several sets of master-slave queues. However, a queue cannot belong to more than one master-slave set. For example, you can define:

  - In queue normal : FAIRSHARE_QUEUES=short license
  - In queue priority: FAIRSHARE_QUEUES=night owners

    Restriction:

    You cannot, however, define night, owners, or priority as slaves in the queue normal ; or normal, short and license as slaves in the priority queue; or short, license, night, owners as master queues of their own.

- Cross-queue fairshare cannot be used with host partition fairshare . It is part of queue-level fairshare.
- Cross-queue fairshare cannot be used with absolute priority scheduling.

Default

Not defined

FILELIMIT

Syntax

FILELIMIT=_integer_

Description

The per-process (hard) file size limit (in KB) for all of the processes belonging to a job from this queue (see `getrlimit(2)`).

Default

Unlimited

HJOB_LIMIT

Syntax

HJOB_LIMIT=_integer_

Description

Per-host job slot limit.

Maximum number of job slots that this queue can use on any host. This limit is configured per host, regardless of the number of processors it may have.

This may be useful if the queue dispatches jobs that require a node-locked license. If there is only one node-locked license per host then the system should not dispatch more than one job to the host even if it is a multiprocessor host.

Example

The following runs a maximum of one job on each of hostA, hostB, and hostC:

```
Begin Queue
...
HJOB_LIMIT = 1
HOSTS=hostA hostB hostC
...
End Queue
```

Default

Unlimited

HOSTS

Syntax

HOSTS=_host_list_ | **none**

- _host_list_ is a space-separated list of the following items:
    - _host_name_[@_cluster_name_][[!] | +_pref_level_]
    - _host_partition_[+_pref_level_]
    - _host_group_[[!] | +_pref_level_]
    - _compute_unit_[[!] | +_pref_level_]
    - [~]_host_name_
    - [~]_host_group_
    - [~]_compute_unit_

- The list can include the following items only once:
  - all@*cluster_name*
  - others[+*pref_level*]
  - all
  - allremote
- The none keyword is only used with the MultiCluster job forwarding model, to specify a remote-only queue.

Description

A space-separated list of hosts on which jobs from this queue can be run.

If compute units, host groups, or host partitions are included in the list, the job can run on any host in the unit, group, or partition. All the members of the host list should either belong to a single host partition or not belong to any host partition. Otherwise, job scheduling may be affected.

Some items can be followed by a plus sign (+) and a positive number to indicate the preference for dispatching a job to that host. A higher number indicates a higher preference. If a host preference is not given, it is assumed to be 0. If there are multiple candidate hosts, LSF dispatches the job to the host with the highest preference; hosts at the same level of preference are ordered by load.

If compute units, host groups, or host partitions are assigned a preference, each host in the unit, group, or partition has the same preference.

Use the keyword others to include all hosts not explicitly listed.

Use the keyword all to include all hosts not explicitly excluded.

Use the keyword all@*cluster_name hostgroup_name* or allremote *hostgroup_name* to include lease in hosts.

Use the not operator (~) to exclude hosts from the all specification in the queue. This is useful if you have a large cluster but only want to exclude a few hosts from the queue definition.

The not operator can only be used with the all keyword. It is *not* valid with the keywords others and none.

The not operator (~) can be used to exclude host groups.

For parallel jobs, specify first execution host candidates when you want to ensure that a host has the required resources or runtime environment to handle processes that run on the first execution host.

To specify one or more hosts, host groups, or compute units as first execution host candidates, add the exclamation point (!) symbol after the name.

Follow these guidelines when you specify first execution host candidates:

- If you specify a compute unit or host group, you must first define the unit or group in the file lsb.hosts.
- Do not specify a dynamic host group as a first execution host.
- Do not specify "all," "allremote," or "others," or a host partition as a first execution host.
- Do not specify a preference (+) for a host identified by (!) as a first execution host candidate.
- For each parallel job, specify enough regular hosts to satisfy the CPU requirement for the job. Once LSF selects a first execution host for the current job, the other first execution host candidates

- • Become unavailable to the current job
  - • Remain available to other jobs as either regular or first execution hosts
- • You cannot specify first execution host candidates when you use the brun command.

Restriction:

If you have enabled EGO, host groups and compute units are not
honored.

With MultiCluster resource leasing model, use the format *host_name@cluster_name* to specify
a borrowed host. LSF does not validate the names of remote hosts. The keyword others
indicates all local hosts not explicitly listed. The keyword all indicates all local hosts not
explicitly excluded. Use the keyword allremote to specify all hosts borrowed from all remote
clusters. Use all@*cluster_name* to specify the group of all hosts borrowed from one remote
cluster. You cannot specify a host group or partition that includes remote resources, unless it
uses the keyword allremote to include all remote hosts. You cannot specify a compute unit
that includes remote resources.

With MultiCluster resource leasing model, the not operator (~) can be used to exclude local
hosts or host groups. You cannot use the not operator (~) with remote hosts.

Restriction:

Hosts that participate in queue-based fairshare cannot be in a
host partition.

Behavior with host intersection

Host preferences specified by bsub -m combine intelligently with the queue specification and
advance reservation hosts. The jobs run on the hosts that are both specified at job submission
and belong to the queue or have advance reservation.

Example 1

```
HOSTS=hostA+1 hostB hostC+1 hostD+3
```

This example defines three levels of preferences: run jobs on hostD as much as possible,
otherwise run on either hostA or hostC if possible, otherwise run on hostB. Jobs should not
run on hostB unless all other hosts are too busy to accept more jobs.

Example 2

```
HOSTS=hostD+1 others
```

Run jobs on hostD as much as possible, otherwise run jobs on the least-loaded host available.

With MultiCluster resource leasing model, this queue does not use borrowed hosts.

Example 3

```
HOSTS=all ~hostA
```

Run jobs on all hosts in the cluster, except for hostA.

With MultiCluster resource leasing model, this queue does not use borrowed hosts.

Example 4

```
HOSTS=Group1 ~hostA hostB hostC
```

Run jobs on hostB, hostC, and all hosts in Group1 except for hostA.

With MultiCluster resource leasing model, this queue uses borrowed hosts if Group1 uses the keyword allremote.

Example 5

```
HOSTS=hostA! hostB+ hostC hostgroup1!
```

Runs parallel jobs using either hostA or a host defined in hostgroup1 as the first execution host. If the first execution host cannot run the entire job due to resource requirements, runs the rest of the job on hostB. If hostB is too busy to accept the job, or if hostB does not have enough resources to run the entire job, runs the rest of the job on hostC.

Example 6

```
HOSTS=computeunit1! hostB hostC
```

Runs parallel jobs using a host in computeunit1 as the first execution host. If the first execution host cannot run the entire job due to resource requirements, runs the rest of the job on other hosts in computeunit1 followed by hostB and finally hostC.

Example 7

```
HOSTS=hostgroup1! computeunitA computeunitB computeunitC
```

Runs parallel jobs using a host in hostgroup1 as the first execution host. If additional hosts are required, runs the rest of the job on other hosts in the same compute unit as the first execution host, followed by hosts in the remaining compute units in the order they are defined in the lsb.hosts ComputeUnit section.

Default

all (the queue can use all hosts in the cluster, and every host has equal preference)

With MultiCluster resource leasing model, this queue can use all local hosts, but no borrowed hosts.

IGNORE_DEADLINE

Syntax

IGNORE_DEADLINE=Y

Description

If Y, disables deadline constraint scheduling (starts all jobs regardless of deadline constraints).

IMPT_JOBBKLG

Syntax

IMPT_JOBBKLG=_integer_ |**infinit**

Description

MultiCluster job forwarding model only. Specifies the MultiCluster pending job limit for a receive-jobs queue. This represents the maximum number of MultiCluster jobs that can be pending in the queue; once the limit has been reached, the queue stops accepting jobs from remote clusters.

Use the keyword infinit to make the queue accept an unlimited number of pending MultiCluster jobs.

Default

50

INTERACTIVE

Syntax

INTERACTIVE=YES | **NO** | **ONLY**

Description

YES causes the queue to accept both interactive and non-interactive batch jobs, NO causes the queue to reject interactive batch jobs, and ONLY causes the queue to accept interactive batch jobs and reject non-interactive batch jobs.

Interactive batch jobs are submitted via bsub -I .

Default

YES. The queue accepts both interactive and non-interactive jobs.

INTERRUPTIBLE_BACKFILL

Syntax

INTERRUPTIBLE_BACKFILL=*seconds*

Description

Configures interruptible backfill scheduling policy, which allows reserved job slots to be used by low priority small jobs that are terminated when the higher priority large jobs are about to start.

There can only be one interruptible backfill queue.It should be the lowest priority queue in the cluster.

Specify the minimum number of seconds for the job to be considered for backfilling.This minimal time slice depends on the specific job properties; it must be longer than at least one useful iteration of the job. Multiple queues may be created if a site has jobs of distinctively different classes.

An interruptible backfill job:

- Starts as a regular job and is killed when it exceeds the queue runtime limit, or
- Is started for backfill whenever there is a backfill time slice longer than the specified minimal time, and killed before the slot-reservation job is about to start

The queue RUNLIMIT corresponds to a maximum time slice for backfill, and should be configured so that the wait period for the new jobs submitted to the queue is acceptable to users. 10 minutes of runtime is a common value.

You should configure REQUEUE_EXIT_VALUES for interruptible backfill queues.

BACKFILL and RUNLIMIT must be configured in the queue. The queue is disabled if BACKFILL and RUNLIMIT are not configured.

Assumptions and limitations:

- The interruptible backfill job holds the slot-reserving job start until its calculated start time, in the same way as a regular backfill job. The interruptible backfill job are not preempted in any way other than being killed when its time come.
- While the queue is checked for the consistency of interruptible backfill, backfill and runtime specifications, the requeue exit value clause is not verified, nor executed automatically. Configure requeue exit values according to your site policies.
- The interruptible backfill job must be able to do at least one unit of useful calculations and save its data within the minimal time slice, and be able to continue its calculations after it has been restarted
- Interruptible backfill paradigm does not explicitly prohibit running parallel jobs, distributed across multiple nodes; however, the chance of success of such job is close to zero.

Default

Not defined. No interruptible backfilling.

JOB_ACCEPT_INTERVAL

Syntax

JOB_ACCEPT_INTERVAL=*integer*

Description

The number you specify is multiplied by the value of l sb. params MBD_SLEEP_TIME (60 seconds by default). The result of the calculation is the number of seconds to wait after dispatching a job to a host, before dispatching a second job to the same host.

If 0 (zero), a host may accept more than one job in each dispatch turn. By default, there is no limit to the total number of jobs that can run on a host, so if this parameter is set to 0, a very large number of jobs might be dispatched to a host all at once. This can overload your system to the point that it is unable to create any more processes. It is not recommended to set this parameter to 0.

JOB_ACCEPT_INTERVAL set at the queue level (l sb. queues) overrides JOB_ACCEPT_INTERVAL set at the cluster level (l sb. params).

Default

Not defined. The queue uses JOB_ACCEPT_INTERVAL defined in l sb. params, which has a default value of 1.

JOB_ACTION_WARNING_TIME

Syntax

JOB_ACTION_WARNING_TIME=[*hour*:]*minute*

Description

Specifies the amount of time before a job control action occurs that a job warning action is to be taken. For example, 2 minutes before the job reaches runtime limit or termination deadline, or the queue's run window is closed, an URG signal is sent to the job.

Job action warning time is not normalized.

A job action warning time must be specified with a job warning action in order for job warning to take effect.

The warning time specified by the `bsub -wt` option overrides JOB_ACTION_WARNING_TIME in the queue. JOB_ACTION_WARNING_TIME is used as the default when no command line option is specified.

Example

JOB_ACTION_WARNING_TIME=2

Default

Not defined

JOB_CONTROLS

Syntax

JOB_CONTROLS=SUSPEND[*signal* | *command* | **CHKPNT] RESUME[***signal* | *command***]** **TERMINATE[***signal* | *command* | **CHKPNT]**

- *signal* is a UNIX signal name (for example, SIGTSTP or SIGTERM). The specified signal is sent to the job. The same set of signals is not supported on all UNIX systems. To display a list of the symbolic names of the signals (without the SIG prefix) supported on your system, use the `kill -l` command.
- *command* specifies a `/bin/sh` command line to be invoked.

Restriction:

Do not quote the command line inside an action definition. Do not specify a signal followed by an action that triggers the same signal. For example, do not specify

JOB_CONTROLS=TERMINATE[bkill] or

JOB_CONTROLS=TERMINATE[brequeue]. This causes a deadlock between the signal and the action.

- CHKPNT is a special action, which causes the system to checkpoint the job. Only valid for SUSPEND and TERMINATE actions:
  - If the SUSPEND action is CHKPNT, the job is checkpointed and then stopped by sending the SIGSTOP signal to the job automatically.

- If the TERMINATE action is CHKPNT, then the job is checkpointed and killed automatically.

Description

Changes the behavior of the SUSPEND, RESUME, and TERMINATE actions in LSF.

- The contents of the configuration line for the action are run with `/bin/sh -c` so you can use shell features in the command.
- The standard input, output, and error of the command are redirected to the NULL device, so you cannot tell directly whether the command runs correctly. The default null device on UNIX is `/dev/null`.
- The command is run as the user of the job.
- All environment variables set for the job are also set for the command action. The following additional environment variables are set:

  - LSB_JOBPGIDS: a list of current process group IDs of the job
  - LSB_JOBPIDS: a list of current process IDs of the job

- For the SUSPEND action command, the following environment variables are also set:

  - LSB_SUSP_REASONS : an integer representing a bitmap of suspending reasons as defined in `lsbatch.h`. The suspending reason can allow the command to take different actions based on the reason for suspending the job.
  - LSB_SUSP_SUBREASONS : an integer representing the load index that caused the job to be suspended. When the suspending reason SUSP_LOAD_REASON (suspended by load) is set in LSB_SUSP_REASONS, LSB_SUSP_SUBREASONS set to one of the load index values defined in `lsf.h`. Use LSB_SUSP_REASONS and LSB_SUSP_SUBREASONS together in your custom job control to determine the exact load threshold that caused a job to be suspended.

- If an additional action is necessary for the SUSPEND command, that action should also send the appropriate signal to the application. Otherwise, a job can continue to run even after being suspended by LSF. For example, `JOB_CONTROLS=SUSPEND[kill $LSB_JOBPIDS;` *command*`]`
- If you set preemption with the signal SIGTSTP you use Platform License Scheduler, define LIC_SCHED_PREEMPT_STOP=Y in `lsf.conf` for License Scheduler preemption to work.

Default

On UNIX, by default, SUSPEND sends SIGTSTP for parallel or interactive jobs and SIGSTOP for other jobs. RESUME sends SIGCONT. TERMINATE sends SIGINT, SIGTERM and SIGKILL in that order.

On Windows, actions equivalent to the UNIX signals have been implemented to do the default job control actions. Job control messages replace the SIGINT and SIGTERM signals, but only customized applications are able to process them. Termination is implemented by the `TerminateProcess()` system call.

JOB_IDLE

Syntax

JOB_IDLE=_number_

Description

Specifies a threshold for idle job exception handling. The value should be a number between 0.0 and 1.0 representing CPU time/runtime. If the job idle factor is less than the specified threshold, LSF invokes LSF_SERVERDIR/eadmin to trigger the action for a job idle exception.

The minimum job run time before mbatchd reports that the job is idle is defined as DETECT_IDLE_JOB_AFTER in lsb.params.

Valid Values

Any positive number between 0.0 and 1.0

Example

```
JOB_IDLE=0.10
```

A job idle exception is triggered for jobs with an idle value (CPU time/runtime) less than 0.10.

Default

Not defined. No job idle exceptions are detected.

JOB_OVERRUN

Syntax

JOB_OVERRUN=*run_time*

Description

Specifies a threshold for job overrun exception handling. If a job runs longer than the specified run time, LSF invokes LSF_SERVERDIR/eadmin to trigger the action for a job overrun exception.

Example

```
JOB_OVERRUN=5
```

A job overrun exception is triggered for jobs running longer than 5 minutes.

Default

Not defined. No job overrun exceptions are detected.

JOB_STARTER

Syntax

JOB_STARTER=*starter* [*starter*] [**"%USRCMD"**] [*starter*]

Description

Creates a specific environment for submitted jobs prior to execution.

starter is any executable that can be used to start the job (i.e., can accept the job as an input argument). Optionally, additional strings can be specified.

By default, the user commands run after the job starter. A special string, %USRCMD, can be used to represent the position of the user's job in the job starter command line. The %USRCMD string and any additional commands must be enclosed in quotation marks (" ").

Example

```
JOB_STARTER=csh -c "%USRCMD;sleep 10"
```

In this case, if a user submits a job

```
% bsub myjob arguments
```

the command that actually runs is:

```
% csh -c "myjob arguments;sleep 10"
```

Default

Not defined. No job starter is used.

JOB_UNDERRUN

Syntax

JOB_UNDERRUN=*run_time*

Description

Specifies a threshold for job underrun exception handling. If a job exits before the specified number of minutes, LSF invokes LSF_SERVERDIR/eadmin to trigger the action for a job underrun exception.

Example

```
JOB_UNDERRUN=2
```

A job underrun exception is triggered for jobs running less than 2 minutes.

Default

Not defined. No job underrun exceptions are detected.

JOB_WARNING_ACTION

Syntax

JOB_WARNING_ACTION=*signal*

Description

Specifies the job action to be taken before a job control action occurs. For example, 2 minutes before the job reaches runtime limit or termination deadline, or the queue's run window is closed, an URG signal is sent to the job.

A job warning action must be specified with a job action warning time in order for job warning to take effect.

If JOB_WARNING_ACTION is specified, LSF sends the warning action to the job before the actual control action is taken. This allows the job time to save its result before being terminated by the job control action.

The warning action specified by the `bsub -wa` option overrides JOB_WARNING_ACTION in the queue. JOB_WARNING_ACTION is used as the default when no command line option is specified.

Example

JOB_WARNING_ACTION=URG

Default

Not defined

load_index

Syntax

load_index=loadSched[/loadStop]

Specify `io`, `it`, `ls`, `mem`, `pg`, `r15s`, `r1m`, `r15m`, `swp`, `tmp`, `ut`, or a non-shared custom external load index. Specify multiple lines to configure thresholds for multiple load indices.

Specify `io`, `it`, `ls`, `mem`, `pg`, `r15s`, `r1m`, `r15m`, `swp`, `tmp`, `ut`, or a non-shared custom external load index as a column. Specify multiple columns to configure thresholds for multiple load indices.

Description

Scheduling and suspending thresholds for the specified dynamic load index.

The `loadSched` condition must be satisfied before a job is dispatched to the host. If a RESUME_COND is not specified, the `loadSched` condition must also be satisfied before a suspended job can be resumed.

If the `loadStop` condition is satisfied, a job on the host is suspended.

The `loadSched` and `loadStop` thresholds permit the specification of conditions using simple AND/OR logic. Any load index that does not have a configured threshold has no effect on job scheduling.

LSF does not suspend a job if the job is the only batch job running on the host and the machine is interactively idle (`it`>0).

The `r15s`, `r1m`, and `r15m` CPU run queue length conditions are compared to the effective queue length as reported by `lsload -E`, which is normalized for multiprocessor hosts. Thresholds for these parameters should be set at appropriate levels for single processor hosts.

Example

```
MEM=100/10
```
```
SWAP=200/30
```

These two lines translate into a `loadSched` condition of

```
mem>=100 && swap>=200
```

and a `loadStop` condition of

```
mem < 10 || swap < 30
```

Default

Not defined

LOCAL_MAX_PREEXEC_RETRY

Syntax

LOCAL_MAX_PREEXEC_RETRY=_integer_

Description

The maximum number of times to attempt the pre-execution command of a job on the local cluster.

Valid values

0 < MAX_PREEXEC_RETRY < INFINIT_INT

INFINIT_INT is defined in l sf . h.

Default

Not defined. The number of preexec retry times is unlimited

MANDATORY_EXTSCHED

Syntax

MANDATORY_EXTSCHED=_external_scheduler_options_

Description

Specifies mandatory external scheduling options for the queue.

-extsched options on the bsub command are merged with MANDATORY_EXTSCHED options, and MANDATORY_EXTSCHED options override any conflicting job-level options set by -extsched.

Default

Not defined

MAX_JOB_PREEMPT

Syntax

MAX_JOB_PREEMPT=_integer_

Description

The maximum number of times a job can be preempted. Applies to queue-level jobs only.

Valid values

0 < MAX_JOB_PREEMPT < INFINIT_INT

INFINIT_INT is defined in l sf . h.

Default

Not defined. The number of preemption times is unlimited.

MAX_JOB_REQUEUE

Syntax

MAX_JOB_REQUEUE=*integer*

Description

The maximum number of times to requeue a job automatically.

Valid values

0 < MAX_JOB_REQUEUE < INFINIT_INT

INFINIT_INT is defined in l sf . h.

Default

Not defined. The number of requeue times is unlimited

MAX_PREEXEC_RETRY

Syntax

MAX_PREEXEC_RETRY=*integer*

Description

MultiCluster job forwarding model only. The maximum number of times to attempt the pre-execution command of a job from a remote cluster.

If the job's pre-execution command fails all attempts, the job is returned to the submission cluster.

Valid values

0 < MAX_PREEXEC_RETRY < INFINIT_INT

INFINIT_INT is defined in l sf . h.

Default

5

MAX_RSCHED_TIME

Syntax

MAX_RSCHED_TIME=*integer* | **infinit**

Description

MultiCluster job forwarding model only. Determines how long a MultiCluster job stays pending in the execution cluster before returning to the submission cluster. The remote timeout limit in seconds is:

```
MAX_RSCHED_TIME * MBD_SLEEP_TIME=timeout
```

Specify infinit to disable remote timeout (jobs always get dispatched in the correct FCFS order because MultiCluster jobs never get rescheduled, but MultiCluster jobs can be pending in the receive-jobs queue forever instead of being rescheduled to a better queue).

> **Note:**
>
> apply to the queue in the submission cluster (only). This
> parameter is ignored by the receiving queue.

Remote timeout limit never affects advance reservation jobs

Jobs that use an advance reservation always behave as if remote timeout is disabled.

Default

20 (20 minutes by default)

MEMLIMIT

Syntax

MEMLIMIT=[*default_limit*] *maximum_limit*

Description

The per-process (hard) process resident set size limit (in KB) for all of the processes belonging to a job from this queue (see getrlimit(2)).

Sets the maximum amount of physical memory (resident set size, RSS) that may be allocated to a process.

By default, if a default memory limit is specified, jobs submitted to the queue without a job-level memory limit are killed when the default memory limit is reached.

If you specify only one limit, it is the maximum, or hard, memory limit. If you specify two limits, the first one is the default, or soft, memory limit, and the second one is the maximum memory limit.

LSF has two methods of enforcing memory usage:
* OS Memory Limit Enforcement
* LSF Memory Limit Enforcement

OS memory limit enforcement

OS memory limit enforcement is the default MEMLIMIT behavior and does not require further configuration. OS enforcement usually allows the process to eventually run to completion. LSF passes MEMLIMIT to the OS that uses it as a guide for the system scheduler and memory allocator. The system may allocate more memory to a process if there is a surplus. When memory is low, the system takes memory from and lowers the scheduling priority (re-nice) of a process that has exceeded its declared MEMLIMIT. Only available on systems that support RLIMIT_RSS for setrlimit().

Not supported on:
- Sun Solaris 2.x
- Windows

LSF memory limit enforcement

To enable LSF memory limit enforcement, set LSB_MEMLIMIT_ENFORCE in `lsf.conf` to y. LSF memory limit enforcement explicitly sends a signal to kill a running process once it has allocated memory past MEMLIMIT.

You can also enable LSF memory limit enforcement by setting LSB_JOB_MEMLIMIT in `lsf.conf` to y. The difference between LSB_JOB_MEMLIMIT set to y and LSB_MEMLIMIT_ENFORCE set to y is that with LSB_JOB_MEMLIMIT, only the per-job memory limit enforced by LSF is enabled. The per-process memory limit enforced by the OS is disabled. With LSB_MEMLIMIT_ENFORCE set to y, both the per-job memory limit enforced by LSF and the per-process memory limit enforced by the OS are enabled.

Available for all systems on which LSF collects total memory usage.

Example

The following configuration defines a queue with a memory limit of 5000 KB:

```
Begin Queue
QUEUE_NAME  = default
DESCRIPTION = Queue with memory limit of 5000 kbytes
MEMLIMIT    = 5000
End Queue
```

Default

Unlimited

MIG

Syntax

MIG=*minutes*

Description

Enables automatic job migration and specifies the migration threshold for checkpointable or rerunnable jobs, in minutes.

LSF automatically migrates jobs that have been in the SSUSP state for more than the specified number of minutes. Specify a value of 0 to migrate jobs immediately upon suspension. The migration threshold applies to all jobs running on the host.

Job-level command line migration threshold overrides threshold configuration in application profile and queue. Application profile configuration overrides queue level configuration.

When a host migration threshold is specified, and is lower than the value for the job, the queue, or the application, the host value is used..

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

Does not affect MultiCluster jobs that are forwarded to a remote cluster.

Default

Not defined. LSF does not migrate checkpointable or rerunnable jobs automatically.

NEW_JOB_SCHED_DELAY

Syntax

NEW_JOB_SCHED_DELAY=*seconds*

Description

The number of seconds that a new job waits, before being scheduled. A value of zero (0) means the job is scheduled without any delay.

Default

2 seconds

NICE

Syntax

NICE=*integer*

Description

Adjusts the UNIX scheduling priority at which jobs from this queue execute.

The default value of 0 (zero) maintains the default scheduling priority for UNIX interactive jobs. This value adjusts the run-time priorities for batch jobs on a queue-by-queue basis, to control their effect on other batch or interactive jobs. See the ni ce(1) manual page for more details.

On Windows, this value is mapped to Windows process priority classes as follows:

*   ni ce>=0 corresponds to an priority class of I DLE
*   ni ce<0 corresponds to an priority class of NORMAL

Platform LSF on Windows does not support HI GH or REAL- TI ME priority classes.

Default

0 (zero)

NQS_QUEUES

Syntax

NQS_QUEUES=*NQS_ queue_name@ NQS_host_name* ...

Description

Makes the queue an NQS forward queue.

NQS_host_name is an NQS host name that can be the official host name or an alias name known to the LSF master host.

NQS_queue_name is the name of an NQS destination queue on this host. NQS destination queues are considered for job routing in the order in which they are listed here. If a queue

accepts the job, it is routed to that queue. If no queue accepts the job, it remains pending in the NQS forward queue.

lsb.nqsmaps must be present for the LSF system to route jobs in this queue to NQS systems.

You must configure LSB_MAX_NQS_QUEUES in lsf.conf to specify the maximum number of NQS queues allowed in the LSF cluster. This is required for LSF to work with NQS.

Since many features of LSF are not supported by NQS, the following queue configuration parameters are ignored for NQS forward queues: PJOB_LIMIT, POLICIES, RUN_WINDOW, DISPATCH_WINDOW, RUNLIMIT, HOSTS, MIG. The application-level RUNTIME parameter in lsb.applications is also ignored. In addition, scheduling load threshold parameters are ignored because NQS does not provide load information about hosts.

Default

Not defined

PJOB_LIMIT

Syntax

PJOB_LIMIT=_float_

Description

Per-processor job slot limit for the queue.

Maximum number of job slots that this queue can use on any processor. This limit is configured per processor, so that multiprocessor hosts automatically run more jobs.

Default

Unlimited

POST_EXEC

Syntax

POST_EXEC=_command_

Description

Enables post-execution processing at the queue level. The POST_EXEC command runs on the execution host after the job finishes. Post-execution commands can be configured at the application and queue levels. Application-level post-execution commands run _before_ queue-level post-execution commands.

The POST_EXEC command uses the same environment variable values as the job, and, by default, runs under the user account of the user who submits the job. To run post-execution commands under a different user account (such as root for privileged operations), configure the parameter LSB_PRE_POST_EXEC_USER in lsf.sudoers.

When a job exits with one of the queue's REQUEUE_EXIT_VALUES, LSF requeues the job and sets the environment variable LSB_JOBPEND. The post-execution command runs after the requeued job finishes.

When the post-execution command is run, the environment variable LSB_JOBEXIT_STAT is set to the exit status of the job. If the execution environment for the job cannot be set up, LSB_JOBEXIT_STAT is set to 0 (zero).

The command path can contain up to 4094 characters for UNIX and Linux, or up to 255 characters for Windows, including the directory, file name, and expanded values for %J *(job_ID)* and %I (*index_ID*).

For UNIX:

*   The pre- and post-execution commands run in the /tmp directory under /bin/sh -c, which allows the use of shell features in the commands. The following example shows valid configuration lines:

    ```
    PRE_EXEC= /usr/share/lsf/misc/testq_pre >> /tmp/pre.out
    ```

    ```
    POST_EXEC= /usr/share/lsf/misc/testq_post | grep -v "Hey!"
    ```
*   LSF sets the PATH environment variable to

    ```
    PATH='/bin /usr/bin /sbin /usr/sbin'
    ```
*   The stdin, stdout, and stderr are set to /dev/null
*   To allow UNIX users to define their own post-execution commands, an LSF administrator specifies the environment variable $USER_POSTEXEC as the POST_EXEC command. A user then defines the post-execution command:

    ```
    setenv USER_POSTEXEC /path_name
    ```

    Note:

    The path name for the post-execution command must be an absolute path. Do not define POST_EXEC= $USER_POSTEXEC when LSB_PRE_POST_EXEC_USER=root.

For Windows:

*   The pre- and post-execution commands run under cmd.exe /c
*   The standard input, standard output, and standard error are set to NULL
*   The PATH is determined by the setup of the LSF Service

Note:

For post-execution commands that execute on a Windows Server 2003, x64 Edition platform, users must have read and execute privileges for cmd.exe.

Default

Not defined. No post-execution commands are associated with the queue.

PRE_EXEC

Syntax

PRE_EXEC=*command*

Description

Enables pre-execution processing at the queue level. The PRE_EXEC command runs on the execution host before the job starts. If the PRE_EXEC command exits with a non-zero exit code, LSF requeues the job to the front of the queue.

Pre-execution commands can be configured at the queue, application, and job levels and run in the following order:

1. The queue-level command
2. The application-level or job-level command. If you specify a command at both the application and job levels, the job-level command overrides the application-level command; the application-level command is ignored.

The PRE_EXEC command uses the same environment variable values as the job, and runs under the user account of the user who submits the job. To run pre-execution commands under a different user account (such as root for privileged operations), configure the parameter LSB_PRE_POST_EXEC_USER in lsf.sudoers.

The command path can contain up to 4094 characters for UNIX and Linux, or up to 255 characters for Windows, including the directory, file name, and expanded values for %J (*job_ID*) and %I (*index_ID*).

For UNIX:

- The pre- and post-execution commands run in the /tmp directory under /bin/sh -c, which allows the use of shell features in the commands. The following example shows valid configuration lines:

  PRE_EXEC= /usr/share/lsf/misc/testq_pre >> /tmp/pre.out

  POST_EXEC= /usr/share/lsf/misc/testq_post | grep -v "Hey!"
- LSF sets the PATH environment variable to

  PATH='/bin /usr/bin /sbin /usr/sbin'
- The stdin, stdout, and stderr are set to /dev/null

For Windows:

- The pre- and post-execution commands run under cmd.exe /c
- The standard input, standard output, and standard error are set to NULL
- The PATH is determined by the setup of the LSF Service

Note:

For pre-execution commands that execute on a Windows Server 2003, x64 Edition platform, users must have read and execute privileges for cmd.exe.

Default

Not defined. No pre-execution commands are associated with the queue.

PREEMPTION

Syntax

PREEMPTION=PREEMPTIVE[[*low_queue_name*[**+***pref_level*]...]]
PREEMPTION=PREEMPTABLE[[*hi_queue_name*...]] PREEMPTION=PREEMPTIVE
[[*low_queue_name*[**+***pref_level*]...]] PREEMPTABLE[[*hi_queue_name*...]]

Description

PREEMPTIVE

Enables preemptive scheduling and defines this queue as preemptive. Jobs in this queue preempt jobs from the specified lower-priority queues or from all lower-priority queues if the parameter is specified with no queue names. PREEMPTIVE can be combined with PREEMPTABLE to specify that jobs in this queue can preempt jobs in lower-priority queues, and can be preempted by jobs in higher-priority queues.

PREEMPTABLE

Enables preemptive scheduling and defines this queue as preemptable. Jobs in this queue can be preempted by jobs from specified higher-priority queues, or from all higher-priority queues, even if the higher-priority queues are not preemptive. PREEMPTIVE can be combined with PREEMPTIVE to specify that jobs in this queue can be preempted by jobs in higher-priority queues, and can preempt jobs in lower-priority queues.

low_queue_name

Specifies the names of lower-priority queues that can be preempted.

To specify multiple queues, separate the queue names with a space, and enclose the list in a single set of square brackets.

+pref_level

Specifies to preempt this queue before preempting other queues. When multiple queues are indicated with a preference level, an order of preference is indicated: queues with higher relative preference levels are preempted before queues with lower relative preference levels set.

hi_queue_name

Specifies the names of higher-priority queues that can preempt jobs in this queue.

To specify multiple queues, separate the queue names with a space and enclose the list in a single set of square brackets.

Example: configure selective, ordered preemption across queues

The following example defines four queues, as follows:

- high

  - Has the highest relative priority of 99
  - Jobs from this queue can preempt jobs from all other queues
- medium

  - Has the second-highest relative priority at 10
  - Jobs from this queue can preempt jobs from `normal` and `low` queues, beginning with jobs from `low`, as indicated by the preference (+1)
- normal

  - Has the second-lowest relative priority, at 5
  - Jobs from this queue can preempt jobs from `low`, and can be preempted by jobs from both `high` and `medium` queues

- low

  - Has the lowest relative priority, which is also the default priority, at 1
  - Jobs from this queue can be preempted by jobs from all preemptive queues, even though it does not have the PREEMPTABLE keyword set

```
Begin Queue
QUEUE_NAME=high
PREEMPTION=PREEMPTIVE
PRIORITY=99
End Queue

Begin Queue
QUEUE_NAME=medium
PREEMPTION=PREEMPTIVE[normal low+1]
PRIORITY=10
End Queue

Begin Queue
QUEUE_NAME=normal
PREEMPTION=PREEMPTIVE[low]
PREEMPTABLE[high medium]
PRIORITY=5
End Queue

Begin Queue
QUEUE_NAME=low
PRIORITY=1
End Queue
```

PRIORITY

Syntax

PRIORITY=*integer*

Description

Specifies the relative queue priority for dispatching jobs. A higher value indicates a higher job-dispatching priority, relative to other queues.

LSF schedules jobs from one queue at a time, starting with the highest-priority queue. If multiple queues have the same priority, LSF schedules all the jobs from these queues in first-come, first-served order.

LSF queue priority is independent of the UNIX scheduler priority system for time-sharing processes. In LSF, the NICE parameter is used to set the UNIX time-sharing priority for batch jobs.

integer

Specify a number greater than or equal to 1, where 1 is the lowest priority.

Default

1

PROCESSLIMIT

Syntax

PROCESSLIMIT=[*default_limit***]** *maximum_limit*

Description

Limits the number of concurrent processes that can be part of a job.

By default, if a default process limit is specified, jobs submitted to the queue without a job-level process limit are killed when the default process limit is reached.

If you specify only one limit, it is the maximum, or hard, process limit. If you specify two limits, the first one is the default, or soft, process limit, and the second one is the maximum process limit.

Default

Unlimited

PROCLIMIT

Syntax

PROCLIMIT=[*minimum_limit* [*default_limit*]] *maximum_limit*

Description

Maximum number of slots that can be allocated to a job. For parallel jobs, the maximum number of processors that can be allocated to the job.

Job-level processor limits (bsub -n) override queue-level PROCLIMIT. Job-level limits must fall within the maximum and minimum limits of the application profile and the queue. Application-level PROCLIMIT in lsb.applications overrides queue-level specificiation.

Optionally specifies the minimum and default number of job slots.

All limits must be positive numbers greater than or equal to 1 that satisfy the following relationship:

1 <= *minimum* <= *default* <= *maximum*

You can specify up to three limits in the PROCLIMIT parameter:

Jobs that request fewer slots than the minimum PROCLIMIT or more slots than the maximum PROCLIMIT cannot use the queue and are rejected. If the job requests minimum and maximum job slots, the maximum slots requested cannot be less than the minimum PROCLIMIT, and the minimum slots requested cannot be more than the maximum PROCLIMIT.

Default

Unlimited, the default number of slots is 1

QJOB_LIMIT

Syntax

QJOB_LIMIT=*integer*

Description

Job slot limit for the queue. Total number of job slots that this queue can use.

Default

Unlimited

QUEUE_GROUP

Syntax

QUEUE_GROUP=*queue1*, *queue2* ...

Description

Configures absolute priority scheduling (APS) across multiple queues.

When APS is enabled in the queue with APS_PRIORITY, the FAIRSHARE_QUEUES parameter is ignored. The QUEUE_GROUP parameter replaces FAIRSHARE_QUEUES, which is obsolete in LSF 7.0.

Default

Not defined

QUEUE_NAME

Syntax

QUEUE_NAME=*string*

Description

Required. Name of the queue.

Specify any ASCII string up to 59 characters long. You can use letters, digits, underscores (_) or dashes (-). You cannot use blank spaces. You cannot specify the reserved name default.

Default

You must specify this parameter to define a queue. The default queue automatically created by LSF is named default.

RCVJOBS_FROM

Syntax

RCVJOBS_FROM=*cluster_name* ... | **allclusters**

Description

MultiCluster only. Defines a MultiCluster receive-jobs queue.

Specify cluster names, separated by a space. The administrator of each remote cluster determines which queues in that cluster forward jobs to the local cluster.

Use the keyword allclusters to specify any remote cluster.

Example

```
RCVJOBS_FROM=cluster2 cluster4 cluster6
```

This queue accepts remote jobs from clusters 2, 4, and 6.

REMOTE_MAX_PREEXEC_RETRY

Syntax

REMOTE_MAX_PREEXEC_RETRY=*integer*

Description

The maximum number of times to attempt the pre-execution command of a job on the remote cluster.

Valid values

0 < MAX_PREEXEC_RETRY < INFINIT_INT

INFINIT_INT is defined in lsf.h.

Default

5

REQUEUE_EXIT_VALUES

Syntax

REQUEUE_EXIT_VALUES=[*exit_code* ...] [**EXCLUDE(***exit_code* ...**)**]

Description

Enables automatic job requeue and sets the LSB_EXIT_REQUEUE environment variable. Use spaces to separate multiple exit codes. Application-level exit values override queue-level values. Job-level exit values (bsub -Q) override application-level and queue-level values.

exit_code has the following form:

```
"[all] [~number ...] | [number ...]"
```

The reserved keyword all specifies all exit codes. Exit codes are typically between 0 and 255. Use a tilde (~) to exclude specified exit codes from the list.

Jobs are requeued to the head of the queue. The output from the failed run is not saved, and the user is not notified by LSF.

Define an exit code as EXCLUDE(*exit_code*) to enable exclusive job requeue. Exclusive job requeue does not work for parallel jobs.

For MultiCluster jobs forwarded to a remote execution cluster, the exit values specified in the submission cluster with the EXCLUSIVE keyword are treated as if they were non-exclusive.

You can also requeue a job if the job is terminated by a signal.

If a job is killed by a signal, the exit value is 128+*signal_value*. The sum of 128 and the signal value can be used as the exit code in the parameter REQUEUE_EXIT_VALUES.

For example, if you want a job to rerun if it is killed with a signal 9 (SIGKILL), the exit value would be 128+9=137. You can configure the following requeue exit value to allow a job to be requeue if it was kill by signal 9:

```
REQUEUE_EXI T_VALUES=137
```

If mbatchd is restarted, it does not remember the previous hosts from which the job exited with an exclusive requeue exit code. In this situation, it is possible for a job to be dispatched to hosts on which the job has previously exited with an exclusive exit code.

You should configure REQUEUE_EXIT_VALUES for interruptible backfill queues (INTERRUPTIBLE_BACKFILL=*seconds*).

Example

REQUEUE_EXIT_VALUES=30 EXCLUDE(20)

means that jobs with exit code 30 are requeued, jobs with exit code 20 are requeued exclusively, and jobs with any other exit code are not requeued.

Default

Not defined. Jobs are not requeued.

RERUNNABLE

Syntax

RERUNNABLE=yes | **no**

Description

If yes, enables automatic job rerun (restart).

Rerun is disabled when RERUNNABLE is set to no. The yes and no arguments are not case sensitive.

For MultiCluster jobs, the setting in the submission queue is used, and the setting in the execution queue is ignored.

Members of a chunk job can be rerunnable. If the execution host becomes unavailable, rerunnable chunk job members are removed from the job chunk and dispatched to a different execution host.

Default

no

RESOURCE_RESERVE

Syntax

RESOURCE_RESERVE=MAX_RESERVE_TIME[*integer*]

Description

Enables processor reservation and memory reservation for pending jobs for the queue. Specifies the number of dispatch turns (MAX_RESERVE_TIME) over which a job can reserve job slots and memory.

Overrides the SLOT_RESERVE parameter. If both RESOURCE_RESERVE and SLOT_RESERVE are defined in the same queue, an error is displayed when the cluster is reconfigured, and SLOT_RESERVE is ignored. Job slot reservation for parallel jobs is enabled by RESOURCE_RESERVE if the LSF scheduler plugin module names for both resource reservation and parallel batch jobs (schmod_parallel and schmod_reserve) are configured in the lsb.modules file: The schmod_parallel name *must* come before schmod_reserve in lsb.modules.

If a job has not accumulated enough memory or job slots to start by the time MAX_RESERVE_TIME expires, it releases all its reserved job slots or memory so that other pending jobs can run. After the reservation time expires, the job cannot reserve memory or slots for one scheduling session, so other jobs have a chance to be dispatched. After one scheduling session, the job can reserve available memory and job slots again for another period specified by MAX_RESERVE_TIME.

If BACKFILL is configured in a queue, and a run limit is specified with -W on bsub or with RUNLIMIT in the queue, backfill jobs can use the accumulated memory reserved by the other jobs in the queue, as long as the backfill job can finish before the predicted start time of the jobs with the reservation.

Unlike slot reservation, which only applies to parallel jobs, memory reservation and backfill on memory apply to sequential and parallel jobs.

Example

```
RESOURCE_RESERVE=MAX_RESERVE_TIME[5]
```

This example specifies that jobs have up to 5 dispatch turns to reserve sufficient job slots or memory (equal to 5 minutes, by default).

Default

Not defined. No job slots or memory is reserved.

RES_REQ

Syntax

RES_REQ=*res_req*

Description

Resource requirements used to determine eligible hosts. Specify a resource requirement string as usual. The resource requirement string lets you specify conditions in a more flexible manner than using the load thresholds. Resource requirement strings can be simple (applying to the entire job) or compound (applying to the specified number of slots).

When a compound resource requirement is set for a queue, it will be ignored unless it is the only resource requirement specified (no resource requirements are set at the job-level or application-level).

When a simple resource requirement is set for a queue and a compound resource requirement is set at the job-level or application-level, the queue-level requirements merge as they do for simple resource requirements. However, any job-based resources defined in the queue only apply to the first term of the merged compound resource requirements.

When LSF_STRICT_RESREQ=Y is configured in lsf.conf, resource requirement strings in select sections must conform to a more strict syntax. The strict resource requirement syntax only applies to the select section. It does not apply to the other resource requirement sections (order, rusage, same, span, or cu). When LSF_STRICT_RESREQ=Y in lsf.conf, LSF rejects resource requirement strings where an rusage section contains a non-consumable resource.

For simple resource requirements, the select sections from all levels must be satisfied and the same sections from all levels are combined. cu, order, and span sections at the job-level overwrite those at the application-level which overwrite those at the queue-level. Multiple rusage definitions are merged, with the job-level rusage taking precedence over the application-level, and application-level taking precedence over the queue-level.

The simple resource requirement rusage section can specify additional requests. To do this, use the OR (||) operator to separate additional rusage strings.

> **Note:**
>
> Compound resource requirements do not support use of the || operator within rusage sections, multiple -R options, or the cu section.

For example:

Queue-level RES_REQ:

```
RES_REQ=rusage[mem=200:lic=1] ...
```

For the job submission:

```
bsub -R'rusage[mem=100]' ...
```

the resulting requirement for the job is

```
rusage[mem=100:lic=1]
```

where mem=100 specified by the job overrides mem=200 specified by the queue. However, lic=1 from queue is kept, since job does not specify it.

Queue-level RES_REQ with decay and duration defined:

```
RES_REQ=rusage[mem=200:duration=20:decay=1] ...
```

For a job submission with no decay or duration:

```
bsub -R'rusage[mem=100]' ...
```

the resulting requirement for the job is:

```
rusage[mem=100:duration=20:decay=1]
```

Queue-level duration and decay are merged with the job-level specification, and mem=100 for the job overrides mem=200 specified by the queue. However, duration=20 and decay=1 from queue are kept, since job does not specify them.

The order section defined at the queue level is ignored if any resource requirements are specified at the job level (if the job-level resource requirements do not include the order section, the default order, r15s:pg, is used instead of the queue-level resource requirement).

If RES_REQ is defined at the queue level and there are no load thresholds defined, the pending reasons for each individual load index are not displayed by bjobs.

The span section defined at the queue level is ignored if the span section is also defined at the job level or in an application profile.

Note:

Define span[hosts=-1] in the application profile or bsub -R resource requirement string to override the span section setting in the queue.

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

Default

select[type==local] order[r15s:pg]. If this parameter is defined and a host model or Boolean resource is specified, the default type is any.

RESUME_COND

Syntax

RESUME_COND=_res_req_

Use the select section of the resource requirement string to specify load thresholds. All other sections are ignored.

Description

LSF automatically resumes a suspended (SSUSP) job in this queue if the load on the host satisfies the specified conditions.

If RESUME_COND is not defined, then the loadSched thresholds are used to control resuming of jobs. The loadSched thresholds are ignored, when resuming jobs, if RESUME_COND is defined.

RUN_WINDOW

Syntax

RUN_WINDOW=_time_window_ ...

Description

Time periods during which jobs in the queue are allowed to run.

When the window closes, LSF suspends jobs running in the queue and stops dispatching jobs from the queue. When the window reopens, LSF resumes the suspended jobs and begins dispatching additional jobs.

Default

Not defined. Queue is always active.

RUNLIMIT

Syntax

RUNLIMIT=[*default_limit*] *maximum_limit*

where *default_limit* and *maximum_limit* are:

[*hour*:]*minute*[/*host_name* | /*host_model*]

Description

The maximum run limit and optionally the default run limit. The name of a host or host model specifies the runtime normalization host to use.

By default, jobs that are in the RUN state for longer than the specified maximum run limit are killed by LSF. You can optionally provide your own termination job action to override this default.

Jobs submitted with a job-level run limit (bsub -W) that is less than the maximum run limit are killed when their job-level run limit is reached. Jobs submitted with a run limit greater than the maximum run limit are rejected by the queue.

If a default run limit is specified, jobs submitted to the queue without a job-level run limit are killed when the default run limit is reached. The default run limit is used with backfill scheduling of parallel jobs.

> **Note:**
>
> If you want to provide an estimated run time for scheduling purposes without killing jobs that exceed the estimate, define the RUNTIME parameter in an application profile instead of a run limit (see lsb.applications for details).

If you specify only one limit, it is the maximum, or hard, run limit. If you specify two limits, the first one is the default, or soft, run limit, and the second one is the maximum run limit. The number of minutes may be greater than 59. Therefore, three and a half hours can be specified either as 3:30, or 210.

The run limit is in the form of [hour:]minute. The minutes can be specified as a number greater than 59. For example, three and a half hours can either be specified as 3:30, or 210.

The run limit you specify is the normalized run time. This is done so that the job does approximately the same amount of processing, even if it is sent to host with a faster or slower CPU. Whenever a normalized run time is given, the actual time on the execution host is the

specified time multiplied by the CPU factor of the normalization host then divided by the CPU factor of the execution host.

If ABS_RUNLIMIT=Y is defined in lsb.params, the runtime limit is not normalized by the host CPU factor. Absolute wall-clock run time is used for all jobs submitted to a queue with a run limit configured.

Optionally, you can supply a host name or a host model name defined in LSF. You must insert '/' between the run limit and the host name or model name. (See lsinfo(1) to get host model information.)

If no host or host model is given, LSF uses the default runtime normalization host defined at the queue level (DEFAULT_HOST_SPEC in lsb.queues) if it has been configured; otherwise, LSF uses the default CPU time normalization host defined at the cluster level (DEFAULT_HOST_SPEC in lsb.params) if it has been configured; otherwise, the host with the largest CPU factor (the fastest host in the cluster).

For MultiCluster jobs, if no other CPU time normalization host is defined and information about the submission host is not available, LSF uses the host with the largest CPU factor (the fastest host in the cluster).

Jobs submitted to a chunk job queue are not chunked if RUNLIMIT is greater than 30 minutes.

RUNLIMIT is required for queues configured with INTERRUPTIBLE_BACKFILL.

Default

Unlimited

SLOT_POOL

Syntax

SLOT_POOL=*pool_name*

Description

Name of the pool of job slots the queue belongs to for queue-based fairshare. A queue can only belong to one pool. All queues in the pool must share the same set of hosts.

Valid value

Specify any ASCII string up to 60 characters long. You can use letters, digits, underscores (_) or dashes (-). You cannot use blank spaces.

Default

Not defined. No job slots are reserved.

SLOT_RESERVE

Syntax

SLOT_RESERVE=MAX_RESERVE_TIME[*integer***]**

Description

Enables processor reservation for the queue and specifies the reservation time. Specify the keyword MAX_RESERVE_TIME and, in square brackets, the number of MBD_SLEEP_TIME cycles over which a job can reserve job slots. MBD_SLEEP_TIME is defined in `lsb.params`; the default value is 60 seconds.

If a job has not accumulated enough job slots to start before the reservation expires, it releases all its reserved job slots so that other jobs can run. Then, the job cannot reserve slots for one scheduling session, so other jobs have a chance to be dispatched. After one scheduling session, the job can reserve job slots again for another period specified by SLOT_RESERVE.

SLOT_RESERVE is overridden by the RESOURCE_RESERVE parameter.

If both RESOURCE_RESERVE and SLOT_RESERVE are defined in the same queue, job slot reservation and memory reservation are enabled and an error is displayed when the cluster is reconfigured. SLOT_RESERVE is ignored.

Job slot reservation for parallel jobs is enabled by RESOURCE_RESERVE if the LSF scheduler plugin module names for both resource reservation and parallel batch jobs (schmod_parallel and schmod_reserve) are configured in the `lsb.modules` file: The schmod_parallel name *must* come before schmod_reserve in `lsb.modules`.

If BACKFILL is configured in a queue, and a run limit is specified at the job level (`bsub -W`), application level (RUNLIMIT in `lsb.applications`), or queue level (RUNLIMIT in `lsb.queues`), or if an estimated run time is specified at the application level (RUNTIME in `lsb.applications`), backfill parallel jobs can use job slots reserved by the other jobs, as long as the backfill job can finish before the predicted start time of the jobs with the reservation.

Unlike memory reservation, which applies both to sequential and parallel jobs, slot reservation applies only to parallel jobs.

Example

```
SLOT_RESERVE=MAX_RESERVE_TIME[5]
```

This example specifies that parallel jobs have up to 5 cycles of MBD_SLEEP_TIME (5 minutes, by default) to reserve sufficient job slots to start.

Default

Not defined. No job slots are reserved.

SLOT_SHARE

Syntax

SLOT_SHARE=_integer_

Description

Share of job slots for queue-based fairshare. Represents the percentage of running jobs (job slots) in use from the queue. SLOT_SHARE must be greater than zero (0) and less than or equal to 100.

The sum of SLOT_SHARE for all queues in the pool does not need to be 100%. It can be more or less, depending on your needs.

Default

Not defined

SNDJOBS_TO

Syntax

SNDJOBS_TO=*queue_name@cluster_name ...*

Description

Defines a MultiCluster send-jobs queue.

Specify remote queue names, in the form *queue_name@cluster_name*, separated by a space.

This parameter is ignored if lsb.queues HOSTS specifies remote (borrowed) resources.

Example

```
SNDJOBS_TO=queue2@cluster2 queue3@cluster2 queue3@cluster3
```

STACKLIMIT

Syntax

STACKLIMIT=*integer*

Description

The per-process (hard) stack segment size limit (in KB) for all of the processes belonging to a job from this queue (see getrlimit(2)).

Default

Unlimited

STOP_COND

Syntax

STOP_COND=*res_req*

Use the select section of the resource requirement string to specify load thresholds. All other sections are ignored.

Description

LSF automatically suspends a running job in this queue if the load on the host satisfies the specified conditions.

- LSF does not suspend the only job running on the host if the machine is interactively idle ($it > 0$).
- LSF does not suspend a forced job (brun -f).
- LSF does not suspend a job because of paging rate if the machine is interactively idle.

If STOP_COND is specified in the queue and there are no load thresholds, the suspending reasons for each individual load index is not displayed by bjobs.

Example

```
STOP_COND= select[((!cs && it < 5) || (cs && mem < 15 && swp < 50))]
```

In this example, assume "cs" is a Boolean resource indicating that the host is a computer server. The stop condition for jobs running on computer servers is based on the availability of swap memory. The stop condition for jobs running on other kinds of hosts is based on the idle time.

SWAPLIMIT

Syntax

SWAPLIMIT=*integer*

Description

The amount of total virtual memory limit (in KB) for a job from this queue.

This limit applies to the whole job, no matter how many processes the job may contain.

The action taken when a job exceeds its SWAPLIMIT or PROCESSLIMIT is to send SIGQUIT, SIGINT, SIGTERM, and SIGKILL in sequence. For CPULIMIT, SIGXCPU is sent before SIGINT, SIGTERM, and SIGKILL.

Default

Unlimited

TERMINATE_WHEN

Syntax

TERMINATE_WHEN=[**LOAD**] [**PREEMPT**] [**WINDOW**]

Description

Configures the queue to invoke the TERMINATE action instead of the SUSPEND action in the specified circumstance.

- LOAD: kills jobs when the load exceeds the suspending thresholds.
- PREEMPT: kills jobs that are being preempted.
- WINDOW: kills jobs if the run window closes.

If the TERMINATE_WHEN job control action is applied to a chunk job, sbatchd kills the chunk job element that is running and puts the rest of the waiting elements into pending state to be rescheduled later.

Example

Set TERMINATE_WHEN to WINDOW to define a night queue that kills jobs if the run window closes:

```
Begin Queue

NAME            = night

RUN_WINDOW      = 20:00-08:00

TERMINATE_WHEN  = WINDOW

JOB_CONTROLS    = TERMINATE[kill -KILL $LS_JOBPGIDS; mail - s "job $LSB_JOBID
killed by queue run window" $USER < /dev/null]

End Queue
```

THREADLIMIT

Syntax

THREADLIMIT=[*default_limit*] *maximum_limit*

Description

Limits the number of concurrent threads that can be part of a job. Exceeding the limit causes the job to terminate. The system sends the following signals in sequence to all processes belongs to the job: SIGINT, SIGTERM, and SIGKILL.

By default, if a default thread limit is specified, jobs submitted to the queue without a job-level thread limit are killed when the default thread limit is reached.

If you specify only one limit, it is the maximum, or hard, thread limit. If you specify two limits, the first one is the default, or soft, thread limit, and the second one is the maximum thread limit.

Both the default and the maximum limits must be positive integers. The default limit must be less than the maximum limit. The default limit is ignored if it is greater than the maximum limit.

Examples

```
THREADLIMIT=6
```

No default thread limit is specified. The value 6 is the default and maximum thread limit.

```
THREADLIMIT=6 8
```

The first value (6) is the default thread limit. The second value (8) is the maximum thread limit.

Default

Unlimited

UJOB_LIMIT

Syntax

UJOB_LIMIT=*integer*

Description

Per-user job slot limit for the queue. Maximum number of job slots that each user can use in this queue.

Default

Unlimited

USE_PAM_CREDS

Syntax

USE_PAM_CREDS=y | **n**

Description

If USE_PAM_CREDS=y, applies PAM limits to a queue when its job is dispatched to a Linux host using PAM. PAM limits are system resource limits defined in limits.conf.

When USE_PAM_CREDS is enabled, PAM limits override others. For example, the PAM limit is used even if queue-level soft limit is less than PAM limit. However, it still cannot exceed queue's hard limit.

If the execution host does not have PAM configured and this parameter is enabled, the job fails.

For parallel jobs, only takes effect on the first execution host.

USE_PAM_CREDS only applies on the following platforms:

- linux2.6-glibc2.3-ia64
- linux2.6-glibc2.3-ia64-slurm
- linux2.6-glibc2.3-ppc64
- linux2.6-glibc2.3-sn-ipf
- linux2.6-glibc2.3-x86
- linux2.6-glibc2.3-x86_64
- linux2.6-glibc2.3-x86_64-slurm

Overrides MEMLIMIT_TYPE=Process.

Overridden (for CPU limit only) by LSB_JOB_CPULIMIT=y.

Overridden (for memory limits only) by LSB_JOB_MEMLIMIT=y.

Default

n

USERS

Syntax

USERS=all [~*user_name* ...] [~*user_group* ...] | [*user_name* ...] [*user_group* [~*user_group* ...] ...]

Description

A space-separated list of user names or user groups that can submit jobs to the queue. LSF cluster administrators are automatically included in the list of users. LSF cluster administrators can submit jobs to this queue, or switch (bswitch) any user's jobs into this queue.

If user groups are specified, each user in the group can submit jobs to this queue. If FAIRSHARE is also defined in this queue, only users defined by both parameters can submit jobs, so LSF administrators cannot use the queue if they are not included in the share assignments.

User names must be valid login names. To specify a Windows user account, include the domain name in uppercase letters (*DOMAIN_NAME\user_name*).

User group names can be LSF user groups or UNIX and Windows user groups. To specify a Windows user group, include the domain name in uppercase letters (*DOMAIN_NAME \user_group*).

Use the keyword all to specify all users or user groups in a cluster.

Use the not operator (~) to exclude users from the all specification or from user groups. This is useful if you have a large number of users but only want to exclude a few users or groups from the queue definition.

The not operator (~) can only be used with the all keyword or to exclude users from user groups.

> **Caution:**
>
> The not operator does not exclude LSF administrators from the queue definintion.

Default

all (all users can submit jobs to the queue)

Examples

- USERS=user1 user2
- USERS=all ~user1 ~user2
- USERS=all ~ugroup1
- USERS=groupA ~user3 ~user4

Automatic time-based configuration

Variable configuration is used to automatically change LSF configuration based on time windows. You define automatic configuration changes in lsb.queues by using if-else constructs and time expressions. After you change the files, reconfigure the cluster with the badmin reconfig command.

The expressions are evaluated by LSF every 10 minutes based on mbatchd start time. When an expression evaluates true, LSF dynamically changes the configuration based on the associated configuration statements. Reconfiguration is done in real time without restarting mbatchd, providing continuous system availability.

Example

```
Begin Queue
...
#if time(8:30-18:30)
INTERACTIVE  = ONLY  # interactive only during day shift #endif
...
End Queue
```

lsb.resources

The lsb.resources file contains configuration information for resource allocation limits, exports, and resource usage limits. This file is optional.

The lsb.resources file is stored in the directory LSB_CONFDIR/*cluster_name*/configdir, where LSB_CONFDIR is defined in lsf.conf.

Changing lsb.resources configuration

After making any changes to lsb.resources, run badmin reconfig to reconfigure mbatchd.

Limit section

The Limit section sets limits for the maximum amount of the specified resources that must be available for different classes of jobs to start, and which resource consumers the limits apply to. Limits are enforced during job resource allocation.

Tip:

For limits to be enforced, jobs must specify rusage resource requirements (bsub -R or RES_REQ in lsb.queues).

The blimits command displays view current usage of resource allocation limits configured in Limit sections in lsb.resources:

Limit section structure

Each set of limits is defined in a Limit section enclosed by Begin Limit and End Limit.

A Limit section has two formats:

- Vertical tabular
- Horizontal

The file can contain sections in both formats. In either format, you must configure a limit for at least one consumer and one resource. The Limit section cannot be empty.

Vertical tabular format

Use the vertical format for simple configuration conditions involving only a few consumers and resource limits.

The first row consists of an optional NAME and the following keywords for:

- Resource types:
    - SLOTS or SLOTS_PER_PROCESSOR
    - MEM (MB or percentage)
    - SWP (MB or percentage)
    - TMP (MB or percentage)
    - LICENSE
    - JOBS
    - RESOURCE
- Consumer types:

- USERS or PER_USER
- QUEUES or PER_QUEUE
- HOSTS or PER_HOST
- PROJECTS or PER_PROJECT

Each subsequent row describes the configuration information for resource consumers and the limits that apply to them. Each line must contain an entry for each keyword. Use empty parentheses () or a dash (-) to specify the default value for an entry. Fields cannot be left blank. For resources, the default is no limit; for consumers, the default is all consumers.

Tip:

Multiple entries must be enclosed in parentheses. For RESOURCE and LICENSE limits, resource and license names must be enclosed in parentheses.

Horizontal format

Use the horizontal format to give a name for your limits and to configure more complicated combinations of consumers and resource limits.

The first line of the Limit section gives the name of the limit configuration.

Each subsequent line in the Limit section consists of keywords identifying the resource limits:

- Job slots and per-processor job slots
- Memory (MB or percentage)
- Swap space (MB or percentage)
- Tmp space (MB or percentage)
- Software licenses
- Running and suspended (RUN, SSUSP, USUSP) jobs
- Other shared resources

and the resource *consumers* to which the limits apply:

- Users and user groups
- Hosts and host groups
- Queues
- Projects

Example: Vertical tabular format

In the following limit configuration:

- Jobs from user1 and user3 are limited to 2 job slots on hostA
- Jobs from user2 on queue normal are limited to 20 MB of memory
- The short queue can have at most 200 running and suspended jobs

```
Begin Limit
NAME      USERS            QUEUES    HOSTS     SLOTS   MEM   SWP   TMP   JOBS
limit1    (user1 user3)    -         hostA     2       -     -     -     -
-         user2            normal    -         -       20    -     -     -
-         -                short     -         -       -     -     -     200
End Limit
```

Jobs that do not match these limits; that is, all users except user1 and user3 running jobs on hostA and all users except user2 submitting jobs to queue normal , have no limits.

Example: Horizontal format

All users in user group ugroup1 except user1 using queue1 and queue2 and running jobs on hosts in host group hgroup1 are limited to 2 job slots per processor on each host:

```
Begin Limit
# ugroup1 except user1 uses queue1 and queue2 with 2 job slots
# on each host in hgroup1
NAME            = limit1
# Resources
SLOTS_PER_PROCESSOR = 2
#Consumers
QUEUES          = queue1 queue2
USERS           = ugroup1 ~user1
PER_HOST        = hgroup1
End Limit
```

Compatibility with lsb.queues, lsb.users, and lsb.hosts

The Limit section of lsb.resources does not support the keywords or format used in lsb.users, lsb.hosts, and lsb.queues. However, your existing job slot limit configuration in these files will continue to apply.

Job slot limits are the only type of limit you can configure in lsb.users, lsb.hosts, and lsb.queues. You cannot configure limits for user groups, host groups, and projects in lsb.users, lsb.hosts, and lsb.queues. You should not configure any new resource allocation limits in lsb.users, lsb.hosts, and lsb.queues. Use lsb.resources to configure all new resource allocation limits, including job slot limits. Limits on running and suspended jobs can only be set in lsb.resources.

Existing limits in lsb.users, lsb.hosts, and lsb.queues with the same scope as a new limit in lsb.resources, but with a different value are ignored. The value of the new limit in lsb.resources is used. Similar limits with different scope enforce the most restrictive limit.

Parameters

- HOSTS
- JOBS
- LICENSE
- MEM
- NAME
- PER_HOST
- PER_PROJECT
- PER_QUEUE
- PER_USER
- PROJECTS
- QUEUES

- RESOURCE
- SLOTS
- SLOTS_PER_PROCESSOR
- SWP
- TMP
- USERS

HOSTS

Syntax

HOSTS=all [~]*host_name* ... | **all** [~]*host_group* ...

HOSTS

([-] | **all** [~]*host_name* ... | **all** [~]*host_group* ... **)**

Description

A space-separated list of hosts, host groups defined in lsb.hosts on which limits are enforced. Limits are enforced on all hosts or host groups listed.

If a group contains a subgroup, the limit also applies to each member in the subgroup recursively.

To specify a per-host limit, use the PER_HOST keyword. Do not configure HOSTS and PER_HOST limits in the same Limit section.

If you specify MEM, TMP, or SWP as a percentage, you must specify PER_HOST and list the hosts that the limit is to be enforced on. You cannot specify HOSTS.

In horizontal format, use only one HOSTS line per Limit section.

Use the keyword all to configure limits that apply to all hosts in a cluster.

Use the not operator (~) to exclude hosts from the all specification in the limit. This is useful if you have a large cluster but only want to exclude a few hosts from the limit definition.

In vertical tabular format, multiple host names must be enclosed in parentheses.

In vertical tabular format, use empty parentheses () or a dash (-) to indicate all hosts. Fields cannot be left blank.

Default

all (limits are enforced on all hosts in the cluster).

Example 1

```
HOSTS=Group1 ~hostA hostB hostC
```

Enforces limits on hostB, hostC, and all hosts in Group1 except for hostA.

Example 2

```
HOSTS=all ~group2 ~hostA
```

Enforces limits on all hosts in the cluster, except for hostA and the hosts in group2.

Example 3

```
HOSTS                     SWP (all ~hostK ~hostM)              10
```

Enforces a 10 MB swap limit on all hosts in the cluster, except for host K and host M

JOBS

Syntax

JOBS=*integer*

JOBS

- | *integer*

Description

Maximum number of running or suspended (RUN, SSUSP, USUSP) jobs available to resource consumers. Specify a positive integer greater than or equal 0. Job limits can be defined in both vertical and horizontal limit formats.

With MultiCluster resource lease model, this limit applies only to local hosts being used by the local cluster. The job limit for hosts exported to a remote cluster is determined by the host export policy, not by this parameter. The job limit for borrowed hosts is determined by the host export policy of the remote cluster.

If SLOTS are configured in the Limit section, the most restrictive limit is applied.

If HOSTS are configured in the Limit section, JOBS is the number of running and suspended jobs on a host. If preemptive scheduling is used, the suspended jobs are not counted against the job limit.

Use this parameter to prevent a host from being overloaded with too many jobs, and to maximize the throughput of a machine.

If only QUEUES are configured in the Limit section, JOBS is the maximum number of jobs that can run in the listed queues for any hosts, users, or projects.

If only USERS are configured in the Limit section, JOBS is the maximum number of jobs that the users or user groups can run on any hosts, queues, or projects.

If only HOSTS are configured in the Limit section, JOBS is the maximum number of jobs that can run on the listed hosts for any users, queues, or projects.

If only PROJECTS are configured in the Limit section, JOBS is the maximum number of jobs that can run under the listed projects for any users, queues, or hosts.

Use QUEUES or PER_QUEUE, USERS or PER_USER, HOSTS or PER_HOST, and PROJECTS or PER_PROJECT in combination to further limit jobs available to resource consumers.

In horizontal format, use only one JOBS line per Limit section.

In vertical format, use empty parentheses () or a dash (-) to indicate the default value (no limit). Fields cannot be left blank.

Default

No limit

Example

```
JOBS=20
```

LICENSE

Syntax

LICENSE=[*license_name,integer*] [[*license_name,integer*] ...]

LICENSE

([*license_name,integer*] [[*license_name,integer*] ...] **)**

Description

Maximum number of specified software licenses available to resource consumers. The value must be a positive integer greater than or equal to zero.

Software licenses must be defined as decreasing numeric shared resources in lsf.shared.

The RESOURCE keyword is a synonym for the LICENSE keyword. You cannot specify RESOURCE and LICENSE in the same Limit section.

In horizontal format, use only one LICENSE line per Limit section.

In vertical tabular format, license entries must be enclosed in parentheses.

In vertical tabular format, use empty parentheses () or a dash (-) to indicate the default value (no limit). Fields cannot be left blank.

Default

None

Examples

```
LICENSE=[verilog, 4] [spice, 2]
```

```
Begin Limit
LICENSE                     PER_HOST
([verilog,  1])             (all ~hostA)
([verilog,  1] [spice, 2])  (hostA)
End Limit
```

MEM

Syntax

MEM=*integer*[**%**]

MEM

- | *integer*[**%**]

Description

Maximum amount of memory available to resource consumers. Specify a value in MB or a percentage (%) as a positive integer greater than or equal 0. If you specify a percentage, you must also specify PER_HOST and list the hosts that the limit is to be enforced on.

The Limit section is ignored if MEM is specified as a percentage:

- Without PER_HOST, or
- With HOSTS

In horizontal format, use only one MEM line per Limit section.

In vertical tabular format, use empty parentheses () or a dash (-) to indicate the default value (no limit). Fields cannot be left blank.

If only QUEUES are configured in the Limit section, MEM must be an integer value. MEM is the maximum amount of memory available to the listed queues for any hosts, users, or projects.

If only USERS are configured in the Limit section, MEM must be an integer value. MEM is the maximum amount of memory that the users or user groups can use on any hosts, queues, or projects.

If only HOSTS are configured in the Limit section, MEM must be an integer value. It cannot be a percentage. MEM is the maximum amount of memory available to the listed hosts for any users, queues, or projects.

If only PROJECTS are configured in the Limit section, MEM must be an integer value. MEM is the maximum amount of memory available to the listed projects for any users, queues, or hosts.

Use QUEUES or PER_QUEUE, USERS or PER_USER, HOSTS or PER_HOST, and PROJECTS or PER_PROJECT in combination to further limit memory available to resource consumers.

Default

No limit

Example

```
MEM=20
```

NAME

Syntax

NAME=_limit_name_

NAME

- | _limit_name_

Description

Name of the Limit section

Specify any ASCII string 40 characters or less. You can use letters, digits, underscores (_) or dashes (-). You cannot use blank spaces.

If duplicate limit names are defined, the Limit section is ignored. If value of NAME is not defined in vertical format, or defined as (-), blimtis displays NONAME*nnn*.

Default

None. In horizontal format, you must provide a name for the Limit section. NAME is optional in the vertical format.

Example

```
NAME=short_limits
```

PER_HOST

Syntax

PER_HOST=all [~]*host_name* ... | all [**~**]*host_group* ...

PER_HOST

([-] | **all** [**~**]*host_name* ... | **all** [**~**]*host_group* ... **)**

Description

A space-separated list of host or host groups defined in lsb.hosts on which limits are enforced. Limits are enforced on each host or individually to each host of the host group listed. If a group contains a subgroup, the limit also applies to each member in the subgroup recursively.

Do not configure PER_HOST and HOSTS limits in the same Limit section.

In horizontal format, use only one PER_HOST line per Limit section.

If you specify MEM, TMP, or SWP as a percentage, you must specify PER_HOST and list the hosts that the limit is to be enforced on. You cannot specify HOSTS.

Use the keyword all to configure limits that apply to each host in a cluster. If host groups are configured, the limit applies to each member of the host group, not the group as a whole.

Use the not operator (~) to exclude hosts or host groups from the all specification in the limit. This is useful if you have a large cluster but only want to exclude a few hosts from the limit definition.

In vertical tabular format, multiple host names must be enclosed in parentheses.

In vertical tabular format, use empty parentheses () or a dash (-) to indicate each host or host group member. Fields cannot be left blank.

Default

None. If no limit is specified for PER_HOST or HOST, no limit is enforced on any host or host group.

Example

```
PER_HOST=hostA hgroup1 ~hostC
```

PER_PROJECT

Syntax

PER_PROJECT=all [**~**]*project_name* ...

PER_PROJECT

([**-**] | **all** [**~**]*project_name* ... **)**

Description

A space-separated list of project names on which limits are enforced. Limits are enforced on each project listed.

Do not configure PER_PROJECT and PROJECTS limits in the same Limit section.

In horizontal format, use only one PER_PROJECT line per Limit section.

Use the keyword all to configure limits that apply to each project in a cluster.

Use the not operator (~) to exclude projects from the all specification in the limit.

In vertical tabular format, multiple project names must be enclosed in parentheses.

In vertical tabular format, use empty parentheses () or a dash (-) to indicate each project. Fields cannot be left blank.

Default

None. If no limit is specified for PER_PROJECT or PROJECTS, no limit is enforced on any project.

Example

```
PER_PROJECT=proj1 proj2
```

PER_QUEUE

Syntax

PER_QUEUE=all [**~**]*queue_name* ..

PER_QUEUE

([**-**] | **all** [**~**]*queue_name* ... **)**

Description

A space-separated list of queue names on which limits are enforced. Limits are enforced on jobs submitted to each queue listed.

Do not configure PER_QUEUE and QUEUES limits in the same Limit section.

In horizontal format, use only one PER_QUEUE line per Limit section.

Use the keyword all to configure limits that apply to each queue in a cluster.

Use the not operator (~) to exclude queues from the all specification in the limit. This is useful if you have a large number of queues but only want to exclude a few queues from the limit definition.

In vertical tabular format, multiple queue names must be enclosed in parentheses.

In vertical tabular format, use empty parentheses () or a dash (-) to indicate each queue. Fields cannot be left blank.

Default

None. If no limit is specified for PER_QUEUE or QUEUES, no limit is enforced on any queue.

Example

```
PER_QUEUE=priority night
```

PER_USER

Syntax

PER_USER=all [**~**]*user_name* ... | **all** [**~**]*user_group* ...

PER_USER

([-] | **all** [**~**]*user_name* ... | **all** [**~**]*user_group* ... **)**

Description

A space-separated list of user names or user groups on which limits are enforced. Limits are enforced on each user or individually to each user in the user group listed. If a user group contains a subgroup, the limit also applies to each member in the subgroup recursively.

User names must be valid login names. User group names can be LSF user groups or UNIX and Windows user groups. Note that for LSF and UNIX user groups, the groups must be specified in a UserGroup section in lsb.users first.

Do not configure PER_USER and USERS limits in the same Limit section.

In horizontal format, use only one PER_USER line per Limit section.

Use the keyword all to configure limits that apply to each user in a cluster. If user groups are configured, the limit applies to each member of the user group, not the group as a whole.

Use the not operator (~) to exclude users or user groups from the all specification in the limit. This is useful if you have a large number of users but only want to exclude a few users from the limit definition.

In vertical tabular format, multiple user names must be enclosed in parentheses.

In vertical tabular format, use empty parentheses () or a dash (-) to indicate user or user group member. Fields cannot be left blank.

Default

None. If no limit is specified for PER_USER or USERS, no limit is enforced on any user or user group.

Example

```
PER_USER=user1 user2 ugroup1 ~user3
```

PROJECTS

Syntax

PROJECTS=all [~]*project_name* ...

PROJECTS

([-] | **all** [~]*project_name* ... **)**

Description

A space-separated list of project names on which limits are enforced. Limits are enforced on all projects listed.

To specify a per-project limit, use the PER_PROJECT keyword. Do not configure PROJECTS and PER_PROJECT limits in the same Limit section.

In horizontal format, use only one PROJECTS line per Limit section.

Use the keyword all to configure limits that apply to all projects in a cluster.

Use the not operator (~) to exclude projects from the all specification in the limit. This is useful if you have a large number of projects but only want to exclude a few projects from the limit definition.

In vertical tabular format, multiple project names must be enclosed in parentheses.

In vertical tabular format, use empty parentheses () or a dash (-) to indicate all projects. Fields cannot be left blank.

Default

all (limits are enforced on all projects in the cluster)

Example

```
PROJECTS=projA projB
```

QUEUES

Syntax

QUEUES=all [~]*queue_name* ...

QUEUES

([-] | **all** [~]*queue_name* ... **)**

Description

A space-separated list of queue names on which limits are enforced. Limits are enforced on all queues listed.

The list must contain valid queue names defined in lsb.queues.

To specify a per-queue limit, use the PER_QUEUE keyword. Do not configure QUEUES and PER_QUEUE limits in the same Limit section.

In horizontal format, use only one QUEUES line per Limit section.

Use the keyword all to configure limits that apply to all queues in a cluster.

Use the not operator (~) to exclude queues from the all specification in the limit. This is useful if you have a large number of queues but only want to exclude a few queues from the limit definition.

In vertical tabular format, multiple queue names must be enclosed in parentheses.

In vertical tabular format, use empty parentheses () or a dash (-) to indicate all queues. Fields cannot be left blank.

Default

all (limits are enforced on all queues in the cluster)

Example

```
QUEUES=normal night
```

RESOURCE

Syntax

RESOURCE=[*shared_resource,integer*] [[*shared_resource,integer*] ...]

RESOURCE

([[*shared_resource,integer*] [[*shared_resource,integer*] ...] **)**

Description

Maximum amount of any user-defined shared resource available to consumers.

The RESOURCE keyword is a synonym for the LICENSE keyword. You can use RESOURCE to configure software licenses. You cannot specify RESOURCE and LICENSE in the same Limit section.

In horizontal format, use only one RESOURCE line per Limit section.

In vertical tabular format, resource names must be enclosed in parentheses.

In vertical tabular format, use empty parentheses () or a dash (-) to indicate all queues. Fields cannot be left blank.

Default

None

Examples

```
RESOURCE=[stat_shared, 4]
Begin Limit
RESOURCE                        PER_HOST
([stat_shared, 4])              (all ~hostA)
([dyn_rsrc, 1] [stat_rsrc, 2])  (hostA)
End Limit
```

SLOTS

Syntax

SLOTS=*integer*

SLOTS

- | *integer*

Description

Maximum number of job slots available to resource consumers. Specify a positive integer greater than or equal 0.

With MultiCluster resource lease model, this limit applies only to local hosts being used by the local cluster. The job slot limit for hosts exported to a remote cluster is determined by the host export policy, not by this parameter. The job slot limit for borrowed hosts is determined by the host export policy of the remote cluster.

If JOBS are configured in the Limit section, the most restrictive limit is applied.

If HOSTS are configured in the Limit section, SLOTS is the number of running and suspended jobs on a host. If preemptive scheduling is used, the suspended jobs are not counted as using a job slot.

To fully use the CPU resource on multiprocessor hosts, make the number of job slots equal to or greater than the number of processors.

Use this parameter to prevent a host from being overloaded with too many jobs, and to maximize the throughput of a machine.

Use "!" to make the number of job slots equal to the number of CPUs on a host.

If the number of CPUs in a host changes dynamically, `mbatchd` adjusts the maximum number of job slots per host accordingly. Allow the `mbatchd` up to 10 minutes to get the number of CPUs for a host. During this period the value of SLOTS is 1.

If only QUEUES are configured in the Limit section, SLOTS is the maximum number of job slots available to the listed queues for any hosts, users, or projects.

If only USERS are configured in the Limit section, SLOTS is the maximum number of job slots that the users or user groups can use on any hosts, queues, or projects.

If only HOSTS are configured in the Limit section, SLOTS is the maximum number of job slots that are available to the listed hosts for any users, queues, or projects.

If only PROJECTS are configured in the Limit section, SLOTS is the maximum number of job slots that are available to the listed projects for any users, queues, or hosts.

Use QUEUES or PER_QUEUE, USERS or PER_USER, HOSTS or PER_HOST, and PROJECTS or PER_PROJECT in combination to further limit job slots per processor available to resource consumers.

In horizontal format, use only one SLOTS line per Limit section.

In vertical format, use empty parentheses () or a dash (-) to indicate the default value (no limit). Fields cannot be left blank.

Default

No limit

Example

SL0TS=20

SLOTS_PER_PROCESSOR

Syntax

SLOTS_PER_PROCESSOR=*number*

SLOTS_PER_PROCESSOR

- | *number*

Description

Per processor job slot limit, based on the number of processors on each host affected by the limit.

Maximum number of job slots that each resource consumer can use per processor. This job slot limit is configured per processor so that multiprocessor hosts will automatically run more jobs.

You must also specify PER_HOST and list the hosts that the limit is to be enforced on. The Limit section is ignored if SLOTS_PER_PROCESSOR is specified:

• Without PER_HOST, or
• With HOSTS

In vertical format, use empty parentheses () or a dash (-) to indicate the default value (no limit). Fields cannot be left blank.

To fully use the CPU resource on multiprocessor hosts, make the number of job slots equal to or greater than the number of processors.

Use this parameter to prevent a host from being overloaded with too many jobs, and to maximize the throughput of a machine.

This number can be a fraction such as 0.5, so that it can also serve as a per-CPU limit on multiprocessor machines. This number is rounded up to the nearest integer equal to or greater than the total job slot limits for a host. For example, if SLOTS_PER_PREOCESSOR is 0.5, on a 4-CPU multiprocessor host, users can only use up to 2 job slots at any time. On a single-processor machine, users can use 1 job slot.

Use "!" to make the number of job slots equal to the number of CPUs on a host.

If the number of CPUs in a host changes dynamically, mbatchd adjusts the maximum number of job slots per host accordingly. Allow the mbatchd up to 10 minutes to get the number of CPUs for a host. During this period the number of CPUs is 1.

If only QUEUES and PER_HOST are configured in the Limit section, SLOTS_PER_PROCESSOR is the maximum amount of job slots per processor available to the listed queues for any hosts, users, or projects.

If only USERS and PER_HOST are configured in the Limit section, SLOTS_PER_PROCESSOR is the maximum amount of job slots per processor that the users or user groups can use on any hosts, queues, or projects.

If only PER_HOST is configured in the Limit section, SLOTS_PER_PROCESSOR is the maximum amount of job slots per processor available to the listed hosts for any users, queues, or projects.

If only PROJECTS and PER_HOST are configured in the Limit section, SLOTS_PER_PROCESSOR is the maximum amount of job slots per processor available to the listed projects for any users, queues, or hosts.

Use QUEUES or PER_QUEUE, USERS or PER_USER, PER_HOST, and PROJECTS or PER_PROJECT in combination to further limit job slots per processor available to resource consumers.

Default

No limit

Example

```
SLOTS_PER_PROCESSOR=2
```

SWP

Syntax

SWP=_integer_[**%**]

SWP

- | _integer_[**%**]

Description

Maximum amount of swap space available to resource consumers. Specify a value in MB or a percentage (%) as a positive integer greater than or equal 0. If you specify a percentage, you must also specify PER_HOST and list the hosts that the limit is to be enforced on.

The Limit section is ignored if SWP is specified as a percentage:

• Without PER_HOST, or
• With HOSTS

In horizontal format, use only one SWP line per Limit section.

In vertical format, use empty parentheses () or a dash (-) to indicate the default value (no limit). Fields cannot be left blank.

If only QUEUES are configured in the Limit section, SWP must be an integer value. SWP is the maximum amount of swap space available to the listed queues for any hosts, users, or projects.

If only USERS are configured in the Limit section, SWP must be an integer value. SWP is the maximum amount of swap space that the users or user groups can use on any hosts, queues, or projects.

If only HOSTS are configured in the Limit section, SWP must be an integer value. SWP is the maximum amount of swap space available to the listed hosts for any users, queues, or projects.

If only PROJECTS are configured in the Limit section, SWP must be an integer value. SWP is the maximum amount of swap space available to the listed projects for any users, queues, or hosts.

Use QUEUES or PER_QUEUE, USERS or PER_USER, HOSTS or PER_HOST, and PROJECTS or PER_PROJECT in combination to further limit swap space available to resource consumers.

Default

No limit

Example

```
SWP=60
```

TMP

Syntax

TMP=_integer_[**%**]

TMP

- | _integer_[**%**]

Description

Maximum amount of `tmp` space available to resource consumers. Specify a value in MB or a percentage (%) as a positive integer greater than or equal 0. If you specify a percentage, you must also specify PER_HOST and list the hosts that the limit is to be enforced on.

The Limit section is ignored if TMP is specified as a percentage:

- Without PER_HOST, or
- With HOSTS

In horizontal format, use only one TMP line per Limit section.

In vertical format, use empty parentheses () or a dash (-) to indicate the default value (no limit). Fields cannot be left blank.

If only QUEUES are configured in the Limit section, TMP must be an integer value. TMP is the maximum amount of `tmp` space available to the listed queues for any hosts, users, or projects.

If only USERS are configured in the Limit section, TMP must be an integer value. TMP is the maximum amount of `tmp` space that the users or user groups can use on any hosts, queues, or projects.

If only HOSTS are configured in the Limit section, TMP must be an integer value. TMP is the maximum amount of `tmp` space available to the listed hosts for any users, queues, or projects.

If only PROJECTS are configured in the Limit section, TMP must be an integer value. TMP is the maximum amount of `tmp` space available to the listed projects for any users, queues, or hosts.

Use QUEUES or PER_QUEUE, USERS or PER_USER, HOSTS or PER_HOST, and PROJECTS or PER_PROJECT in combination to further limit `tmp` space available to resource consumers.

Default

No limit

Example

```
TMP=20%
```

USERS

Syntax

USERS=all [**~**]*user_name* ... | **all** [**~**]*user_group* ...

USERS

([**-**] | **all** [**~**]*user_name* ... | **all** [**~**]*user_group* ... **)**

Description

A space-separated list of user names or user groups on which limits are enforced. Limits are enforced on all users or groups listed. Limits apply to a group as a whole.

If a group contains a subgroup, the limit also applies to each member in the subgroup recursively.

User names must be valid login names. User group names can be LSF user groups or UNIX and Windows user groups.

To specify a per-user limit, use the PER_USER keyword. Do not configure USERS and PER_USER limits in the same Limit section.

In horizontal format, use only one USERS line per Limit section.

Use the keyword all to configure limits that apply to all users or user groups in a cluster.

Use the not operator (~) to exclude users or user groups from the all specification in the limit. This is useful if you have a large number of users but only want to exclude a few users or groups from the limit definition.

In vertical format, multiple user names must be enclosed in parentheses.

In vertical format, use empty parentheses () or a dash (-) to indicate all users or groups. Fields cannot be left blank.

Default

all (limits are enforced on all users in the cluster)

Example

```
USERS=user1 user2
```

HostExport section

Defines an export policy for a host or a group of related hosts. Defines how much of each host's resources are exported, and how the resources are distributed among the consumers.

Each export policy is defined in a separate HostExport section, so it is normal to have multiple HostExport sections in lsb. resources.

HostExport section structure

Use empty parentheses ( ) or a dash (-) to specify the default value for an entry. Fields cannot be left blank.

Example HostExport section

```
Begin HostExport PER_HOST= hostA hostB SLOTS= 4 DISTRIBUTION= [cluster1, 1]
[cluster2, 3] MEM= 100 SWAP= 100 End HostExport
```

Parameters

- PER_HOST
- RES_SELECT
- NHOSTS
- DISTRIBUTION
- MEM
- SLOTS
- SWAP
- TYPE

PER_HOST

Syntax

PER_HOST=*host_name...*

Description

Required when exporting special hosts.

Determines which hosts to export. Specify one or more LSF hosts by name. Separate names by space.

RES_SELECT

Syntax

RES_SELECT=*res_req*

Description

Required when exporting workstations.

Determines which hosts to export. Specify the selection part of the resource requirement string (without quotes or parentheses), and LSF will automatically select hosts that meet the specified criteria. For this parameter, if you do not specify the required host type, the default is type==any.

When LSF_STRICT_RESREQ=Y is configured in lsf.conf, resource requirement strings in select sections must conform to a more strict syntax. The strict resource requirement syntax only applies to the select section. It does not apply to the other resource requirement sections (order, rusage, same, span, or cu). When LSF_STRICT_RESREQ=Y in lsf.conf, LSF rejects resource requirement strings where an rusage section contains a non-consumable resource.

The criteria is only evaluated once, when a host is exported.

NHOSTS

Syntax

NHOSTS=*integer*

Description

Required when exporting workstations.

Maximum number of hosts to export. If there are not this many hosts meeting the selection criteria, LSF exports as many as it can.

DISTRIBUTION

Syntax

DISTRIBUTION=([*cluster_name*, *number_shares*]...**)**

Description

Required. Specifies how the exported resources are distributed among consumer clusters.

The syntax for the distribution list is a series of share assignments. The syntax of each share assignment is the cluster name, a comma, and the number of shares, all enclosed in square brackets, as shown. Use a space to separate multiple share assignments. Enclose the full distribution list in a set of round brackets.

cluster_name

Specify the name of a remote cluster that will be allowed to use the exported resources. If you specify a local cluster, the assignment is ignored.

number_shares

Specify a positive integer representing the number of shares of exported resources assigned to the cluster.

The number of shares assigned to a cluster is only meaningful when you compare it to the number assigned to other clusters, or to the total number. The total number of shares is just the sum of all the shares assigned in each share assignment.

MEM

Syntax

MEM=*megabytes*

Description

Used when exporting special hosts. Specify the amount of memory to export on each host, in MB.

Default

- (provider and consumer clusters compete for available memory)

SLOTS

Syntax

SLOTS=*integer*

Description

Required when exporting special hosts. Specify the number of job slots to export on each host.

To avoid overloading a partially exported host, you can reduce the number of job slots in the configuration of the local cluster.

SWAP

Syntax

SWAP=*megabytes*

Description

Used when exporting special hosts. Specify the amount of swap space to export on each host, in MB.

Default

- (provider and consumer clusters compete for available swap space)

TYPE

Syntax

TYPE=shared

Description

Changes the lease type from exclusive to shared.

If you export special hosts with a shared lease (using PER_HOST), you cannot specify multiple consumer clusters in the distribution policy.

Default

Undefined (the lease type is exclusive; exported resources are never available to the provider cluster)

SharedResourceExport section

Optional. Requires HostExport section. Defines an export policy for a shared resource. Defines how much of the shared resource is exported, and the distribution among the consumers.

The shared resource must be available on hosts defined in the HostExport sections.

SharedResourceExport section structure

All parameters are required.

Example SharedResourceExport section

```
Begin SharedResourceExport
NAME= AppLicense
NINSTANCES= 10
DISTRIBUTION= ([C1, 30] [C2, 70])
End SharedResourceExport
```

Parameters

- NAME
- NINSTANCES
- DISTRIBUTION

NAME

Syntax

NAME=*shared_resource_name*

Description

Shared resource to export. This resource must be available on the hosts that are exported to the specified clusters; you cannot export resources without hosts.

NINSTANCES

Syntax

NINSTANCES=*integer*

Description

Maximum quantity of shared resource to export. If the total number available is less than the requested amount, LSF exports all that are available.

DISTRIBUTION

Syntax

DISTRIBUTION=([*cluster_name*, *number_shares*]...**)**

Description

Specifies how the exported resources are distributed among consumer clusters.

The syntax for the distribution list is a series of share assignments. The syntax of each share assignment is the cluster name, a comma, and the number of shares, all enclosed in square brackets, as shown. Use a space to separate multiple share assignments. Enclose the full distribution list in a set of round brackets.

cluster_name

Specify the name of a cluster allowed to use the exported resources.

number_shares

Specify a positive integer representing the number of shares of exported resources assigned to the cluster.

The number of shares assigned to a cluster is only meaningful when you compare it to the number assigned to other clusters, or to the total number. The total number of shares is the sum of all the shares assigned in each share assignment.

ResourceReservation section

By default, only LSF administrators or root can add or delete advance reservations.

The ResourceReservation section defines an advance reservation policy. It specifies:

*   Users or user groups that can create reservations
*   Hosts that can be used for the reservation
*   Time window when reservations can be created

Each advance reservation policy is defined in a separate ResourceReservation section, so it is normal to have multiple ResourceReservation sections in lsb.resources.

Example ResourceReservation section

Only user1 and user2 can make advance reservations on hostA and hostB. The reservation time window is between 8:00 a.m. and 6:00 p.m. every day:

```
Begin ResourceReservation
NAME         = dayPolicy
USERS        = user1 user2      # optional
HOSTS        = hostA hostB      # optional
TIME_WINDOW  = 8:00-18:00            # weekly recurring reservation
End ResourceReservation
```

user1 can add the following reservation for user user2 to use on hostA every Friday between 9:00 a.m. and 11:00 a.m.:

```
% user1@hostB> brsvadd -m "hostA" -n 1 -u "user2" -t "5:9:0-5:11:0"
Reservation "user2#2" is created
```

Users can only delete reservations they created themselves. In the example, only user user1 can delete the reservation; user2 cannot. Administrators can delete any reservations created by users.

Parameters

*   HOSTS
*   NAME
*   TIME_WINDOW
*   USERS

HOSTS

Syntax

HOSTS=[~]*host_name* | [~]*host_group* | **all** | **allremote** | **all@***cluster_name* ...

Description

A space-separated list of hosts, host groups defined in lsb.hosts on which administrators or users specified in the USERS parameter can create advance reservations.

The hosts can be local to the cluster or hosts leased from remote clusters.

If a group contains a subgroup, the reservation configuration applies to each member in the subgroup recursively.

Use the keyword all to configure reservation policies that apply to all local hosts in a cluster not explicitly excluded. This is useful if you have a large cluster but you want to use the not operator (~) to exclude a few hosts from the list of hosts where reservations can be created.

Use the keyword allremote to specify all hosts borrowed from all remote clusters.

Tip:

You cannot specify host groups or host partitions that contain the allremote keyword.

Use all@*cluster_name* to specify the group of all hosts borrowed from one remote cluster. You cannot specify a host group or partition that includes remote resources.

With MultiCluster resource leasing model, the not operator (~) can be used to exclude local hosts or host groups. You cannot use the not operator (~) with remote hosts.

Examples

```
HOSTS=hgroup1 ~hostA hostB hostC
```

Advance reservations can be created on hostB, hostC, and all hosts in hgroup1 except for hostA.

```
HOSTS=all ~group2 ~hostA
```

Advance reservations can be created on all hosts in the cluster, except for hostA and the hosts in group2.

Default

all allremote (users can create reservations on all server hosts in the local cluster, and all leased hosts in a remote cluster).

NAME
Syntax

NAME=*text*

Description

Required. Name of the ResourceReservation section

Specify any ASCII string 40 characters or less. You can use letters, digits, underscores (_) or dashes (-). You cannot use blank spaces.

Example

```
NAME=reservation1
```

Default

None. You must provide a name for the ResourceReservation section.

TIME_WINDOW

Syntax

TIME_WINDOW=*time_window* ...

Description

Optional. Time window for users to create advance reservations. The time for reservations that users create must fall within this time window.

Use the same format for *time_window* as the recurring reservation option (-t) of brsvadd. To specify a time window, specify two time values separated by a hyphen (-), with no space in between:

```
time_window = begin_time-end_time
```

Time format

Times are specified in the format:

```
[day:]hour[:minute]
```

where all fields are numbers with the following ranges:

- *day of the week*: 0-6 (0 is Sunday)
- *hour*: 0-23
- *minute*: 0-59

Specify a time window one of the following ways:

- *hour-hour*
- *hour.minute-hour.minute*
- *day.hour.minute-day.hour.minute*

The default value for minute is 0 (on the hour); the default value for day is every day of the week.

You must specify at least the hour. Day of the week and minute are optional. Both the start time and end time values must use the same syntax. If you do not specify a minute, LSF assumes the first minute of the hour (:00). If you do not specify a day, LSF assumes every day of the week. If you do specify the day, you must also specify the minute.

You can specify multiple time windows, but they cannot overlap. For example:

```
timeWindow(8:00-14:00  18:00-22:00)
```

is correct, but

```
timeWindow(8:00-14:00  11:00-15:00)
```

is not valid.

Example

```
TIME_WINDOW=8:00-14:00
```

Users can create advance reservations with begin time (brsvadd -b), end time (brsvadd -e), or time window (brsvadd -t) on any day between 8:00 a.m. and 2:00 p.m.

Default

Undefined (any time)

USERS

Syntax

USERS=[**~**]*user_name* | [**~**]*user_group* ... | **all**

Description

A space-separated list of user names or user groups who are allowed to create advance reservations. Administrators, root, and all users or groups listed can create reservations.

If a group contains a subgroup, the reservation policy applies to each member in the subgroup recursively.

User names must be valid login names. User group names can be LSF user groups or UNIX and Windows user groups.

Use the keyword all to configure reservation policies that apply to all users or user groups in a cluster. This is useful if you have a large number of users but you want to exclude a few users or groups from the reservation policy.

Use the not operator (~) to exclude users or user groups from the list of users who can create reservations.

> **Caution:**
>
> The not operator does not exclude LSF administrators from the policy.

Example

```
USERS=user1 user2
```

Default

all (all users in the cluster can create reservations)

ReservationUsage section

To enable greater flexibility for reserving numeric resources that are reserved by jobs, configure the ReservationUsage section in lsb.resources to reserve resources like license tokens per resource as PER_JOB, PER_SLOT, or PER_HOST. For example:

Example ReservationUsage section

Begin ReservationUsage		
RESOURCE	METHOD	RESERVE
licenseX	PER_JOB	Y
licenseY	PER_HOST	N
licenseZ	PER_SLOT	N
End ReservationUsage		

Parameters

- RESOURCE
- METHOD
- RESERVE

RESOURCE

The name of the resource to be reserved. User-defined numeric resources can be reserved, but only if they are shared (they are not specific to one host).

The following built-in resources can be configured in the ReservationUsage section and reserved:

- mem
- tmp
- swp

Any custom resource can also be reserved if it is shared (defined in the Resource section of lsf.shared) or host based (listed in the Host section of the lsf.cluster file in the resource column).

METHOD

The resource reservation method. One of:

- PER_JOB
- PER_HOST
- PER_SLOT

The cluster-wide RESOURCE_RESERVE_PER_SLOT parameter in lsb.params is obsolete.

RESOURCE_RESERVE_PER_SLOT parameter still controls resources not configured in lsb.resources. Resources not reserved in lsb.resources are reserved per job.

PER_HOST reservation means that for the parallel job, LSF reserves one instance of a for each host. For example, some application licenses are charged only once no matter how many applications are running provided those applications are running on the same host under the same user.

Use no method ("-") when setting mem, swp, or tmp as RESERVE=Y.

RESERVE

Reserves the resource for pending jobs that are waiting for another resource to become available.

For example, job A requires resources X, Y, and Z to run, but resource Z is a high demand or scarce resource. This job pends until Z is available. In the meantime, other jobs requiring only X and Y resources run. If X and Y are set as reservable resources (the RESERVE parameter is set to "Y"), as soon as Z resource is available, job A runs. If they are not, job A may never be able to run because all resources are never available at the same time.

Restriction:

Only the following built-in resources can be defined as reservable:

- mem
- swp
- tmp

Use no method ("-") when setting mem, swp, or tmp as RESERVE=Y.

When submitting a job, the queue must have RESOURCE_RESERVE defined.

Backfill of the reservable resources is also supported when you submit a job with reservable resources to a queue with BACKFILL defined.

Valid values are Y and N. If not specified, resources are not reserved.

Assumptions and limitations

- Per-resource configuration defines resource usage for individual resources, but it does not change any existing resource limit behavior (PER_JOB, PER_SLOT).
- In a MultiCluster environment, you should configure resource usage in the scheduling cluster (submission cluster in lease model or receiving cluster in job forward model).

Automatic time-based configuration

Variable configuration is used to automatically change LSF configuration based on time windows. You define automatic configuration changes in lsb. resources by using if-else constructs and time expressions. After you change the files, reconfigure the cluster with the badmin reconfig command.

The expressions are evaluated by LSF every 10 minutes based on mbatchd start time. When an expression evaluates true, LSF dynamically changes the configuration based on the associated configuration statements. Reconfiguration is done in real time without restarting mbatchd, providing continuous system availability.

Example

```
# limit usage of hosts in 'license1' group and time
# based configuration
# - 10 jobs can run from normal queue
# - any number can run from short queue between 18:30
#   and 19:30
#   all other hours you are limited to 100 slots in the
#   short queue
# - each other queue can run 30 jobs

Begin Limit
PER_QUEUE               HOSTS       SLOTS     # Example
normal                  license1    10
# if time(18:30-19:30)
short                   license1    -
#else
short                   license1    100
#endif
(all ~normal ~short)    license1    30
End Limit
```

lsb.serviceclasses

The `lsb.serviceclasses` file defines the service-level agreements (SLAs) in an LSF cluster as *service classes*, which define the properties of the SLA.

This file is optional.

You can configure as many service class sections as you need.

Use `bsla` to display the properties of service classes configured in `lsb.serviceclasses` and dynamic information about the state of each configured service class.

By default, `lsb.serviceclasses` is installed in LSB_CONFDIR/*cluster_name*/`configdir`.

Changing lsb.serviceclasses configuration

After making any changes to `lsb.serviceclasses`, run `badmin reconfig` to reconfigure mbatchd.

lsb.serviceclasses structure

Each service class definition begins with the line Begin ServiceClass and ends with the line End ServiceClass.

Syntax

```
Begin ServiceClass
NAME            = string
PRIORITY        = integer
GOALS           = [throughput | velocity | deadline] [\
                   [throughput | velocity | deadline] ...]
CONTROL_ACTION = VIOLATION_PERIOD[minutes] CMD [action]
USER_GROUP      = all | [user_name] [user_group] ...
DESCRIPTION     = text
End ServiceClass
```

You must specify:

- Service class name
- Goals
- Priority

To configure EGO-enabled SLA scheduling, you must specifiy an existing EGO consumer name to allow the SLA to get host allocations from EGO.

All other parameters are optional.

Example

```
Begin ServiceClass
NAME=Uclulet
PRIORITY=20
GOALS=[DEADLINE timeWindow (8:30-16:00)]
DESCRIPTION="working hours"
End ServiceClass
```

Parameters

- CONSUMER
- CONTROL_ACTION
- DESCRIPTION
- EGO_RES_REQ
- GOALS
- MAX_HOST_IDLE_TIME
- NAME
- PRIORITY
- USER_GROUP

CONSUMER

Syntax

CONSUMER=_ego_consumer_name_

Description

For EGO-enabled SLA service classes, the name of the EGO consumer from which hosts are allocated to the SLA. This parameter is not mandatory, but must be configured for the SLA to receive hosts from EGO.

> **Important:**
>
> CONSUMER must specify the name of a valid consumer in EGO.
> If a default SLA is configured with
> ENABLE_DEFAULT_EGO_SLA in lsb.params, all services
> classes configured in lsb.servicecl asses must specifiy a
> consumer name.

Default

None

CONTROL_ACTION

Syntax

CONTROL_ACTION=VIOLATION_PERIOD[_minutes_] CMD [_action_]

Description

Optional. Configures a control action to be run if the SLA goal is delayed for a specified number of minutes.

If the SLA goal is delayed for longer than VIOLATION_PERIOD, the action specified by CMD is invoked. The violation period is reset and if the SLA is still active when the violation period expires again, the action runs again. If the SLA has multiple active goals that are in violation, the action is run for each of them.

Example

```
CONTROL_ACTION=VIOLATION_PERIOD[10] CMD [echo `date`: SLA is in violation
>> !/tmp/sla_violation.log]
```

Default

None

DESCRIPTION

Syntax

DESCRIPTION=*text*

Description

Optional. Description of the service class. Use bsla to display the description text.

This description should clearly describe the features of the service class to help users select the proper service class for their jobs.

The text can include any characters, including white space. The text can be extended to multiple lines by ending the preceding line with a backslash (\).

Default

None

EGO_RES_REQ

Syntax

EGO_RES_REQ=*res_req*

Description

For EGO-enabled SLA service classes, the EGO resource requirement that specifies the characteristics of the hosts that EGO will assign to the SLA.

Must be a valid EGO resource requirement. The EGO resource requirement string supports the select section, but the format is different from LSF resource requirements.

Example

```
EGO_RES_REQ=select(linux && maxmem > 100)
```

Default

None

GOALS

Syntax

GOALS=[*throughput* | *velocity* | *deadline*] [\

[*throughput* | *velocity* | *deadline*] ...]

Description

Required. Defines the service-level goals for the service class. A service class can have more than one goal, each active at different times of the day and days of the week. Outside of the

time window, the SLA is inactive and jobs are scheduled as if no service class is defined. LSF does not enforce any service-level goal for an inactive SLA.

The time windows of multiple service-level goals can overlap. In this case, the largest number of jobs is run.

An active SLA can have a status of On time if it is meeting the goal, and a status Delayed, if it is missing its goals.

A service-level goal defines:

throughput — expressed as *finished* jobs per hour and an optional time window when the goal is active. *throughput* has the form:

```
GOALS=[THROUGHPUT num_jobs timeWindow [(time_window)]]
```

If no time window is configured, THROUGHPUT can be the only goal in the service class. The service class is always active, and bsla displays ACTIVE WINDOW: Always Open.

velocity — expressed as *concurrently* running jobs and an optional time window when the goal is active. *velocity* has the form:

```
GOALS=[VELOCITY num_jobs timeWindow [(time_window)]]
```

If no time window is configured, VELOCITY can be the only goal in the service class. The service class is always active, and bsla displays ACTIVE WINDOW: Always Open.

deadline — indicates that all jobs in the service class should complete by the end of the specified time window. The time window is required for a deadline goal. *deadline* has the form:

```
GOALS=[DEADLINE timeWindow (time_window)]
```

Restriction:

EGO-enabled SLA service classes only support velocity goals. Deadline and throughput goals are not supported. The configured configured velocity value for EGO-enabled SLA service classes is considered to be a *minimum* number of jobs that should be in run state from the SLA

Time window format

The time window of an SLA goal has the standard form:

```
begin_time-end_time
```

Times are specified in the format:

```
[day:]hour[:minute]
```

where all fields are numbers with the following ranges:

- *day of the week*: 0-6 (0 is Sunday)
- *hour*: 0-23
- *minute*: 0-59

Specify a time window one of the following ways:

- *hour-hour*
- *hour*:*minute*-*hour*:*minute*
- *day*:*hour*:*minute*-*day*:*hour*:*minute*

The default value for minute is 0 (on the hour); the default value for day is every day of the week.

You must specify at least the hour. Day of the week and minute are optional. Both the start time and end time values must use the same syntax. If you do not specify a minute, LSF assumes the first minute of the hour (: 00). If you do not specify a day, LSF assumes every day of the week. If you do specify the day, you must also specify the minute.

You can specify multiple time windows, but they cannot overlap. For example:

```
timeWindow(8:00-14:00 18:00-22:00)
```

is correct, but

```
timeWindow(8:00-14:00 11:00-15:00)
```

is not valid.

Tip:

To configure a time window that is always open, use the timeWindow keyword with empty parentheses.

Examples

```
GOALS=[THROUGHPUT 2 timeWindow ()]
```
```
GOALS=[THROUGHPUT 10 timeWindow (8:30-16:30)]
```
```
GOALS=[VELOCITY 5 timeWindow ()]
```
```
GOALS=[DEADLINE timeWindow (16:30-8:30)]\
        [VELOCITY 10 timeWindow (8:30-16:30)]
```

MAX_HOST_IDLE_TIME

Syntax

MAX_HOST_IDLE_TIME=*seconds*

Description

For EGO-enabled SLA service classes, number of seconds that the SLA will hold its idle hosts before LSF releases them to EGO. Each SLA can configure a different idle time. Do not set this parameter to a small value, or LSF may release hosts too quickly.

Default

120 seconds

NAME

Syntax

NAME=*string*

Description

Required. A unique name that identifies the service class.

Specify any ASCII string 60 characters or less. You can use letters, digits, underscores (_) or dashes (-). You cannot use blank spaces.

Important:

The name you use cannot be the same as an existing host partition, user group name, or fairshare queue name.

Example

```
NAME=Tofino
```

Default

None. You must provide a unique name for the service class.

PRIORITY

Syntax

PRIORITY=*integer*

Description

Required. The service class priority. A higher value indicates a higher priority, relative to other service classes. Similar to queue priority, service classes access the cluster resources in priority order.

LSF schedules jobs from one service class at a time, starting with the highest-priority service class. If multiple service classes have the same priority, LSF runs all the jobs from these service classes in first-come, first-served order.

Service class priority in LSF is completely independent of the UNIX scheduler's priority system for time-sharing processes. In LSF, the NICE parameter is used to set the UNIX time-sharing priority for batch jobs.

Default

1 (lowest possible priority)

USER_GROUP

Syntax

USER_GROUP=all | [*user_name*] [*user_group*] ...

Description

Optional. A space-separated list of user names or user groups who can submit jobs to the service class. Administrators, root, and all users or groups listed can use the service class.

Use the reserved word all to specify all LSF users. LSF cluster administrators are automatically included in the list of users, so LSF cluster administrators can submit jobs to any service class, or switch any user's jobs into this service class, even if they are not listed.

If user groups are specified in lsb.users, each user in the group can submit jobs to this service class. If a group contains a subgroup, the service class policy applies to each member in the subgroup recursively. If the group can define fairshare among its members, the SLA defined by the service class enforces the fairshare policy among the users of the SLA.

User names must be valid login names. User group names can be LSF user groups (in lsb.users) or UNIX and Windows user groups.

Example

USER_GROUP=user1 user2 ugroup1

Default

all (all users in the cluster can submit jobs to the service class)

Examples

- The service class UcluIet defines one deadline goal that is active during working hours between 8:30 AM and 4:00 PM. All jobs in the service class should complete by the end of the specified time window. Outside of this time window, the SLA is inactive and jobs are scheduled without any goal being enforced:

  Begin ServiceClass

  NAME=Uclulet

  PRIORITY=20

  GOALS=[DEADLINE timeWindow (8:30-16:00)]

  DESCRIPTION="working hours"

  End ServiceClass

- The service class Nanaimo defines a deadline goal that is active during the weekends and at nights.

  Begin ServiceClass

  NAME=Nanaimo

  PRIORITY=20

  GOALS=[DEADLINE timeWindow (5:18:00-1:8:30 20:00-8:30)]

  DESCRIPTION="weekend nighttime regression tests"

  End ServiceClass

- The service class Inuvik defines a throughput goal of 6 jobs per hour that is always active:

  Begin ServiceClass

  NAME=Inuvik

  PRIORITY=20

  GOALS=[THROUGHPUT 6 timeWindow ()]

  DESCRIPTION="constant throughput"

  End ServiceClass

- The service class Tofino defines two velocity goals in a 24 hour period. The first goal is to have a maximum of 10 concurrently running jobs during business hours (9:00 a.m. to 5:00 p.m). The second goal is a maximum of 30 concurrently running jobs during off-hours (5:30 p.m. to 8:30 a.m.)

  Begin ServiceClass

  NAME=Tofino

  PRIORITY=20

  GOALS=[VELOCITY 10 timeWindow (9:00-17:00)] \

      [VELOCITY 30 timeWindow (17:30-8:30)]

  DESCRIPTION="day and night velocity"

  End ServiceClass

- The service class Kyuquot defines a velocity goal that is active during working hours (9:00 a.m. to 5:30 p.m.) and a deadline goal that is active during off-hours (5:30 p.m. to 9:00 a.m.) Only users user1 and user2 can submit jobs to this service class.

```
Begin ServiceClass
NAME=Kyuquot
PRIORITY=23
USER_GROUP=user1 user2
GOALS=[VELOCITY 8 timeWindow (9:00-17:30)] \
       [DEADLINE timeWindow (17:30-9:00)]
DESCRIPTION="Daytime/Nighttime SLA"
End ServiceClass
```

- The service class Tevere defines a combination similar to Kyuquot, but with a deadline goal that takes effect overnight and on weekends. During the working hours in weekdays the velocity goal favors a mix of short and medium jobs.

```
Begin ServiceClass
NAME=Tevere
PRIORITY=20
GOALS=[VELOCITY 100 timeWindow (9:00-17:00)] \
       [DEADLINE timeWindow (17:30-8:30 5:17:30-1:8:30)]
DESCRIPTION="nine to five"
End ServiceClass
```

lsb.users

The lsb.users file is used to configure user groups, hierarchical fairshare for users and user groups, and job slot limits for users and user groups. It is also used to configure account mappings in a MultiCluster environment.

This file is optional.

The lsb.users file is stored in the directory LSB_CONFDIR/*cluster_name*/configdir, where LSB_CONFDIR is defined in lsf.conf.

Changing lsb.users configuration

After making any changes to lsb.users, run badmin reconfig to reconfigure mbatchd.

UserGroup section

Optional. Defines user groups.

The name of the user group can be used in other user group and queue definitions, as well as on the command line. Specifying the name of a user group in the GROUP_MEMBER section has exactly the same effect as listing the names of all users in the group.

The total number of user groups cannot be more than 1024.

Structure

The first line consists of two mandatory keywords, GROUP_NAME and GROUP_MEMBER. The USER_SHARES and GROUP_ADMIN keywords are optional. Subsequent lines name a group and list its membership and optionally its share assignments and administrator.

Each line must contain one entry for each keyword. Use empty parentheses () or a dash - to specify the default value for an entry.

Restriction:

If specifying a specific user name for a user group, that entry must precede all user groups.

Example of a UserGroup section

```
Begin UserGroup
GROUP_NAME GROUP_MEMBER GROUP_ADMIN
groupA       (user1 user2 user3 user4) (user5)
groupB       (groupA user5) (groupA)
groupC       (!) ()
End UserGroup
Begin UserGroup
GROUP_NAME    GROUP_MEMBER            USER_SHARES
groupB        (user1 user2)           ()
groupC        (user3 user4)           ([User3, 3] [User4, 4])
groupA        (GroupB GroupC user5)   ([User5, 1] [default, 10])
End UserGroup
```

GROUP_NAME

An alphanumeric string representing the user group name. You cannot use the reserved name all or a "/" in a group name.

GROUP_MEMBER

A list of user names or user group names that belong to the group, enclosed in parentheses and separated by spaces.

User and user group names can appear on multiple lines because users can belong to multiple groups.

Note:

When a user belongs to more than one group, any of the administrators specified for any of the groups the user belongs to can control that users' jobs. Limit administrative control by submitting jobs with the -G option, specifying which user group the job is submitted with.

User groups may be defined recursively but must not create a loop.

Syntax

(*user_name* | *user_group* ...) | (all) | (!)

Specify the following, all enclosed in parentheses:

user_name | *user_group*

User and user group names, separated by spaces. User names must be valid login names. To specify a Windows user account, include the domain name in uppercase letters (*DOMAIN_NAME\user_name*).

User group names can be LSF user groups defined previously in this section, or UNIX and Windows user groups. To specify a Windows user group, include the domain name in uppercase letters (*DOMAIN_NAME\user_group*).

all

The reserved name all specifies all users in the cluster.

!

An exclamation mark (!) indicates an externally-defined user group, which the egroup executable retrieves.

GROUP_ADMIN

User group administrators are a list of user names or user group names that administer the jobs of the group members, enclosed in parentheses and separated by spaces.

A user group administrator is allowed to control any jobs of the members of the user group they administer. A user group administrator can also resume jobs stopped by the LSF administrator or queue administrator if the job belongs to a member of their user group.

A user group administrator has privileges equivalent to those of a job owner. A user group administrator can control any job belonging to member users of the group they administer.

> **Restriction:**
>
> Unlike a job owner, a user group administrator cannot run
> `brestart` and `bread -a` *data_file*.

To manage security concerns, you cannot specify the keyword ALL for any user group administrators.

Syntax

(*user_name* | *user_group* ...)

Specify the following, all enclosed in parentheses:

user_name | *user_group*

User and user group names, separated by spaces. User names must be valid login names. To specify a Windows user account, include the domain name in uppercase letters (*DOMAIN_NAME\user_name*).

User group names can be LSF user groups defined previously in this section, or UNIX and Windows user groups. To specify a Windows user group, include the domain name in uppercase letters (*DOMAIN_NAME\user_group*).

Valid values

- You can specify a user group as an administrator for another user group. In that case, all members of the first user group become administrators for the second user group.
- You can also specify that all users of a group are also administrators of that same group.
- Users can be administrators for more than one user group at the same time.

> **Note:**
>
> When a user belongs to more than one group, any of the administrators specified for any of the groups the user belongs to can control that users' jobs.

Restrictions

- Wildcard and special characters are not supported (for example: *, !, $, #, &, ~)
- The reserved keywords ALL, others, default, allremote are not supported.

  User groups with members defined with the keyword ALL are also not allowed as a user group administrator.
- User groups and user groups administrator definitions cannot be recursive or create a loop.

USER_SHARES

Optional. Enables hierarchical fairshare and defines a share tree for users and user groups.

By default, when resources are assigned collectively to a group, the group members compete for the resources according to FCFS scheduling. You can use hierarchical fairshare to further divide the shares among the group members.

Syntax

([*user*, *number_shares*])

Specify the arguments as follows:

- Enclose the list in parentheses, even if you do not specify any user share assignments.
- Enclose each user share assignment in square brackets, as shown.
- Separate the list of share assignments with a space.
- *user*—Specify users or user groups. You can assign the shares to:

    - A single user (specify *user_name*). To specify a Windows user account, include the domain name in uppercase letters (*DOMAIN_NAME\user_name*).
    - Users in a group, individually (specify *group_name@*) or collectively (specify *group_name*). To specify a Windows user group, include the domain name in uppercase letters (*DOMAIN_NAME\group_name*).
    - Users not included in any other share assignment, individually (specify the keyword default@) or collectively (specify the keyword default).

    Note:

    By default, when resources are assigned collectively to a group, the group members compete for the resources on a first-come, first-served (FCFS) basis. You can use hierarchical fairshare to further divide the shares among the group members. When resources are assigned to members of a group individually, the share assignment is recursive. Members of the group and of all subgroups always compete for the resources according to FCFS scheduling, regardless of hierarchical fairshare policies.

- *number_shares*—Specify a positive integer representing the number of shares of the cluster resources assigned to the user. The number of shares assigned to each user is only meaningful when you compare it to the shares assigned to other users or to the total number of shares. The total number of shares is just the sum of all the shares assigned in each share assignment.

User section

Optional. If this section is not defined, all users and user groups can run an unlimited number of jobs in the cluster.

This section defines the maximum number of jobs a user or user group can run concurrently in the cluster. This is to avoid situations in which a user occupies all or most of the system resources while other users' jobs are waiting.

Structure

Three fields are mandatory: USER_NAME, MAX_JOBS, JL/P.

MAX_PEND_JOBS is optional.

You must specify a dash (-) to indicate the default value (unlimited) if a user or user group is specified. Fields cannot be left blank.

Example of a User section

```
Begin User
```

USER_NAME	MAX_JOBS	JL/P	MAX_PEND_JOBS
user1	10	-	1000
user2	4	-	-
user3	-	-	-
groupA	10	1	100000
groupA@	-	1	100
groupC	-	-	500
default	6	1	10

```
End User
```

USER_NAME

User or user group for which job slot limits are defined.

Use the reserved user name default to specify a job slot limit that applies to each user and user group not explicitly named. Since the limit specified with the keyword default applies to user groups also, make sure you select a limit that is high enough, or explicitly define limits for user groups.

User group names can be the LSF user groups defined previously, and/or UNIX and Windows user groups. To specify a Windows user account or user group, include the domain name in uppercase letters (*DOMAIN_NAME\user_name* or *DOMAIN_NAME\user_group*).

Job slot limits apply to a group as a whole. Append the at sign (@) to a group name to make the job slot limits apply individually to each user in the group. If a group contains a subgroup, the job slot limit also applies to each member in the subgroup recursively.

If the group contains the keyword all in the user list, the at sign (@) has no effect. To specify job slot limits for each user in a user group containing all, use the keyword default.

MAX_JOBS

Per-user or per-group job slot limit for the cluster. Total number of job slots that each user or user group can use in the cluster.

Note:

If a group contains the keyword all as a member, all users and user groups are included in the group. The per-group job slot limit set for the group applies to the group as a whole, limiting the entire cluster even when ENFORCE_ONE_UG_LIMIT is set in lsb.params.

JL/P

Per processor job slot limit per user or user group.

Total number of job slots that each user or user group can use per processor. This job slot limit is configured per processor so that multiprocessor hosts will automatically run more jobs.

This number can be a fraction such as 0.5, so that it can also serve as a per-host limit. This number is rounded up to the nearest integer equal to or greater than the total job slot limits

for a host. For example, if JL/P is 0.5, on a 4-CPU multiprocessor host, the user can only use up to 2 job slots at any time. On a uniprocessor machine, the user can use 1 job slot.

MAX_PEND_JOBS

Per-user or per-group pending job limit. This is the total number of pending job slots that each user or user group can have in the system. If a user is a member of multiple user groups, the user's pending jobs are counted towards the pending job limits of all groups from which the user has membership.

If ENFORCE_ONE_UG_LIMITS is set to Y in lsb.params and you submit a job while specifying a user group, only the limits for that user group (or any parent user group) apply to the job even if there are overlapping user group members.

UserMap section

Optional. Used only in a MultiCluster environment with a non-uniform user name space. Defines system-level cross-cluster account mapping for users and user groups, which allows users to submit a job from a local host and run the job as a different user on a remote host. Both the local and remote clusters must have corresponding user account mappings configured.

Structure

The following three fields are all required:

- LOCAL
- REMOTE
- DIRECTION

LOCAL

A list of users or user groups in the local cluster. To specify a Windows user account or user group, include the domain name in uppercase letters (*DOMAIN_NAME \user_name* or *DOMAIN_NAME\user_group*). Separate multiple user names by a space and enclose the list in parentheses ( ):

```
(user4 user6)
```

REMOTE

A list of remote users or user groups in the form *user_name@cluster_name* or *user_group@cluster_name*. To specify a Windows user account or user group, include the domain name in uppercase letters (*DOMAIN_NAME\user_name@cluster_name* or *DOMAIN_NAME\user_group@cluster_name*). Separate multiple user names by a space and enclose the list in parentheses ( ):

```
(user4@cluster2 user6@cluster2)
```

DIRECTION

Specifies whether the user account runs jobs locally or remotely. Both directions must be configured on the local and remote clusters.

- The export keyword configures local users/groups to run jobs as remote users/ groups.

- The import keyword configures remote users/groups to run jobs as local users/ groups.

Example of a UserMap section

On cluster1:

```
Begin UserMap
LOCAL      REMOTE                          DIRECTION
user1      user2@cluster2                  export
user3      user6@cluster2      export
End UserMap
```

On cluster2:

```
Begin UserMap
LOCAL      REMOTE                          DIRECTION
user2      user1@cluster1                  import
user6      user3@cluster1                  import
End UserMap
```

Cluster1 configures user1 to run jobs as user2 and user3 to run jobs as user6.

Cluster2 configures user1 to run jobs as user2 and user3 to run jobs as user6.

Automatic time-based configuration

Variable configuration is used to automatically change LSF configuration based on time windows. You define automatic configuration changes in lsb.users by using if-else constructs and time expressions. After you change the files, reconfigure the cluster with the badmin reconfig command.

The expressions are evaluated by LSF every 10 minutes based on mbatchd start time. When an expression evaluates true, LSF dynamically changes the configuration based on the associated configuration statements. Reconfiguration is done in real time without restarting mbatchd, providing continuous system availability.

Example

From 12 - 1 p.m. daily, user smith has 10 job slots, but during other hours, user has only 5 job slots.

```
Begin User
USER_NAME   MAX_JOBS   JL/P
#if time (12-13)
smith       10         -
#else
smith       5          -
default     1          -
#endif
End User
```

lsf.acct

The `lsf.acct` file is the LSF task log file.

The LSF Remote Execution Server, RES (see res(8)), generates a record for each task completion or failure. If the RES task logging is turned on (see `lsadmin`(8)), it appends the record to the task log file `lsf.acct.<host_name>`.

lsf.acct structure

The task log file is an ASCII file with one task record per line. The fields of each record are separated by blanks. The location of the file is determined by the LSF_RES_ACCTDIR variable defined in `lsf.conf`. If this variable is not defined, or the RES cannot access the log directory, the log file is created in `/tmp` instead.

Fields

The fields in a task record are ordered in the following sequence:

pid (%d)

> Process ID for the remote task

userName (%s)

> User name of the submitter

exitStatus (%d)

> Task exit status

dispTime (%ld)

> Dispatch time – time at which the task was dispatched for execution

termTime (%ld)

> Completion time – time when task is completed/failed

fromHost (%s)

> Submission host name

execHost (%s)

> Execution host name

cwd (%s)

> Current working directory

cmdln (%s)

> Command line of the task

lsfRusage

> The following fields contain resource usage information for the job (see `getrusage` (2)). If the value of some field is unavailable (due to job exit or the difference among the operating systems), -1 will be logged. Times are measured in seconds, and sizes are measured in KB.

ru_utime (%f)

>   User time used

ru_stime (%f)

>   System time used

ru_maxrss (%f)

>   Maximum shared text size

ru_ixrss (%f)

>   Integral of the shared text size over time (in KB seconds)

ru_ismrss (%f)

>   Integral of the shared memory size over time (valid only on Ultrix)

ru_idrss (%f)

>   Integral of the unshared data size over time

ru_isrss (%f)

>   Integral of the unshared stack size over time

ru_minflt (%f)

>   Number of page reclaims

ru_majflt (%f)

>   Number of page faults

ru_nswap (%f)

>   Number of times the process was swapped out

ru_inblock (%f)

>   Number of block input operations

ru_oublock (%f)

>   Number of block output operations

ru_ioch (%f)

>   Number of characters read and written (valid only on HP-UX)

ru_msgsnd (%f)

>   Number of System V IPC messages sent

ru_msgrcv (%f)

>   Number of messages received

ru_nsignals (%f)

>   Number of signals received

ru_nvcsw (%f)

Number of voluntary context switches

ru_nivcsw (%f)

Number of involuntary context switches

ru_exutime (%f)

Exact user time used (valid only on ConvexOS)

lsf.cluster

Contents

About lsf.cluster

This is the cluster configuration file. There is one for each cluster, called `lsf.cluster.`*cluster_name*. The *cluster_name* suffix is the name of the cluster defined in the Cluster section of `lsf.shared`. All LSF hosts are listed in this file, along with the list of LSF administrators and the installed LSF features.

The `lsf.cluster.` *cluster_name* file contains two types of configuration information:

- Cluster definition information — affects all LSF applications. Defines cluster administrators, hosts that make up the cluster, attributes of each individual host such as host type or host model, and resources using the names defined in `lsf.shared`.
- LIM policy information — affects applications that rely on LIM job placement policy. Defines load sharing and job placement policies provided by LIM.

Changing lsf.cluster configuration

After making any changes to `lsf.cluster.` *cluster_name*, run the following commands:

- `lsadmin reconfig` to reconfigure LIM
- `badmin mbdrestart` to restart `mbatchd`

Location

This file is typically installed in the directory defined by LSF_ENVDIR.

Structure

The `lsf.cluster.` *cluster_name* file contains the following configuration sections:

- Parameters section
- ClusterAdmins section
- Host section
- ResourceMap section
- RemoteClusters section

Parameters

- ADJUST_DURATION
- ELIM_POLL_INTERVAL
- ELIMARGS

- EXINTERVAL
- FLOAT_CLIENTS
- FLOAT_CLIENTS_ADDR_RANGE
- HOST_INACTIVITY_LIMIT
- LSF_ELIM_BLOCKTIME
- LSF_ELIM_DEBUG
- LSF_ELIM_RESTARTS
- LSF_HOST_ADDR_RANGE
- MASTER_INACTIVITY_LIMIT
- PROBE_TIMEOUT
- PRODUCTS
- RETRY_LIMIT

ADJUST_DURATION

Syntax

ADJUST_DURATION=*integer*

Description

Integer reflecting a multiple of EXINTERVAL that controls the time period during which load adjustment is in effect

The lsplace(1) and lsloadadj (1) commands artificially raise the load on a selected host. This increase in load decays linearly to 0 over time.

Default

3

ELIM_POLL_INTERVAL

Syntax

ELIM_POLL_INTERVAL=*seconds*

Description

Time interval, in seconds, that the LIM samples external load index information. If your elim executable is programmed to report values more frequently than every 5 seconds, set the ELIM_POLL_INTERVAL so that it samples information at a corresponding rate.

Valid values

0.001 to 5

Default

5 seconds

ELIMARGS

Syntax

ELIMARGS=*cmd_line_args*

Description

Specifies command-line arguments required by an el i m executable on startup. Used only when the external load indices feature is enabled.

Default

Undefined

EXINTERVAL

Syntax

EXINTERVAL=*time_in_seconds*

Description

Time interval, in seconds, at which the LIM daemons exchange load information

On extremely busy hosts or networks, or in clusters with a large number of hosts, load may interfere with the periodic communication between LIM daemons. Setting EXINTERVAL to a longer interval can reduce network load and slightly improve reliability, at the cost of slower reaction to dynamic load changes.

Note that if you define the time interval as less than 5 seconds, LSF automatically resets it to 5 seconds.

Default

15 seconds

FLOAT_CLIENTS

Syntax

FLOAT_CLIENTS=*number_of_floating_client_licenses*

Description

Sets the size of your license pool in the cluster

When the master LIM starts, up to *number_of_floating_client_licenses* will be checked out for use as floating client licenses. If fewer licenses are available than specified by *number_of_floating_client_licenses*, only the available licenses will be checked out and used.

If FLOAT_CLIENTS is not specified in l sf. cl ust er. *cluster_name* or there is an error in either l i cense. dat or in l sf. cl ust er. *cluster_name*, the floating LSF client license feature is disabled.

Caution:

When the LSF floating client feature is enabled, any host can submit jobs to the cluster. You can limit which hosts can be LSF floating clients with the parameter FLOAT_CLIENTS_ADDR_RANGE in lsf.cluster.cluster_name.

LSF Floating Client

Although an LSF Floating Client requires a license, LSF_Float_Client does not need to be added to the PRODUCTS line. LSF_Float_Client also cannot be added as a resource for specific hosts already defined in lsf.cluster.cluster_name. Should these lines be present, they are ignored by LSF.

Default

Undefined

FLOAT_CLIENTS_ADDR_RANGE

Syntax

FLOAT_CLIENTS_ADDR_RANGE=*IP_address* ...

Description

Optional. IP address or range of addresses of domains from which floating client hosts can submit requests. Multiple ranges can be defined, separated by spaces. The IP address can have either a dotted quad notation (IPv4) or IP Next Generation (IPv6) format. LSF supports both formats; you do not have to map IPv4 addresses to an IPv6 format.

> **Note:**
>
> To use IPv6 addresses, you must define the parameter LSF_ENABLE_SUPPORT_IPV6 in `lsf.conf`.

If the value of FLOAT_CLIENT_ADDR_RANGE is undefined, there is no security and any hosts can be LSF floating clients.

If a value is defined, security is enabled. If there is an error in the configuration of this variable, by default, no hosts will be allowed to be LSF floating clients.

When this parameter is defined, client hosts that do not belong to the domain will be denied access.

If a requesting host belongs to an IP address that falls in the specified range, the host will be accepted to become a floating client.

IP addresses are separated by spaces, and considered "OR" alternatives.

If you define FLOAT_CLIENT_ADDR_RANGE with:
- No range specified, all IPv4 and IPv6 clients can submit requests.
- Only an IPv4 range specified, only IPv4 clients within the range can submit requests.
- Only an IPv6 range specified, only IPv6 clients within the range can submit requests.
- Both an IPv6 and IPv4 range specified, IPv6 and IPv4 clients within the ranges can submit requests.

The asterisk (*) character indicates any value is allowed.

The dash (-) character indicates an explicit range of values. For example 1-4 indicates 1,2,3,4 are allowed.

Open ranges such as *-30, or 10-*, are allowed.

If a range is specified with fewer fields than an IP address such as 10.161, it is considered as 10.161.*.*.

Address ranges are validated at configuration time so they must conform to the required format. If any address range is not in the correct format, no hosts will be accepted as LSF floating clients, and an error message will be logged in the LIM log.

This parameter is limited to 2048 characters.

For IPv6 addresses, the double colon symbol (::) indicates multiple groups of 16-bits of zeros. You can also use (::) to compress leading and trailing zeros in an address filter, as shown in the following example:

FLOAT_CLIENTS_ADDR_RANGE=1080::8:800:20fc:*

This definition allows hosts with addresses 1080:0:0:0:8:800:20fc:* (three leading zeros).

You cannot use the double colon (::) more than once within an IP address. You cannot use a zero before or after (::). For example, 1080:0::8:800:20fc:* is not a valid address.

Notes

After you configure FLOAT_CLIENTS_ADDR_RANGE, check the lim.log. *host_name* file to make sure this parameter is correctly set. If this parameter is not set or is wrong, this will be indicated in the log file.

Examples

FLOAT_CLIENTS_ADDR_RANGE=100

All IPv4 and IPv6 hosts with a domain address starting with 100 will be allowed access.

- To specify only IPv4 hosts, set the value to **100.***
- To specify only IPv6 hosts, set the value to **100:***

FLOAT_CLIENTS_ADDR_RANGE=100-110.34.1-10.4-56

All client hosts belonging to a domain with an address having the first number between 100 and 110, then 34, then a number between 1 and 10, then, a number between 4 and 56 will be allowed access. Example: 100.34.9.45, 100.34.1.4, 102.34.3.20, etc. No IPv6 hosts are allowed.

FLOAT_CLIENTS_ADDR_RANGE=100.172.1.13 100.*.30-54 124.24-*.1.*-34

All client hosts belonging to a domain with the address 100.172.1.13 will be allowed access. All client hosts belonging to domains starting with 100, then any number, then a range of 30 to 54 will be allowed access. All client hosts belonging to domains starting with 124, then from 24 onward, then 1, then from 0 to 34 will be allowed access. No IPv6 hosts are allowed.

FLOAT_CLIENTS_ADDR_RANGE=12.23.45.*

All client hosts belonging to domains starting with 12.23.45 are allowed. No IPv6 hosts are allowed.

FLOAT_CLIENTS_ADDR_RANGE=100.*43

The * character can only be used to indicate any value. In this example, an error will be inserted in the LIM log and no hosts will be accepted to become LSF floating clients. No IPv6 hosts are allowed.

FLOAT_CLIENTS_ADDR_RANGE=100.*43 100.172.1.13

Although one correct address range is specified, because *43 is not correct format, the entire line is considered not valid. An error will be inserted in the LIM log and no hosts will be accepted to become LSF floating clients. No IPv6 hosts are allowed.

FLOAT_CLIENTS_ADDR_RANGE = 3ffe

All client IPv6 hosts with a domain address starting with 3ffe will be allowed access. No IPv4 hosts are allowed.

FLOAT_CLIENTS_ADDR_RANGE = 3ffe:fffe::88bb:*

Expands to 3ffe:fffe:0:0:0:0:88bb:*. All IPv6 client hosts belonging to domains starting with 3ffe:fffe::88bb:* are allowed. No IPv4 hosts are allowed.

FLOAT_CLIENTS_ADDR_RANGE = 3ffe-4fff:fffe::88bb:aa-ff 12.23.45.*

All IPv6 client hosts belonging to domains starting with 3ffe up to 4fff, then fffe::88bb, and ending with aa up to ff are allowed. All IPv4 client hosts belonging to domains starting with 12.23.45 are allowed.

FLOAT_CLIENTS_ADDR_RANGE = 3ffe-*:fffe::88bb:*-ff

All IPv6 client hosts belonging to domains starting with 3ffe up to ffff and ending with 0 up to ff are allowed. No IPv4 hosts are allowed.

Default

Undefined. No security is enabled. Any host in any domain is allowed access to LSF floating client licenses.

See also

LSF_ENABLE_SUPPORT_IPV6

HOST_INACTIVITY_LIMIT

Syntax

HOST_INACTIVITY_LIMIT=*integer*

Description

Integer that is multiplied by EXINTERVAL, the time period you set for the communication between the master and slave LIMs to ensure all parties are functioning.

A slave LIM can send its load information any time from EXINTERVAL to (HOST_INACTIVITY_LIMIT-1)*EXINTERVAL seconds. A master LIM sends a master announce to each host at least every EXINTERVAL*HOST_INACTIVITY_LIMIT seconds.

The HOST_INACTIVITY_LIMIT must be greater than or equal to 2.

Increase or decrease the host inactivity limit to adjust for your tolerance for communication between master and slaves. For example, if you have hosts that frequently become inactive, decrease the host inactivity limit. Note that to get the right interval, you may also have to adjust your EXINTERVAL.

Default

5

LSF_ELIM_BLOCKTIME

Syntax

LSF_ELIM_BLOCKTIME=*seconds*

Description

UNIX only; used when the external load indices feature is enabled.

Maximum amount of time the master external load information manager (MELIM) waits for a complete load update string from an el im executable. After the time period specified by LSF_ELIM_BLOCKTIME, the MELIM writes the last string sent by an el im in the LIM log file (lim.log.*host_name*) and restarts the el im.

Defining LSF_ELIM_BLOCKTIME also triggers the MELIM to restart el im executables if the el im does not write a complete load update string within the time specified for LSF_ELIM_BLOCKTIME.

Valid Values

Non-negative integers. For example, if your el im writes name-value pairs with 1 second intervals between them, and your el im reports 12 load indices, allow at least 12 seconds for the el im to finish writing the entire load update string. In this case, define LSF_ELIM_BLOCKTIME as 15 seconds or more.

A value of 0 indicates that the MELIM expects to receive the entire load string all at once.

If you comment out or delete LSF_ELIM_BLOCKTIME, the MELIM waits 2 seconds for a complete load update string.

Default

4 seconds

See also

LSF_ELIM_RESTARTS to limit how many times the ELIM can be restarted.

LSF_ELIM_DEBUG

Syntax

LSF_ELIM_DEBUG=y

Description

UNIX only; used when the external load indices feature is enabled.

When this parameter is set to y, all external load information received by the load information manager (LIM) from the master external load information manager (MELIM) is logged in the LIM log file (lim.log.*host_name*).

Defining LSF_ELIM_DEBUG also triggers the MELIM to restart el im executables if the el im does not write a complete load update string within the time specified for LSF_ELIM_BLOCKTIME.

Default

Undefined; external load information sent by an to the MELIM is not logged.

See also

LSF_ELIM_BLOCKTIME to configure how long LIM waits before restarting the ELIM.

LSF_ELIM_RESTARTS to limit how many times the ELIM can be restarted.

LSF_ELIM_RESTARTS

Syntax

LSF_ELIM_RESTARTS=*integer*

Description

UNIX only; used when the external load indices feature is enabled.

Maximum number of times the master external load information manager (MELIM) can restart elim executables on a host. Defining this parameter prevents an ongoing restart loop in the case of a faulty elim. The MELIM waits the LSF_ELIM_BLOCKTIME to receive a complete load update string before restarting the elim. The MELIM does not restart any elim executables that exit with ELIM_ABORT_VALUE.

Important:

Either LSF_ELIM_BLOCKTIME or LSF_ELIM_DEBUG must also be defined; defining these parameters triggers the MELIM to restart elim executables.

Valid Values

Non-negative integers.

Default

Undefined; the number of elim restarts is unlimited.

See also

LSF_ELIM_BLOCKTIME, LSF_ELIM_DEBUG

LSF_HOST_ADDR_RANGE

Syntax

LSF_HOST_ADDR_RANGE=*IP_address*...

Description

Identifies the range of IP addresses that are allowed to be LSF hosts that can be dynamically added to or removed from the cluster.

Caution:

To enable dynamically added hosts after installation, you must
define LSF_HOST_ADDR_RANGE in lsf.cluster.cluster_name,
and LSF_DYNAMIC_HOST_WAIT_TIME in lsf.conf. If you
enable dynamic hosts during installation, you must define an IP
address range after installation to enable security.

If a value is defined, security for dynamically adding and removing hosts is enabled, and only hosts with IP addresses within the specified range can be added to or removed from a cluster dynamically.

Specify an IP address or range of addresses, using either a dotted quad notation (IPv4) or IP Next Generation (IPv6) format. LSF supports both formats; you do not have to map IPv4 addresses to an IPv6 format. Multiple ranges can be defined, separated by spaces.

Note:

To use IPv6 addresses, you must define the parameter
LSF_ENABLE_SUPPORT_IPV6 in lsf.conf.

If there is an error in the configuration of LSF_HOST_ADDR_RANGE (for example, an address range is not in the correct format), no host will be allowed to join the cluster dynamically and an error message will be logged in the LIM log. Address ranges are validated at startup, reconfiguration, or restart, so they must conform to the required format.

If a requesting host belongs to an IP address that falls in the specified range, the host will be accepted to become a dynamic LSF host.

IP addresses are separated by spaces, and considered "OR" alternatives.

If you define the parameter LSF_HOST_ADDR_RANGE with:

- No range specified, all IPv4 and IPv6 clients are allowed.
- Only an IPv4 range specified, only IPv4 clients within the range are allowed.
- Only an IPv6 range specified, only IPv6 clients within the range are allowed.
- Both an IPv6 and IPv4 range specified, IPv6 and IPv4 clients within the ranges are allowed.

The asterisk (*) character indicates any value is allowed.

The dash (-) character indicates an explicit range of values. For example 1-4 indicates 1,2,3,4 are allowed.

Open ranges such as *-30, or 10-*, are allowed.

For IPv6 addresses, the double colon symbol (::) indicates multiple groups of 16-bits of zeros. You can also use (::) to compress leading and trailing zeros in an address filter, as shown in the following example:

LSF_HOST_ADDR_RANGE=1080::8:800:20fc:*

This definition allows hosts with addresses 1080:0:0:0:8:800:20fc:* (three leading zeros).

You cannot use the double colon (::) more than once within an IP address. You cannot use a zero before or after (::). For example, 1080:0::8:800:20fc:* is not a valid address.

If a range is specified with fewer fields than an IP address such as 10.161, it is considered as 10.161.*.*.

This parameter is limited to 2048 characters.

Notes

After you configure LSF_HOST_ADDR_RANGE, check the lim.log.*host_name* file to make sure this parameter is correctly set. If this parameter is not set or is wrong, this will be indicated in the log file.

Examples

LSF_HOST_ADDR_RANGE=100

All IPv4 and IPv6 hosts with a domain address starting with 100 will be allowed access.

- To specify only IPv4 hosts, set the value to **100.***
- To specify only IPv6 hosts, set the value to **100:***

LSF_HOST_ADDR_RANGE=100-110.34.1-10.4-56

All hosts belonging to a domain with an address having the first number between 100 and 110, then 34, then a number between 1 and 10, then, a number between 4 and 56 will be allowed access. No IPv6 hosts are allowed. Example: 100.34.9.45, 100.34.1.4, 102.34.3.20, etc.

LSF_HOST_ADDR_RANGE=100.172.1.13 100.*.30-54 124.24-*.1.*-34

The host with the address 100.172.1.13 will be allowed access. All hosts belonging to domains starting with 100, then any number, then a range of 30 to 54 will be allowed access. All hosts belonging to domains starting with 124, then from 24 onward, then 1, then from 0 to 34 will be allowed access. No IPv6 hosts are allowed.

LSF_HOST_ADDR_RANGE=12.23.45.*

All hosts belonging to domains starting with 12.23.45 are allowed. No IPv6 hosts are allowed.

LSF_HOST_ADDR_RANGE=100.*43

The * character can only be used to indicate any value. The format of this example is not correct, and an error will be inserted in the LIM log and no hosts will be able to join the cluster dynamically. No IPv6 hosts are allowed.

LSF_HOST_ADDR_RANGE=100.*43 100.172.1.13

Although one correct address range is specified, because *43 is not correct format, the entire line is considered not valid. An error will be inserted in the LIM log and no hosts will be able to join the cluster dynamically. No IPv6 hosts are allowed.

LSF_HOST_ADDR_RANGE = 3ffe

All client IPv6 hosts with a domain address starting with 3ffe will be allowed access. No IPv4 hosts are allowed.

LSF_HOST_ADDR_RANGE = 3ffe:fffe::88bb:*

Expands to 3ffe:fffe:0:0:0:0:88bb:*.All IPv6 client hosts belonging to domains starting with 3ffe:fffe::88bb:* are allowed. No IPv4 hosts are allowed.

LSF_HOST_ADDR_RANGE = 3ffe-4fff:fffe::88bb:aa-ff 12.23.45.*

All IPv6 client hosts belonging to domains starting with 3ffe up to 4fff, then fffe::88bb, and ending with aa up to ff are allowed. IPv4 client hosts belonging to domains starting with 12.23.45 are allowed.

LSF_HOST_ADDR_RANGE = 3ffe-*:fffe::88bb:*-ff

All IPv6 client hosts belonging to domains starting with 3ffe up to ffff and ending with 0 up to ff are allowed. No IPv4 hosts are allowed.

Default

Undefined (dynamic host feature disabled). If you enable dynamic hosts during installation, no security is enabled and all hosts can join the cluster.

See also

LSF_ENABLE_SUPPORT_IPV6

MASTER_INACTIVITY_LIMIT

Syntax

MASTER_INACTIVITY_LIMIT=*integer*

Description

An integer reflecting a multiple of EXINTERVAL. A slave will attempt to become master if it does not hear from the previous master after (HOST_INACTIVITY_LIMIT +*host_number**MASTER_INACTIVITY_LIMIT)*EXINTERVAL seconds, where *host_number* is the position of the host in lsf.cluster.*cluster_name*.

The master host is *host_number* 0.

Default

2

PROBE_TIMEOUT

Syntax

PROBE_TIMEOUT=*time_in_seconds*

Description

Specifies the timeout in seconds to be used for the connect(2) system call

Before taking over as the master, a slave LIM will try to connect to the last known master via TCP.

Default

2 seconds

PRODUCTS

Syntax

PRODUCTS=*product_name* ...

Description

Specifies the LSF products and features that the cluster will run (you must also have a license for every product you want to run). The list of items is separated by space.

The PRODUCTS parameter is set automatically during installation to include core features. Here are some of the optional products and features that can be specified:

- LSF_Make
- LSF_MultiCluster

Default

LSF_Base LSF_Manager LSF_Make

RETRY_LIMIT

Syntax

RETRY_LIMIT=*integer*

Description

Integer reflecting a multiple of EXINTERVAL that controls the number of retries a master or slave LIM makes before assuming that the slave or master is unavailable.

If the master does not hear from a slave for HOST_INACTIVITY_LIMIT exchange intervals, it will actively poll the slave for RETRY_LIMIT exchange intervals before it will declare the slave as unavailable. If a slave does not hear from the master for HOST_INACTIVITY_LIMIT exchange intervals, it will actively poll the master for RETRY_LIMIT intervals before assuming that the master is down.

Default

2

ClusterAdmins section

(Optional) The ClusterAdmins section defines the LSF administrators for the cluster. The only keyword is ADMINISTRATORS.

If the ClusterAdmins section is not present, the default LSF administrator is root. Using root as the primary LSF administrator is not recommended.

ADMINISTRATORS

Syntax

ADMINISTRATORS=*administrator_name ...*

Description

Specify UNIX user names.

You can also specify UNIX user group names, Windows user names, and Windows user group names.To specify a Windows user account or user group, include the domain name in uppercase letters (*DOMAIN_NAME\user_name* or *DOMAIN_NAME\user_group*).

The first administrator of the expanded list is considered the primary LSF administrator. The primary administrator is the owner of the LSF configuration files, as well as the working files under LSB_SHAREDIR/*cluster_name*. If the primary administrator is changed, make sure the owner of the configuration files and the files under LSB_SHAREDIR/*cluster_name* are changed as well.

Administrators other than the primary LSF administrator have the same privileges as the primary LSF administrator except that they do not have permission to change LSF configuration files. They can perform clusterwide operations on jobs, queues, or hosts in the system.

For flexibility, each cluster may have its own LSF administrators, identified by a user name, although the same administrators can be responsible for several clusters.

Use the -l option of the lsclusters command to display all of the administrators within a cluster.

Windows domain:

- If the specified user or user group is a domain administrator, member of the Power Users group or a group with domain administrative privileges, the specified user or user group must belong to the LSF user domain.
- If the specified user or user group is a user or user group with a lower degree of privileges than outlined in the previous point, the user or user group must belong to the LSF user domain and be part of the Global Admins group.

    Windows workgroup
- If the specified user or user group is not a workgroup administrator, member of the Power Users group, or a group with administrative privileges on each host, the specified user or user group must belong to the Local Admins group on each host.

Compatibility

For backwards compatibility, ClusterManager and Manager are synonyms for ClusterAdmins and ADMINISTRATORS respectively. It is possible to have both sections present in the same lsf.cluster.*cluster_name* file to allow daemons from different LSF versions to share the same file.

Example

The following gives an example of a cluster with two LSF administrators. The user listed first, user2, is the primary administrator.

```
Begin ClusterAdmins
```
```
ADMINISTRATORS = user2 user7
```
```
End ClusterAdmins
```

Default

lsfadmin

Host section

The Host section is the last section in lsf.cluster.*cluster_name* and is the only required section. It lists all the hosts in the cluster and gives configuration information for each host.

The order in which the hosts are listed in this section is important, because the first host listed becomes the LSF master host. Since the master LIM makes all placement decisions for the cluster, it should be on a fast machine.

The LIM on the first host listed becomes the master LIM if this host is up; otherwise, that on the second becomes the master if its host is up, and so on. Also, to avoid the delays involved in switching masters if the first machine goes down, the master should be on a reliable machine. It is desirable to arrange the list such that the first few hosts in the list are always in the same

subnet. This avoids a situation where the second host takes over as master when there are communication problems between subnets.

Configuration information is of two types:

- Some fields in a host entry simply describe the machine and its configuration.
- Other fields set thresholds for various resources.

Example Host section

This example Host section contains descriptive and threshold information for three hosts:

```
Begin Host
HOSTNAME    model     type    server r1m pg tmp RESOURCES        RUNWINDOW
hostA       SparcIPC  Sparc   1      3.5 15   0 (sunos frame)    ()
hostD       Sparc10   Sparc   1      3.5 15   0 (sunos)          (5:18:30-1:8:30)
hostD       !         !       1      2.0 10   0 ()               ()
hostE       !         !       1      2.0 10   0 (linux !bigmem)  ()
End Host
```

Descriptive fields

The following fields are required in the Host section:

- HOSTNAME
- RESOURCES
- type
- model

The following fields are optional:

- server
- nd
- RUNWINDOW
- REXPRI

HOSTNAME

Description

Official name of the host as returned by hostname(1)

The name must be listed in lsf.shared as belonging to this cluster.

model

Description

Host model

The name must be defined in the HostModel section of lsf.shared. This determines the CPU speed scaling factor applied in load and placement calculations.

Optionally, the ! keyword for the model or type column, indicates that the host model or type is to be automatically detected by the LIM running on the host.

nd

Description

Number of local disks

This corresponds to the ndisks static resource. On most host types, LSF automatically determines the number of disks, and the nd parameter is ignored.

nd should only count local disks with file systems on them. Do not count either disks used only for swapping or disks mounted with NFS.

Default

The number of disks determined by the LIM, or 1 if the LIM cannot determine this

RESOURCES

Description

The static Boolean resources and static or dynamic numeric and string resources available on this host.

The resource names are strings defined in the Resource section of lsf.shared. You may list any number of resources, enclosed in parentheses and separated by blanks or tabs. For example:

```
(fs frame hpux)
```

Optionally, you can specify an exclusive resource by prefixing the resource with an exclamation mark (!). For example, resource bigmem is defined in lsf.shared, and is defined as an exclusive resource for hostE:

```
Begin Host
HOSTNAME    model    type    server r1m pg tmp RESOURCES        RUNWINDOW
...
hostE       !        !       1      2.0 10   0 (linux !bigmem)  ()
...
End Host
```

You must explicitly specify the exclusive resources in the resource requirements for the job to select a host with an exclusive resource for a job. For example:

bsub -R "bigmem" myjob

or

bsub -R "defined(bigmem)" myjob

You can specify static and dynamic numeric and string resources in the resource column of
the Host clause. For example:

```
Begin    Host

HOSTNAME   model  type  server  r1m   mem   swp  RESOURCES    #Keywords

hostA       !      !      1      3.5   ()    ()   (mg elimres patchrev=3 owner=user1)

hostB       !      !      1      3.5   ()    ()   (specman=5 switch=1 owner=test)

hostC       !      !      1      3.5   ()    ()   (switch=2 rack=rack2_2_3 owner=test)

hostD       !      !      1      3.5   ()    ()   (switch=1 rack=rack2_2_3 owner=test)

End      Host
```

REXPRI

Description

UNIX only

Default execution priority for interactive remote jobs run under the RES

The range is from -20 to 20. REXPRI corresponds to the BSD-style nice value used for remote
jobs. For hosts with System V-style nice values with the range 0 - 39, a REXPRI of -20
corresponds to a nice value of 0, and +20 corresponds to 39. Higher values of REXPRI
correspond to lower execution priority; -20 gives the highest priority, 0 is the default priority
for login sessions, and +20 is the lowest priority.

Default

0

RUNWINDOW

Description

Dispatch window for interactive tasks.

When the host is not available for remote execution, the host status is lockW (locked by run
window). LIM does not schedule interactive tasks on hosts locked by dispatch windows. Run
windows only apply to interactive tasks placed by LIM. The LSF batch system uses its own
(optional) host dispatch windows to control batch job processing on batch server hosts.

Format

A dispatch window consists of one or more time windows in the format *begin_time-
end_time*. No blanks can separate *begin_time* and *end_time*. Time is specified in the form
[*day:*]*hour*[:*minute*]. If only one field is specified, LSF assumes it is an *hour*. Two fields are
assumed to be *hour:minute*. Use blanks to separate time windows.

Default

Always accept remote jobs

server

Description

Indicates whether the host can receive jobs from other hosts

Specify 1 if the host can receive jobs from other hosts; specify 0 otherwise. Servers that are set to 0 are LSF clients. Client hosts do not run the LSF daemons. Client hosts can submit interactive and batch jobs to the cluster, but they cannot execute jobs sent from other hosts.

Default

1

type

Description

Host type as defined in the HostType section of lsf.shared

The strings used for host types are determined by the system administrator: for example, SUNSOL, DEC, or HPPA. The host type is used to identify binary-compatible hosts.

The host type is used as the default resource requirement. That is, if no resource requirement is specified in a placement request, the task is run on a host of the same type as the sending host.

Often one host type can be used for many machine models. For example, the host type name SUNSOL6 might be used for any computer with a SPARC processor running SunOS 6. This would include many Sun models and quite a few from other vendors as well.

Optionally, the ! keyword for the model or type column, indicates that the host model or type is to be automatically detected by the LIM running on the host.

Threshold fields

The LIM uses these thresholds in determining whether to place remote jobs on a host. If one or more LSF load indices exceeds the corresponding threshold (too many users, not enough swap space, etc.), then the host is regarded as busy, and LIM will not recommend jobs to that host.

The CPU run queue length threshold values (r15s, r1m, and r15m) are taken as effective queue lengths as reported by lsload -E.

All of these fields are optional; you only need to configure thresholds for load indices that you wish to use for determining whether hosts are busy. Fields that are not configured are not considered when determining host status. The keywords for the threshold fields are not case sensitive.

Thresholds can be set for any of the following:
- The built-in LSF load indexes (r15s, r1m, r15m, ut, pg, it, io, ls, swp, mem, tmp)
- External load indexes defined in the Resource section of lsf.shared

ResourceMap section

The ResourceMap section defines shared resources in your cluster. This section specifies the mapping between shared resources and their sharing hosts. When you define resources in the Resources section of lsf.shared, there is no distinction between a shared and non-shared

resource. By default, all resources are not shared and are local to each host. By defining the ResourceMap section, you can define resources that are shared by all hosts in the cluster or define resources that are shared by only some of the hosts in the cluster.

This section must appear after the Host section of `lsf.cluster.` *cluster_name*, because it has a dependency on host names defined in the Host section.

ResourceMap section structure

The first line consists of the keywords RESOURCENAME and LOCATION. Subsequent lines describe the hosts that are associated with each configured resource.

Example ResourceMap section

```
Begin ResourceMap

RESOURCENAME    LOCATION

verilog         (5@[all])

local           ([host1 host2] [others])

End ResourceMap
```

The resource `verilog` must already be defined in the RESOURCE section of the `lsf.shared` file. It is a static numeric resource shared by all hosts. The value for `verilog` is 5. The resource `local` is a numeric shared resource that contains two instances in the cluster. The first instance is shared by two machines, `host1` and `host2`. The second instance is shared by all other hosts.

Resources defined in the ResourceMap section can be viewed by using the -s option of the `lshosts` (for static resource) and `lsload` (for dynamic resource) commands.

LOCATION

Description

Defines the hosts that share the resource

For a static resource, you must define an initial value here as well. Do not define a value for a dynamic resource.

instance is a list of host names that share an instance of the resource. The reserved words all, others, and default can be specified for the instance:

all — Indicates that there is only one instance of the resource in the whole cluster and that this resource is shared by all of the hosts

Use the not operator (~) to exclude hosts from the all specification. For example:
```
(2@[all ~host3 ~host4])
```

means that 2 units of the resource are shared by all server hosts in the cluster made up of `host1 host2 ... host`*n*, except for `host3` and `host4`. This is useful if you have a large cluster but only want to exclude a few hosts.

The parentheses are required in the specification. The not operator can only be used with the all keyword. It is not valid with the keywords others and default.

others — Indicates that the rest of the server hosts not explicitly listed in the LOCATION field comprise one instance of the resource

For example:

```
2@[host1] 4@[others]
```

indicates that there are 2 units of the resource on host1 and 4 units of the resource shared by all other hosts.

default — Indicates an instance of a resource on each host in the cluster

This specifies a special case where the resource is in effect not shared and is local to every host. default means at each host. Normally, you should not need to use default, because by default all resources are local to each host. You might want to use ResourceMap for a non-shared static resource if you need to specify different values for the resource on different hosts.

RESOURCENAME

Description

Name of the resource

This resource name must be defined in the Resource section of lsf.shared. You must specify at least a name and description for the resource, using the keywords RESOURCENAME and DESCRIPTION.

- A resource name cannot begin with a number.
- A resource name cannot contain any of the following characters:

```
: . ( ) [ + - * / ! & | < > @ =
```

- A resource name cannot be any of the following reserved names:

```
cpu cpuf io logins ls idle maxmem maxswp maxtmp type model status it
```

```
mem ncpus define_ncpus_cores define_ncpus_procs
```

```
define_ncpus_threads ndisks pg r15m r15s r1m swap swp tmp ut
```

- To avoid conflict with inf and nan keywords in 3rd-party libraries, resource names should not begin with inf or nan (upper case or lower case). Resource requirment strings, such as -R "infra" or -R "nano" will cause an error. Use -R "defined(infxx)" or -R "defined(nanxx)", to specify these resource names.
- Resource names are case sensitive
- Resource names can be up to 39 characters in length

RemoteClusters section

Optional. This section is used only in a MultiCluster environment. By default, the local cluster can obtain information about all other clusters specified in lsf.shared. The RemoteClusters section limits the clusters that the local cluster can obtain information about.

The RemoteClusters section is required if you want to configure cluster equivalency, cache interval, daemon authentication across clusters, or if you want to run parallel jobs across clusters. To maintain compatibility in this case, make sure the list includes all clusters specified in lsf.shared, even if you only configure the default behavior for some of the clusters.

The first line consists of keywords. CLUSTERNAME is mandatory and the other parameters are optional.

Subsequent lines configure the remote cluster.

Example RemoteClusters section

```
Begin RemoteClusters
CLUSTERNAME   EQUIV    CACHE_INTERVAL   RECV_FROM   AUTH
cluster1       Y           60              Y        KRB
cluster2       N           60              Y        -
cluster4       N           60              N        PKI
End RemoteClusters
```

CLUSTERNAME

Description

Remote cluster name

Defines the Remote Cluster list. Specify the clusters you want the local cluster will recognize. Recognized clusters must also be defined in lsf.shared. Additional clusters listed in lsf.shared but not listed here will be ignored by this cluster.

EQUIV

Description

Specify 'Y' to make the remote cluster equivalent to the local cluster. Otherwise, specify 'N'. The master LIM considers all equivalent clusters when servicing requests from clients for load, host, or placement information.

EQUIV changes the default behavior of LSF commands and utilities and causes them to automatically return load (lsload(1)), host (lshosts(1)), or placement (lsplace(1)) information about the remote cluster as well as the local cluster, even when you don't specify a cluster name.

CACHE_INTERVAL

Description

Specify the load information cache threshold, in seconds. The host information threshold is twice the value of the load information threshold.

To reduce overhead and avoid updating information from remote clusters unnecessarily, LSF displays information in the cache, unless the information in the cache is older than the threshold value.

Default

60 (seconds)

RECV_FROM

Description

Specifies whether the local cluster accepts parallel jobs that originate in a remote cluster

RECV_FROM does not affect regular or interactive batch jobs.

Specify 'Y' if you want to run parallel jobs across clusters. Otherwise, specify 'N'.

Default

Y

AUTH

Description

Defines the preferred authentication method for LSF daemons communicating across clusters. Specify the same method name that is used to identify the corresponding `eauth` program (eauth.*method_name*). If the remote cluster does not prefer the same method, LSF uses default security between the two clusters.

Default

- (only privileged port (setuid) authentication is used between clusters)

lsf.*cluster_name*.license.acct

This is the license accounting file. There is one for each cluster, called `lsf.`*cluster_name*`.license.acct`. The *cluster_name* variable is the name of the cluster defined in the Cluster section of `lsf.shared`.

The `lsf.`*cluster_name*`.license.acct` file contains three types of configuration information:

- LSF license information
- MultiCluster license information

lsf.*cluster_name*.license.acct structure

The license audit log file is an ASCII file with one record per line. The fields of a record are separated by blanks.

File properties

Location

> The default location of this file is defined by LSF_LOGDIR in `lsf.conf`, but you can override this by defining LSF_LICENSE_ACCT_PATH in `lsf.conf`.

Owner

> The primary LSF admin is the owner of this file.

Permissions

```
-rw-r--r--
```

Records and fields

The fields of a record are separated by blanks. The fields in order of occurrence are as follows:

timestamp (%d)

> Time stamp of the logged event (in seconds since the epoch).

type (%s)

> The LSF product type. The valid values are as follows:
>
> - LSF_MANAGER
> - LSF_MULTI CLUSTER

version (%s)

> The version of the LSF product.

value (%s)

> The actual tracked value. The format of this field depends on the product type as specified by the `type` field:

LSF_MANAGER

> E *e_peak e_max_avail* S *s_peak s_max_avail* B *b_peak b_max_avail*
>
> Where

e_peak, *s_peak*, and *b_peak* are the peak usage values (in number of CPUs) of the E, S, and B class licenses, respectively.

e_max_avail, *s_max_avail*, and *b_max_avail* are the maximum availability and usage values (in number of CPUs) of the E, S, and B class licenses, respectively. This is determined by the license that you purchased.

LSF_MULTICLUSTER

mc_peak mc_max_avail

Where

mc_peak is the peak usage value (in number of CPUs) of the LSF MultiCluster license

mc_max_avail is the maximum availability and usage (in number of CPUs) of the LSF MultiCluster license. This is determined by the license that you purchased.

status (%s)

The results of the license usage check. The valid values are as follows:

OK

Peak usage is less than the maximum license availability

OVERUSE

Peak usage is more than the maximum license availability

hash (%s)

Line encryption used to authenticate the record.

Example record Format

```
1128372131 LSF_MANAGER 7.0 E hostA OVERUSE 7c7998a6861ea119cd48414a820be18cd641
1128372131 LSF_MULTICLUSTER 7.0 8 10 OK 281288c606a50065ea0e2f3e7161972c56491dc
1128372185 LSF_MANAGER 7.0 E 8 0 S 0 2 B 0 10 OVERUSE fb439ee293821761af9ed0785 1128372185
LSF_MANAGER 7.0 E hostA OVERUSE 2d22a06d6c5cfd5aba40875c2cb8544444a5
```

lsf.conf

The lsf.conf file controls the operation of LSF.

About lsf.conf

lsf.conf is created during installation and records all the settings chosen when LSF was installed. The lsf.conf file dictates the location of the specific configuration files and operation of individual servers and applications.

The lsf.conf file is used by LSF and applications built on top of it. For example, information in lsf.conf is used by LSF daemons and commands to locate other configuration files, executables, and network services. lsf.conf is updated, if necessary, when you upgrade to a new version.

This file can also be expanded to include application-specific parameters.

Corresponding parameters in ego.conf

When Platform EGO is enabled in LSF Version 7, you can configure some LSF parameters in lsf.conf that have corresponding Platform EGO parameter names in EGO_CONFDIR/ego.conf (LSF_CONFDIR/lsf.conf is a separate file from EGO_CONFDIR/ego.conf). If both the LSF and the EGO parameters are set in their respective files, the definition in ego.conf is used. You must continue to set LSF parameters only in lsf.conf.

When EGO is enabled in the LSF cluster (LSF_ENABLE_EGO=Y), you also can set the following EGO parameters related to LIM, PIM, and ELIM in either lsf.conf or ego.conf:

- EGO_DISABLE_UNRESOLVABLE_HOST (dynamically added hosts only)
- EGO_PREDEFINED_RESOURCES
- EGO_ENABLE_AUTO_DAEMON_SHUTDOWN
- EGO_DAEMONS_CPUS
- EGO_DEFINE_NCPUS
- EGO_SLAVE_CTRL_REMOTE_HOST
- EGO_WORKDIR
- EGO_PIM_SWAP_REPORT
- EGO_ESLIM_TIMEOUT

If EGO is not enabled, you can set these parameters only in lsf.conf.

See *Administering Platform LSF* for more information about configuring LSF for EGO. See the *Platform EGO Reference* for information about ego.conf parameters.

Changing lsf.conf configuration

After making any changes to lsf.conf, run the following commands:

- lsadmin reconfig to reconfigure LIM
- badmin mbdrestart to restart mbatchd

If you have installed LSF in a mixed cluster, you must make sure that lsf.conf parameters set on UNIX and Linux match any corresponding parameters in the local lsf.conf files on your Windows hosts.

Location

The default location of lsf.conf is in /conf. This default location can be overridden when necessary by either the environment variable LSF_ENVDIR or the command line option -d available to some of the applications.

Format

Each entry in lsf.conf has one of the following forms:

```
NAME=VALUE
```

```
NAME=
```

```
NAME="STRING1 STRING2 ..."
```

The equal sign = must follow each NAME even if no value follows and there should be no space beside the equal sign.

A value that contains multiple strings separated by spaces must be enclosed in quotation marks.

Lines starting with a pound sign (#) are comments and are ignored. Do not use #if as this is reserved syntax for time-based configuration.

DAEMON_SHUTDOWN_DELAY

Syntax

DAEMON_SHUTDOWN_DELAY=_time_in_seconds_

Description

Applies when EGO_ENABLE_AUTO_DAEMON_SHUTDOWN=Y. Controls amount of time the slave LIM waits to communicate with other (RES and SBD) local daemons before exiting. Used to shorten or lengthen the time interval between a host attempting to join the cluster and, if it was unsuccessful, all of the local daemons shutting down.

The value should not be less than the minimum interval of RES and SBD housekeeping. Most administrators should set this value to somewhere between 3 minutes and 60 minutes.

Default

1800 seconds (30 minutes)

EGO_DEFINE_NCPUS

Syntax

EGO_DEFINE_NCPUS=procs | **cores** | **threads**

Description

If defined, enables an administrator to define a value other than the number of processors available. Follow one of the three equations below for an accurate value.

- EGO_DEFINE_NCPUS=procs-ncpus=number of processors
- EGO_DEFINE_NCPUS=cores-ncpus=number of processors x number of cores
- EGO_DEFINE_NCPUS=threads-ncpus=number of processors x number of cores x number of threads.

Note:

When PARALLEL_SCHED_BY_SLOT=Y in lsb.params, the resource requirement string keyword ncpus refers to the number of slots instead of the number of processors, however lshosts output will continue to show ncpus as defined by EGO_DEFINE_NCPUS in lsf.conf.

Default

EGO_DEFINE_NCPUS=procs

EGO_ENABLE_AUTO_DAEMON_SHUTDOWN

Syntax

EGO_ENABLE_AUTO_DAEMON_SHUTDOWN="Y" | "N"

Description

For hosts that attempted to join the cluster but failed to communicate within the LSF_DYNAMIC_HOST_WAIT_TIME period, automatically shuts down any running daemons.

This parameter can be useful if an administrator remove machines from the cluster regularly (by editing lsf.cluster file) or when a host belonging to the cluster is imaged, but the new host should not be part of the cluster. An administrator no longer has to go to each host that is not a part of the cluster to shut down any running daemons.

Default

N (daemons continue to run on hosts that were not successfully added to the cluster)

EGO_PARAMETER

EGO_ENABLE_AUTO_DAEMON_SHUTDOWN

EGO_ESLIM_TIMEOUT

Syntax

EGO_ESLIM_TIMEOUT=*time_seconds*

Description

Controls how long the LIM waits for any external static LIM scripts to run. After the timeout period expires, the LIM stops the scripts.

Use the external static LIM to automatically detect the operating system type and version of hosts.

LSF automatically detects the operating systems types and versions and displays them when running lshosts -l or lshosts -s. You can then specify those types in any -R resource requriement string. For example, **bsub -R "select[ostype=RHEL4.6]"**.

Default

10 seconds

EGO_PARAMETER

EGO_ESLIM_TIMEOUT

LSB_API_CONNTIMEOUT

Syntax

LSB_API_CONNTIMEOUT=*time_seconds*

Description

The timeout in seconds when connecting to LSF.

Valid values

Any positive integer or zero

Default

10

See also

LSB_API_RECVTIMEOUT

LSB_API_RECVTIMEOUT

Syntax

LSB_API_RECVTIMEOUT=*time_seconds*

Description

Timeout in seconds when waiting for a reply from LSF.

Valid values

Any positive integer or zero

Default

10

See also

LSB_API_CONNTIMEOUT

LSB_API_VERBOSE

Syntax

LSB_API_VERBOSE=Y | N

Description

When LSB_API_VERBOSE=Y, LSF batch commands will display a retry error meesage to
stderr when LIM is not available:

```
LSF daemon (LIM) not responding ... still trying
```

When LSB_API_VERBOSE=N, LSF batch commands will not display a retry error message when LIM is not available.

Default

Y. Retry message is displayed to `stderr`.

LSB_BJOBS_CONSISTENT_EXIT_CODE

Syntax

LSB_BJOBS_CONSISTENT_EXIT_CODE=Y | N

Description

When LSB_BJOBS_CONSISTENT_EXIT_CODE=Y, the `bjobs` command exits with 0 only when unfinished jobs are found, and 255 when no jobs are found, or a non-existent job ID is entered.

No jobs are running:

bjobs

```
No unfinished job found
```

echo $?

```
255
```

Job 123 does not exist:

bjobs 123

```
Job <123> is not found
```

echo $?

```
255
```

Job 111 is running:

bjobs 111

JOBID	USER	STAT	QUEUE	FROM_HOST	EXEC_HOST	JOB_NAME	SUBMIT_TIME
111	user1	RUN	normal	hostA	hostB	myjob	Oct 22 09:22

echo $?

```
0
```

Job 111 is running, and job 123 does not exist:

bjobs 111 123

JOBID	USER	STAT	QUEUE	FROM_HOST	EXEC_HOST	JOB_NAME	SUBMIT_TIME
111	user1	RUN	normal	hostA hostB myjob Oct 22 09:22			

```
Job <123> is not found
```

echo $?

```
255
```

Job 111 is finished:

bjobs 111

```
No unfinished job found
```

echo $?

```
255
```

When LSB_BJOBS_CONSISTENT_EXIT_CODE=N, the bjobs command exits with 255 only when a non-existent job ID is entered. bjobs returns 0 when no jobs are found, all jobs are finished, or if at least one job ID is valid.

No jobs are running:

bjobs

```
No unfinished job found
```

echo $?

```
0
```

Job 123 does not exist:

bjobs 123

```
Job <123> is not found
```

echo $?

```
0
```

Job 111 is running:

bjobs 111

JOBID	USER	STAT	QUEUE	FROM_HOST	EXEC_HOST	JOB_NAME	SUBMIT_TIME
111	user1	RUN	normal	hostA	hostB	myjob	Oct 22 09:22

echo $?

```
0
```

Job 111 is running, and job 123 does not exist:

bjobs 111 123

JOBID	USER	STAT	QUEUE	FROM_HOST	EXEC_HOST	JOB_NAME	SUBMIT_TIME
111	user1	RUN	normal	hostA	hostB	myjob	Oct 22 09:22

```
Job <123> is not found
```

echo $?

```
255
```

Job 111 is finished:

bjobs 111

```
No unfinished job found
```

echo $?

```
0
```

Default

N.

LSB_BLOCK_JOBINFO_TIMEOUT

Syntax

LSB_BLOCK_JOBINFO_TIMEOUT=*time_minutes*

Description

Timeout in minutes for job information query commands (e.g., bjobs).

Valid values

Any positive integer

Default

Not defined (no timeout)

See also

MAX_JOBINFO_QUERY_PERIOD in lsb.params

LSB_BPEEK_METHOD

Syntax

LSB_BPEEK_METHOD="rsh" | "lsrun"

Description

Specifies to bpeek how to get output of a remote running job.

Valid values

Specify "rsh" or "lsrun" or both, in the order you want to invoke the bpeek method.

Default

"rsh lsrun"

LSB_BPEEK_WAIT_TIME

Syntax

LSB_BPEEK_WAIT_TIME=*seconds*

Description

Defines how long the bpeek process waits to get the output of a remote running job.

Valid values

Any positive integer

Default

80 seconds

LSB_CHUNK_RUSAGE

Syntax

LSB_CHUNK_RUSAGE=y

Description

Applies only to chunk jobs. When set, sbatchd contacts PIM to retrieve resource usage information to enforce resource usage limits on chunk jobs.

By default, resource usage limits are not enforced for chunk jobs because chunk jobs are typically too short to allow LSF to collect resource usage.

If LSB_CHUNK_RUSAGE=Y is defined, limits may not be enforced for chunk jobs that take less than a minute to run.

Default

Not defined. No resource usage is collected for chunk jobs.

LSB_CMD_LOG_MASK

Syntax

LSB_CMD_LOG_MASK=*log_level*

Description

Specifies the logging level of error messages from LSF batch commands.

To specify the logging level of error messages for LSF commands, use LSF_CMD_LOG_MASK. To specify the logging level of error messages for LSF daemons, use LSF_LOG_MASK.

LSB_CMD_LOG_MASK sets the log level and is used in combination with LSB_DEBUG_CMD, which sets the log class for LSF batch commands. For example:

```
LSB_CMD_LOG_MASK=LOG_DEBUG LSB_DEBUG_CMD="LC_TRACE LC_EXEC"
```

LSF commands log error messages in different levels so that you can choose to log all messages, or only log messages that are deemed critical. The level specified by LSB_CMD_LOG_MASK determines which messages are recorded and which are discarded. All messages logged at the specified level or higher are recorded, while lower level messages are discarded.

For debugging purposes, the level LOG_DEBUG contains the fewest number of debugging messages and is used for basic debugging. The level LOG_DEBUG3 records all debugging messages, and can cause log files to grow very large; it is not often used. Most debugging is done at the level LOG_DEBUG2.

The commands log to the syslog facility unless LSB_CMD_LOGDIR is set.

Valid values

The log levels from highest to lowest are:

- LOG_EMERG
- LOG_ALERT
- LOG_CRIT
- LOG_ERR
- LOG_WARNING
- LOG_NOTICE
- LOG_INFO
- LOG_DEBUG
- LOG_DEBUG1
- LOG_DEBUG2
- LOG_DEBUG3

Default

LOG_WARNING

See also

LSB_CMD_LOGDIR, LSB_DEBUG, LSB_DEBUG_CMD, LSB_TIME_CMD,
LSF_CMD_LOGDIR, LSF_CMD_LOG_MASK, LSF_LOG_MASK, LSF_LOGDIR,
LSF_TIME_CMD

LSB_CMD_LOGDIR

Syntax

LSB_CMD_LOGDIR=*path*

Description

Specifies the path to the LSF command log files.

Default

`/tmp`

See also

LSB_CMD_LOGDIR, LSB_DEBUG, LSB_DEBUG_CMD, LSB_TIME_CMD,
LSF_CMD_LOGDIR, LSF_CMD_LOG_MASK, LSF_LOG_MASK, LSF_LOGDIR,
LSF_TIME_CMD

LSB_CPUSET_BESTCPUS

Syntax

LSB_CPUSET_BESTCPUS=y | **Y**

Description

If set, enables the best-fit algorithm for SGI cpusets

Default

Y (best-fit)

LSB_CONFDIR

Syntax

LSB_CONFDIR=*path*

Description

Specifies the path to the directory containing the LSF configuration files.

The configuration directories are installed under LSB_CONFDIR.

Configuration files for each cluster are stored in a subdirectory of LSB_CONFDIR. This subdirectory contains several files that define user and host lists, operation parameters, and queues.

All files and directories under LSB_CONFDIR must be readable from all hosts in the cluster. LSB_CONFDIR/cluster_name/configdir must be owned by the LSF administrator.

Caution:

Do not change this parameter after LSF has been installed.

Default

LSF_CONFDIR/lsbatch

See also

LSF_CONFDIR

LSB_CRDIR

Syntax

LSB_CRDIR=_path_

Description

Specifies the path and directory to the checkpointing executables on systems that support kernel-level checkpointing. LSB_CRDIR specifies the directory containing the chkpnt and restart utility programs that sbatchd uses to checkpoint or restart a job.

For example:
```
LSB_CRDIR=/usr/bin
```

If your platform supports kernel-level checkpointing, and if you want to use the utility programs provided for kernel-level checkpointing, set LSB_CRDIR to the location of the utility programs.

Default

Not defined. The system uses /bin.

LSB_DEBUG

Syntax

LSB_DEBUG=1 | 2

Description

Sets the LSF batch system to debug.

If defined, LSF runs in single user mode:

- No security checking is performed
- Daemons do not run as root

When LSB_DEBUG is defined, LSF does not look in the system services database for port numbers. Instead, it uses the port numbers defined by the parameters LSB_MBD_PORT/

LSB_SBD_PORT in lsf.conf. If these parameters are not defined, it uses port number 40000 for mbatchd and port number 40001 for sbatchd.

You should always specify 1 for this parameter unless you are testing LSF.

Can also be defined from the command line.

Valid values

LSB_DEBUG=1

The LSF system runs in the background with no associated control terminal.

LSB_DEBUG=2

The LSF system runs in the foreground and prints error messages to tty.

Default

Not defined

See also

LSB_DEBUG, LSB_DEBUG_CMD, LSB_DEBUG_MBD, LSB_DEBUG_NQS, LSB_DEBUG_SBD, LSB_DEBUG_SCH, LSF_DEBUG_LIM, LSF_DEBUG_RES, LSF_LIM_PORT, LSF_RES_PORT, LSB_MBD_PORT, LSB_SBD_PORT, LSF_LOGDIR, LSF_LIM_DEBUG, LSF_RES_DEBUG

LSB_DEBUG_CMD

Syntax

LSB_DEBUG_CMD=*log_class*

Description

Sets the debugging log class for commands and APIs.

Specifies the log class filtering to be applied to LSF batch commands or the API. Only messages belonging to the specified log class are recorded.

LSB_DEBUG_CMD sets the log class and is used in combination with LSB_CMD_LOG_MASK, which sets the log level. For example:

```
LSB_CMD_LOG_MASK=LOG_DEBUG LSB_DEBUG_CMD="LC_TRACE LC_EXEC"
```

Debugging is turned on when you define both parameters.

The daemons log to the syslog facility unless LSB_CMD_LOGDIR is defined.

To specify multiple log classes, use a space-separated list enclosed by quotation marks. For example:

```
LSB_DEBUG_CMD="LC_TRACE LC_EXEC"
```

Can also be defined from the command line.

Valid values

Valid log classes are:

• LC_ADVRSV - Log advance reservation modifications

- LC_AFS - Log AFS messages
- LC_AUTH - Log authentication messages
- LC_CHKPNT - Log checkpointing messages
- LC_COMM - Log communication messages
- LC_DCE - Log messages pertaining to DCE support
- LC_EEVENTD - Log eeventd messages
- LC_ELIM - Log ELIM messages
- LC_EXEC - Log significant steps for job execution
- LC_FAIR - Log fairshare policy messages
- LC_FILE - Log file transfer messages
- LC_HANG - Mark where a program might hang
- LC_JARRAY - Log job array messages
- LC_JLIMIT - Log job slot limit messages
- LC_LICENSE - Log license management messages (LC_LICENCE is also supported for backward compatibility)
- LC_LOADINDX - Log load index messages
- LC_M_LOG - Log multievent logging messages
- LC_MPI - Log MPI messages
- LC_MULTI - Log messages pertaining to MultiCluster
- LC_PEND - Log messages related to job pending reasons
- LC_PERFM - Log performance messages
- LC_PIM - Log PIM messages
- LC_PREEMPT - Log preemption policy messages
- LC_RESREQ - Log resource requirement messages
- LC_SIGNAL - Log messages pertaining to signals
- LC_SYS - Log system call messages
- LC_TRACE - Log significant program walk steps
- LC_XDR - Log everything transferred by XDR

Default

Not defined

See also

LSB_CMD_LOG_MASK, LSB_CMD_LOGDIR, LSB_DEBUG, LSB_DEBUG_MBD, LSB_DEBUG_NQS, LSB_DEBUG_SBD, LSB_DEBUG_SCH, LSF_DEBUG_LIM, LSF_DEBUG_RES, LSF_LIM_PORT, LSF_RES_PORT, LSB_MBD_PORT, LSB_SBD_PORT, LSF_LOGDIR, LSF_LIM_DEBUG, LSF_RES_DEBUG

LSB_DEBUG_MBD

Syntax

LSB_DEBUG_MBD=*log_class*

Description

Sets the debugging log class for mbatchd.

Specifies the log class filtering to be applied to mbatchd. Only messages belonging to the specified log class are recorded.

LSB_DEBUG_MBD sets the log class and is used in combination with LSF_LOG_MASK, which sets the log level. For example:

```
LSF_LOG_MASK=LOG_DEBUG LSB_DEBUG_MBD="LC_TRACE LC_EXEC"
```

To specify multiple log classes, use a space-separated list enclosed in quotation marks. For example:

```
LSB_DEBUG_MBD="LC_TRACE LC_EXEC"
```

You need to restart the daemons after setting LSB_DEBUG_MBD for your changes to take effect.

If you use the command badmin mbddebug to temporarily change this parameter without changing lsf.conf, you do not need to restart the daemons.

Valid values

Valid log classes are the same as for LSB_DEBUG_CMD except for the log class LC_ELIM, which cannot be used with LSB_DEBUG_MBD. See LSB_DEBUG_CMD.

Default

Not defined

See also

LSB_CMD_LOG_MASK, LSB_CMD_LOGDIR, LSB_DEBUG, LSB_DEBUG_MBD, LSB_DEBUG_NQS, LSB_DEBUG_SBD, LSB_DEBUG_SCH, LSF_DEBUG_LIM, LSF_DEBUG_RES, LSF_LIM_PORT, LSF_RES_PORT, LSB_MBD_PORT, LSB_SBD_PORT, LSF_LOGDIR, LSF_LIM_DEBUG, LSF_RES_DEBUG

LSB_DEBUG_NQS

Syntax

LSB_DEBUG_NQS=_log_class_

Description

Sets the log class for debugging the NQS interface.

Specifies the log class filtering to be applied to NQS. Only messages belonging to the specified log class are recorded.

LSB_DEBUG_NQS sets the log class and is used in combination with LSF_LOG_MASK, which sets the log level. For example:

```
LSF_LOG_MASK=LOG_DEBUG LSB_DEBUG_NQS="LC_TRACE LC_EXEC"
```

Debugging is turned on when you define both parameters.

To specify multiple log classes, use a space-separated list enclosed in quotation marks. For example:

```
LSB_DEBUG_NQS="LC_TRACE LC_EXEC"
```

This parameter can also be defined from the command line.

Valid values

For a list of valid log classes, see LSB_DEBUG_CMD.

Default

Not defined

See also

LSB_DEBUG_CMD, LSF_CMD_LOGDIR, LSF_CMD_LOG_MASK, LSF_LOG_MASK, LSF_LOGDIR

LSB_DEBUG_SBD

Syntax

LSB_DEBUG_SBD=*log_class*

Description

Sets the debugging log class for sbatchd.

Specifies the log class filtering to be applied to sbatchd. Only messages belonging to the specified log class are recorded.

LSB_DEBUG_SBD sets the log class and is used in combination with LSF_LOG_MASK, which sets the log level. For example:

```
LSF_LOG_MASK=LOG_DEBUG  LSB_DEBUG_SBD="LC_TRACE LC_EXEC"
```

To specify multiple log classes, use a space-separated list enclosed in quotation marks. For example:

```
LSB_DEBUG_SBD="LC_TRACE LC_EXEC"
```

You need to restart the daemons after setting LSB_DEBUG_SBD for your changes to take effect.

If you use the command badmin sbddebug to temporarily change this parameter without changing lsf.conf, you do not need to restart the daemons.

Valid values

Valid log classes are the same as for LSB_DEBUG_CMD except for the log class LC_ELIM, which cannot be used with LSB_DEBUG_SBD. See LSB_DEBUG_CMD.

Default

Not defined

See also

LSB_DEBUG_MBD, LSF_CMD_LOGDIR, LSF_CMD_LOG_MASK, LSF_LOG_MASK, LSF_LOGDIR, badmin

LSB_DEBUG_SCH

Syntax

LSB_DEBUG_SCH=*log_class*

Description

Sets the debugging log class for mbschd.

Specifies the log class filtering to be applied to mbschd. Only messages belonging to the specified log class are recorded.

LSB_DEBUG_SCH sets the log class and is used in combination with LSF_LOG_MASK, which sets the log level. For example:

```
LSF_LOG_MASK=LOG_DEBUG LSB_DEBUG_SCH="LC_SCHED"
```

To specify multiple log classes, use a space-separated list enclosed in quotation marks. For example:

```
LSB_DEBUG_SCH="LC_SCHED LC_TRACE LC_EXEC"
```

You need to restart the daemons after setting LSB_DEBUG_SCH for your changes to take effect.

Valid values

Valid log classes are the same as for LSB_DEBUG_CMD except for the log class LC_ELIM, which cannot be used with LSB_DEBUG_SCH, and LC_HPC and LC_SCHED, which are only valid for LSB_DEBUG_SCH. See LSB_DEBUG_CMD.

Default

Not defined

See also

LSB_DEBUG_MBD, LSB_DEBUG_SBD, LSF_CMD_LOGDIR, LSF_CMD_LOG_MASK, LSF_LOG_MASK, LSF_LOGDIR, badmin

LSB_DISABLE_LIMLOCK_EXCL

Syntax

LSB_DISABLE_LIMLOCK_EXCL=y | n

Description

If preemptive scheduling is enabled, this parameter enables preemption of and preemption by exclusive jobs when PREEMPT_JOBTYPE=EXCLUSIVE in lsb.params. Changing this parameter requires a restart of all sbatchds in the cluster (badmin hrestart). Do not change this parameter while exclusive jobs are running.

When LSB_DISABLE_LIMLOCK_EXCL=y, for a host running an exclusive job:

- LIM is not locked on a host running an exclusive job
- lsload displays the host status ok.
- bhosts displays the host status closed.
- Users can run tasks on the host using lsrun or lsgrun. To prevent users from running tasks during execution of an exclusive job, the parameter LSF_DISABLE_LSRUN=y must be defined in lsf.conf.

Default

n. LSF locks the LIM on a host running an exclusive job and unlocks the LIM when the exclusive job finishes.

LSB_DISABLE_RERUN_POST_EXEC

Syntax

LSB_DISABLE_RERUN_POST_EXEC=y | Y

Description

If set, and the job is rerunnable, the POST_EXEC configured in the queue is not executed if the job is rerun.

Running of post-execution commands upon restart of a rerunnable job may not always be desirable. For example, if the post-exec removes certain files, or does other cleanup that should only happen if the job finishes successfully, use LSB_DISABLE_RERUN_POST_EXEC to prevent the post-exec from running and allow the successful continuation of the job when it reruns.

Default

Not defined

LSB_ECHKPNT_KEEP_OUTPUT

Syntax

LSB_ECHKPNT_KEEP_OUTPUT=y | Y

Description

Saves the standard output and standard error of custom `echkpnt` and `erestart` methods to:

- `checkpoint_dir/$LSB_JOBID/echkpnt.out`
- `checkpoint_dir/$LSB_JOBID/echkpnt.err`
- `checkpoint_dir/$LSB_JOBID/erestart.out`
- `checkpoint_dir/$LSB_JOBID/erestart.err`

Can also be defined as an environment variable.

Default

Not defined. Standard error and standard output messages from custom `echkpnt` and `erestart` programs is directed to `/dev/null` and discarded by LSF.

See also

LSB_ECHKPNT_METHOD, LSB_ECHKPNT_METHOD_DIR

LSB_ECHKPNT_METHOD

Syntax

LSB_ECHKPNT_METHOD="*method_name* [*method_name*] ..."

Description

Name of custom `echkpnt` and `erestart` methods.

Can also be defined as an environment variable, or specified through the `bsub -k` option.

The name you specify here is used for both your custom `echkpnt` and `erestart` programs. You must assign your custom `echkpnt` and `erestart` programs the name `echkpnt.`*`method_name`* and `erestart.`*`method_name`*. The programs `echkpnt.`*`method_name`* and `erestart.`*`method_name`*. must be in LSF_SERVERDIR or in the directory specified by LSB_ECHKPNT_METHOD_DIR.

Do not define LSB_ECHKPNT_METHOD=default as default is a reserved keyword to indicate to use the default `echkpnt` and `erestart` methods of LSF. You can however, specify `bsub -k "my_dir method=default" my_job` to indicate that you want to use the default checkpoint and restart methods.

When this parameter is not defined in `lsf.conf` or as an environment variable and no custom method is specified at job submission through `bsub -k`, LSF uses `echkpnt.default` and `erestart.default` to checkpoint and restart jobs.

When this parameter is defined, LSF uses the custom checkpoint and restart methods specified.

Limitations

The method name and directory (LSB_ECHKPNT_METHOD_DIR) combination must be unique in the cluster.

For example, you may have two `echkpnt` applications with the same name such as `echkpnt.mymethod` but what differentiates them is the different directories defined with LSB_ECHKPNT_METHOD_DIR. It is the cluster administrator's responsibility to ensure that method name and method directory combinations are unique in the cluster.

Default

Not defined. LSF uses `echkpnt.default` and `erestart.default` to checkpoint and restart jobs

See also

LSB_ECHKPNT_METHOD_DIR, LSB_ECHKPNT_KEEP_OUTPUT

LSB_ECHKPNT_METHOD_DIR

Syntax

LSB_ECHKPNT_METHOD_DIR=*path*

Description

Absolute path name of the directory in which custom `echkpnt` and `erestart` programs are located.

The checkpoint method directory should be accessible by all users who need to run the custom `echkpnt` and `erestart` programs.

Can also be defined as an environment variable.

Default

Not defined. LSF searches in LSF_SERVERDIR for custom echkpnt and erestart programs.

See also

LSB_ESUB_METHOD, LSB_ECHKPNT_KEEP_OUTPUT

LSB_ESUB_METHOD

Syntax

LSB_ESUB_METHOD="*esub_application* [*esub_application*] ...**"**

Description

Specifies a mandatory esub that applies to all job submissions. LSB_ESUB_METHOD lists the names of the application-specific esub executables used in addition to any executables specified by the bsub -a option.

For example, LSB_ESUB_METHOD="dce fluent" runs LSF_SERVERDIR/esub.dce and LSF_SERVERDIR/esub.fluent for all jobs submitted to the cluster. These esubs define, respectively, DCE as the mandatory security system and FLUENT as the mandatory application for all jobs.

LSB_ESUB_METHOD can also be defined as an environment variable.

The value of LSB_ESUB_METHOD must correspond to an actual esub file. For example, to use LSB_ESUB_METHOD=fluent, the file esub.fluent must exist in LSF_SERVERDIR.

The name of the esub program must be a valid file name. Valid file names contain only alphanumeric characters, underscore (_) and hyphen (-).

Restriction:

The name esub.user is reserved. Do not use the name esub.user for an application-specific esub.

The master esub (mesub) uses the name you specify to invoke the appropriate esub program. The esub and esub.*esub_application* programs must be located in LSF_SERVERDIR.

LSF does not detect conflicts based on esub names. For example, if LSB_ESUB_METHOD="openmpi" and **bsub -a pvm** is specified at job submission, the job could fail because these esubs define two different types of parallel job handling.

Default

Not defined. LSF does not apply a mandatory esub to jobs submitted to the cluster.

LSB_INTERACT_MSG_ENH

Syntax

LSB_INTERACT_MSG_ENH=y | Y

Description

If set, enables enhanced messaging for interactive batch jobs. To disable interactive batch job messages, set LSB_INTERACT_MSG_ENH to any value other than y or Y; for example, LSB_INTERACT_MSG_ENH=N.

Default

Not defined

See also

LSB_INTERACT_MSG_INTVAL

LSB_INTERACT_MSG_INTVAL

Syntax

LSB_INTERACT_MSG_INTVAL=*time_seconds*

Description

Specifies the update interval in seconds for interactive batch job messages. LSB_INTERACT_MSG_INTVAL is ignored if LSB_INTERACT_MSG_ENH is not set.

Job information that LSF uses to get the pending or suspension reason is updated according to the value of PEND_REASON_UPDATE_INTERVAL in lsb.params.

Default

Not defined. If LSB_INTERACT_MSG_INTVAL is set to an incorrect value, the default update interval is 60 seconds.

See also

LSB_INTERACT_MSG_ENH

LSB_JOBID_DISP_LENGTH

Syntax

LSB_JOBID_DISP_LENGTH=*integer*

Description

By default, LSF commands bjobs and bhist display job IDs with a maximum length of 7 characters. Job IDs greater than 9999999 are truncated on the left.

When LSB_JOBID_DISP_LENGTH=10, the width of the JOBID column in bjobs and bhist increases to 10 characters.

Valid values

Specify an integer between 7 and 10.

Default

Not defined. LSF uses the default 7-character length for job ID display.

LSB_JOB_CPULIMIT

Syntax

LSB_JOB_CPULIMIT=y | n

Description

Determines whether the CPU limit is a per-process limit enforced by the OS or whether it is a per-job limit enforced by LSF:

- The per-process limit is enforced by the OS when the CPU time of one process of the job exceeds the CPU limit.
- The per-job limit is enforced by LSF when the total CPU time of all processes of the job exceed the CPU limit.

This parameter applies to CPU limits set when a job is submitted with bsub -c, and to CPU limits set for queues by CPULIMIT in lsb.queues.

- LSF-enforced per-job limit: When the sum of the CPU time of all processes of a job exceed the CPU limit, LSF sends a SIGXCPU signal (where supported by the operating system) from the operating system to all processes belonging to the job, then SIGINT, SIGTERM and SIGKILL. The interval between signals is 10 seconds by default. The time interval between SIGXCPU, SIGINT, SIGKILL, SIGTERM can be configured with the parameter JOB_TERMINATE_INTERVAL in lsb.params.

 Restriction:

 SIGXCPU is not supported by Windows.

- OS-enforced per process limit: When one process in the job exceeds the CPU limit, the limit is enforced by the operating system. For more details, refer to your operating system documentation for setrlimit().

The setting of LSB_JOB_CPULIMIT has the following effect on how the limit is enforced:

When LSB_JOB_CPULIMIT is	LSF-enforced per-job limit	OS-enforced per-process limit
y	Enabled	Disabled
n	Disabled	Enabled
Not defined	Enabled	Enabled

Default

Not defined

Notes

To make LSB_JOB_CPULIMIT take effect, use the command badmin hrestart all to restart all sbatchds in the cluster.

Changing the default Terminate job control action: You can define a different terminate action in lsb.queues with the parameter JOB_CONTROLS if you do not want the job to be killed. For more details on job controls, see *Administering Platform LSF*.

Limitations

If a job is running and the parameter is changed, LSF is not able to reset the type of limit enforcement for running jobs.

- If the parameter is changed from per-process limit enforced by the OS to per-job limit enforced by LSF (LSB_JOB_CPULIMIT=n changed to LSB_JOB_CPULIMIT=y), both per-process limit and per-job limit affect the running job. This means that signals may be sent to the job either when an individual process exceeds the CPU limit or the sum of the CPU time of all processes of the job exceed the limit. A job that is running may be killed by the OS or by LSF.
- If the parameter is changed from per-job limit enforced by LSF to per-process limit enforced by the OS (LSB_JOB_CPULIMIT=y changed to LSB_JOB_CPULIMIT=n), the job is allowed to run without limits because the per-process limit was previously disabled.

See also

lsb.queues, bsub, JOB_TERMINATE_INTERVAL in lsb.params, LSB_MOD_ALL_JOBS

LSB_JOB_MEMLIMIT

Syntax

LSB_JOB_MEMLIMIT=y | n

Description

Determines whether the memory limit is a per-process limit enforced by the OS or whether it is a per-job limit enforced by LSF.

- The per-process limit is enforced by the OS when the memory allocated to one process of the job exceeds the memory limit.
- The per-job limit is enforced by LSF when the sum of the memory allocated to all processes of the job exceeds the memory limit.

This parameter applies to memory limits set when a job is submitted with bsub -M *mem_limit*, and to memory limits set for queues with MEMLIMIT in lsb.queues.

The setting of LSB_JOB_MEMLIMIT has the following effect on how the limit is enforced:

When LSB_JOB_MEMLIMIT is	LSF-enforced per-job limit	OS-enforced per-process limit
y	Enabled	Disabled
n or not defined	Disabled	Enabled

When LSB_JOB_MEMLIMIT is Y, the LSF-enforced per-job limit is enabled, and the OS-enforced per-process limit is disabled.

When LSB_JOB_MEMLIMIT is N or not defined, the LSF-enforced per-job limit is disabled, and the OS-enforced per-process limit is enabled.

LSF-enforced per-job limit: When the total memory allocated to all processes in the job exceeds the memory limit, LSF sends the following signals to kill the job: SIGINT, SIGTERM, then SIGKILL. The interval between signals is 10 seconds by default.

On UNIX, the time interval between SIGINT, SIGKILL, SIGTERM can be configured with the parameter JOB_TERMINATE_INTERVAL in lsb.params.

OS-enforced per process limit: When the memory allocated to one process of the job exceeds the memory limit, the operating system enforces the limit. LSF passes the memory limit to the operating system. Some operating systems apply the memory limit to each process, and some do not enforce the memory limit at all.

OS memory limit enforcement is only available on systems that support RLIMIT_RSS for setrlimit().

The following operating systems do not support the memory limit at the OS level and the job is allowed to run without a memory limit:

- Windows
- Sun Solaris 2.x

Default

Not defined. Per-process memory limit enforced by the OS; per-job memory limit enforced by LSF disabled

Notes

To make LSB_JOB_MEMLIMIT take effect, use the command badmin hrestart all to restart all sbatchds in the cluster.

If LSB_JOB_MEMLIMIT is set, it overrides the setting of the parameter LSB_MEMLIMIT_ENFORCE. The parameter LSB_MEMLIMIT_ENFORCE is ignored.

The difference between LSB_JOB_MEMLIMIT set to y and LSB_MEMLIMIT_ENFORCE set to y is that with LSB_JOB_MEMLIMIT, only the per-job memory limit enforced by LSF is enabled. The per-process memory limit enforced by the OS is disabled. With LSB_MEMLIMIT_ENFORCE set to y, both the per-job memory limit enforced by LSF and the per-process memory limit enforced by the OS are enabled.

Changing the default Terminate job control action: You can define a different Terminate action in lsb.queues with the parameter JOB_CONTROLS if you do not want the job to be killed. For more details on job controls, see *Administering Platform LSF*.

Limitations

If a job is running and the parameter is changed, LSF is not able to reset the type of limit enforcement for running jobs.

- If the parameter is changed from per-process limit enforced by the OS to per-job limit enforced by LSF (LSB_JOB_MEMLIMIT=n or not defined changed to LSB_JOB_MEMLIMIT=y), both per-process limit and per-job limit affect the running job. This means that signals may be sent to the job either when the memory allocated to an individual process exceeds the memory limit or the sum of memory allocated to all processes of the job exceed the limit. A job that is running may be killed by LSF.
- If the parameter is changed from per-job limit enforced by LSF to per-process limit enforced by the OS (LSB_JOB_MEMLIMIT=y changed to LSB_JOB_MEMLIMIT=n or

not defined), the job is allowed to run without limits because the per-process limit was previously disabled.

See also

LSB_MEMLIMIT_ENFORCE, LSB_MOD_ALL_JOBS, lsb.queues, bsub, JOB_TERMINATE_INTERVAL in lsb.params

LSB_KEEP_SYSDEF_RLIMIT

Syntax

LSB_KEEP_SYSDEF_RLIMIT=y | **n**

Description

If resource limits are configured for a user in the SGI IRIX User Limits Database (ULDB) domain specified in LSF_ULDB_DOMAIN, and there is no domain default, the system default is honored.

If LSB_KEEP_SYSDEF_RLIMIT=n, and no resource limits are configured in the domain for the user and there is no domain default, LSF overrides the system default and sets system limits to unlimited.

Default

Not defined. No resource limits are configured in the domain for the user and there is no domain default.

LSB_LOAD_TO_SERVER_HOSTS (OBSOLETE)

Syntax

LSB_LOAD_TO_SERVER_HOSTS=Y | **y**

Description

Note:

This parameter is obsolete in LSF 7 Update 2. By default, client sbatchd contacts the local LIM for host status and load information.

Highly recommended for large clusters to decrease the load on the master LIM. Forces the client sbatchd to contact the local LIM for host status and load information. The client sbatchd only contacts the master LIM or a LIM on one of the LSF_SERVER_HOSTS if sbatchd cannot find the information locally.

Default

Y. Client sbatchd contacts the local LIM for host status and load information.

See also

LSF_SERVER_HOSTS in slave.config

LSB_LOCALDIR

Syntax

LSB_LOCALDIR=*path*

Description

Enables duplicate logging.

Specify the path to a local directory that exists only on the first LSF master host. LSF puts the primary copies of the event and accounting log files in this directory. LSF puts the duplicates in LSB_SHAREDIR.

> **Important:**
>
> Always restart both the mbactchd and sbatchd when modifying LSB_LOCALDIR.

Example

```
LSB_LOCALDIR=/usr/share/lsbatch/loginfo
```

Default

Not defined

See also

LSB_SHAREDIR, EVENT_UPDATE_INTERVAL in lsb.params

LSB_MAILPROG

Syntax

LSB_MAILPROG=*file_name*

Description

Path and file name of the mail program used by LSF to send email. This is the electronic mail program that LSF uses to send system messages to the user. When LSF needs to send email to users it invokes the program defined by LSB_MAILPROG in lsf.conf. You can write your own custom mail program and set LSB_MAILPROG to the path where this program is stored.

LSF administrators can set the parameter as part of cluster reconfiguration. Provide the name of any mail program. For your convenience, LSF provides the sendmail mail program, which supports the sendmail protocol on UNIX.

In a mixed cluster, you can specify different programs for Windows and UNIX. You can set this parameter during installation on Windows. For your convenience, LSF provides the lsmail.exe mail program, which supports SMTP and Microsoft Exchange Server protocols on Windows. If lsmail is specified, the parameter LSB_MAILSERVER must also be specified.

If you change your mail program, the LSF administrator must restart sbatchd on all hosts to retrieve the new value.

UNIX

By default, LSF uses `/usr/lib/sendmail` to send email to users. LSF calls LSB_MAILPROG with two arguments; one argument gives the full name of the sender, and the other argument gives the return address for mail.

LSB_MAILPROG must read the body of the mail message from the standard input. The end of the message is marked by end-of-file. Any program or shell script that accepts the arguments and input, and delivers the mail correctly, can be used.

LSB_MAILPROG must be executable by any user.

Windows

If LSB_MAILPROG is not defined, no email is sent.

Examples

```
LSB_MAILPROG=lsmail.exe
```

```
LSB_MAILPROG=/serverA/tools/lsf/bin/unixhost.exe
```

Default

`/usr/lib/sendmail` (UNIX)

blank (Windows)

See also

LSB_MAILSERVER, LSB_MAILTO

LSB_MAILSERVER

Syntax

LSB_MAILSERVER=_mail_protocol_:_mail_server_

Description

Part of mail configuration on Windows.

This parameter only applies when `lsmail` is used as the mail program (LSB_MAILPROG=lsmail.exe).Otherwise, it is ignored.

Both *mail_protocol* and *mail_server* must be indicated.

Set this parameter to either SMTP or Microsoft Exchange protocol (SMTP or EXCHANGE) and specify the name of the host that is the mail server.

This parameter is set during installation of LSF on Windows or is set or modified by the LSF administrator.

If this parameter is modified, the LSF administrator must restart sbatchd on all hosts to retrieve the new value.

Examples

```
LSB_MAILSERVER=EXCHANGE: Host2@company.com
```

```
LSB_MAILSERVER=SMTP: MailHost
```

Default

Not defined

See also

LSB_LOCALDIR

LSB_MAILSIZE_LIMIT

Syntax

LSB_MAILSIZE_LIMIT=*email_size_KB*

Description

Limits the size in KB of the email containing job output information.

The system sends job information such as CPU, process and memory usage, job output, and errors in email to the submitting user account. Some batch jobs can create large amounts of output. To prevent large job output files from interfering with your mail system, use LSB_MAILSIZE_LIMIT to set the maximum size in KB of the email containing the job information. Specify a positive integer.

If the size of the job output email exceeds LSB_MAILSIZE_LIMIT, the output is saved to a file under JOB_SPOOL_DIR or to the default job output directory if JOB_SPOOL_DIR is not defined. The email informs users of where the job output is located.

If the -o option of bsub is used, the size of the job output is not checked against LSB_MAILSIZE_LIMIT.

If you use a custom mail program specified by the LSB_MAILPROG parameter that can use the LSB_MAILSIZE environment variable, it is not necessary to configure LSB_MAILSIZE_LIMIT.

Default

By default, LSB_MAILSIZE_LIMIT is not enabled. No limit is set on size of batch job output email.

See also

LSB_MAILPROG, LSB_MAILTO

LSB_MAILTO

Syntax

LSB_MAILTO=*mail_account*

Description

LSF sends electronic mail to users when their jobs complete or have errors, and to the LSF administrator in the case of critical errors in the LSF system. The default is to send mail to the user who submitted the job, on the host on which the daemon is running; this assumes that your electronic mail system forwards messages to a central mailbox.

The LSB_MAILTO parameter changes the mailing address used by LSF. LSB_MAILTO is a format string that is used to build the mailing address.

Common formats are:

- !U: Mail is sent to the submitting user's account name on the local host. The substring !U, if found, is replaced with the user's account name.
- !U@company_name.com: Mail is sent to user@company_name.com on the mail server. The mail server is specified by LSB_MAILSERVER.
- !U@!H: Mail is sent to *user@submission_hostname*. The substring !H is replaced with the name of the submission host. This format is valid on UNIX only. It is not supported on Windows.

All other characters (including any other '!') are copied exactly.

If this parameter is modified, the LSF administrator must restart sbatchd on all hosts to retrieve the new value.

Windows only: When a job exception occurs (for example, a job is overrun or underrun), an email is sent to the primary administrator set in the lsf.cluster.*cluster_name* file to the doman set in LSB_MAILTO. For example, if the primary administrator is lsfadmin and LSB_MAILTO=fred@company.com, an email is sent to lsfadmin@company.com. The email must be a valid Windows email account.

Default

!U

See also

LSB_MAILPROG, LSB_MAILSIZE_LIMIT

LSB_MAX_ASKED_HOSTS_NUMBER

Syntax

LSB_MAX_ASKED_HOSTS_NUMBER=integer

Description

Limits the number of hosts a user can specify with the -m (host preference) option of the following commands:

- bsub
- brun
- bmod
- brestart
- brsvadd
- brsvmod
- brsvs

The job is rejected if more hosts are specified than the value of LSB_MAX_ASKED_HOSTS_NUMBER.

Caution:

> If this value is set high, there will be a performance effect if users submit or modify jobs using the -m option and specify a large number of hosts. 512 hosts is the suggested upper limit.

Valid values

Any whole, positive integer.

Default

512

LSB_MAX_JOB_DISPATCH_PER_SESSION

Syntax

LSB_MAX_JOB_DISPATCH_PER_SESSION=*integer*

Description

Defines the maximum number of jobs that mbatchd can dispatch during one job scheduling session.

Both mbatchd and sbatchd must be restarted when you change the value of this parameter.

If set to a value greater than 300, the file descriptor limit is increased on operating systems that support a file descriptor limit greater than 1024.

Use together with MAX_SBD_CONNS in `lsb.params`. Set LSB_MAX_JOB_DISPATCH_PER_SESSION to a value no greater than one-half the value of MAX_SBD_CONNS. This setting configures mbatchd to dispatch jobs at a high rate while maintaining the processing speed of other mbatchd tasks.

Examples

LSB_MAX_JOB_DISPATCH_PER_SESSION=300

The file descriptor limit is 1024.

LSB_MAX_JOB_DISPATCH_PER_SESSION=1000

The file descriptor limit is greater than 1024 on operating systems that support a greater limit.

Default

300

See also

MAX_SBD_CONNS in lsb.params

LSB_MAX_PROBE_SBD

Syntax

LSB_MAX_PROBE_SBD=*integer*

Description

Specifies the maximum number of sbatchd instances can be polled by mbatchd in the interval MBD_SLEEP_TIME/10 (6 seconds by default). Use this parameter in large clusters to reduce the time it takes for mbatchd to probe all sbatchds.

The value of LSB_MAX_PROBE_SBD cannot be greater than the number of hosts in the cluster. If it is, mbatchd adjusts the value of LSB_MAX_PROBE_SBD to be same as the number of hosts.

After modifying LSB_MAX_PROBE_SBD, use `badmin mbdrestart` to restart mbatchd and let the modified value take effect.

If LSB_MAX_PROBE_SBD is defined, the value of MAX_SBD_FAIL in `lsb.params` can be less than 3.

Valid values

Any positive integer between 0 and 64

Default

20

See also

MAX_SBD_FAIL in lsb.params

LSB_MAX_NQS_QUEUES

Syntax

LSB_MAX_NQS_QUEUES=_nqs_queues_

Description

The maximum number of NQS queues allowed in the LSF cluster. Required for LSF to work with NQS. You must restart mbatchd if you change the value of LSB_MAX_NQS_QUEUES.

The total number of NQS queues configured by NQS_QUEUES in `lsb.queues` cannot exceed the value of LSB_MAX_NQS_QUEUES. NQS queues in excess of the maximum queues are ignored.

If you do not define LSB_MAX_NQS_QUEUES or define an incorrect value, LSF-NQS interoperation is disabled.

Valid values

Any positive integer

Default

None

LSB_MBD_BUSY_MSG

Syntax

LSB_MBD_BUSY_MSG="*message_string***"**

Description

Specifies the message displayed when mbatchd is too busy to accept new connections or respond to client requests.

Define this parameter if you want to customize the message.

Valid values

String, either non-empty or empty.

Default

Not defined. By default, LSF displays the message `"LSF is processing your request. Please wait..."`

Batch commands retry the connection to mbatchd at the intervals specified by the parameters LSB_API_CONNTIMEOUT and LSB_API_RECVTIMEOUT.

LSB_MBD_CONNECT_FAIL_MSG

Syntax

LSB_MBD_CONNECT_FAIL_MSG="*message_string***"**

Description

Specifies the message displayed when internal system connections to mbatchd fail.

Define this parameter if you want to customize the message.

Valid values

String, either non-empty or empty.

Default

Not defined. By default, LSF displays the message `"Cannot connect to LSF. Please wait..."`

Batch commands retry the connection to mbatchd at the intervals specified by the parameters LSB_API_CONNTIMEOUT and LSB_API_RECVTIMEOUT.

LSB_MBD_DOWN_MSG

Syntax

LSB_MBD_DOWN_MSG="*message_string***"**

Description

Specifies the message displayed by the `bhosts` command when mbatchd is down or there is no process listening at either the LSB_MBD_PORT or the LSB_QUERY_PORT.

Define this parameter if you want to customize the message.

Valid values

String, either non-empty or empty.

Default

Not defined. By default, LSF displays the message `"LSF is down. Please wait..."`

Batch commands retry the connection to mbatchd at the intervals specified by the parameters LSB_API_CONNTIMEOUT and LSB_API_RECVTIMEOUT.

LSB_MBD_MAX_SIG_COUNT

Syntax

LSB_MBD_MAX_SIG_COUNT=*integer*

Description

When a host enters an unknown state, the mbatchd attempts to retry any pending jobs. This parameter specifies the maximum number of pending signals that the `mbatchd` deals with concurrently in order not to overload it. A high value for LSB_MBD_MAX_SIG_COUNT can negatively impact the performance of your cluster.

Valid Valid values

Integers between 5-100, inclusive.

Default

5

LSB_MBD_PORT

See LSF_LIM_PORT, LSF_RES_PORT, LSB_MBD_PORT, LSB_SBD_PORT.

LSB_MC_CHKPNT_RERUN

Syntax

LSB_MC_CHKPNT_RERUN=y | n

Description

For checkpointable MultiCluster jobs, if a restart attempt fails, the job is rerun from the beginning (instead of from the last checkpoint) without administrator or user intervention.

The submission cluster does not need to forward the job again. The execution cluster reports the job's new pending status back to the submission cluster, and the job is dispatched to the same host to restart from the beginning

Default

n

LSB_MC_INITFAIL_MAIL

Syntax

LSB_MC_INITFAIL_MAIL=Y | All | Administrator

Description

MultiCluster job forwarding model only.

Specify Y to make LSF email the job owner when a job is suspended after reaching the retry threshold.

Specify **Administrator** to make LSF email the primary administrator when a job is suspended after reaching the retry threshold.

Specify All to make LSF email both the job owner and the primary administrator when a job is suspended after reaching the retry threshold.

Default

not defined

LSB_MC_INITFAIL_RETRY

Syntax

LSB_MC_INITFAIL_RETRY=_integer_

Description

MultiCluster job forwarding model only. Defines the retry threshold and causes LSF to suspend a job that repeatedly fails to start. For example, specify 2 retry attempts to make LSF attempt to start a job 3 times before suspending it.

Default

5

LSB_MEMLIMIT_ENFORCE

Syntax

LSB_MEMLIMIT_ENFORCE=y | n

Description

Specify y to enable LSF memory limit enforcement.

If enabled, LSF sends a signal to kill all processes that exceed queue-level memory limits set by MEMLIMIT in lsb.queues or job-level memory limits specified by bsub -M _mem_limit._

Otherwise, LSF passes memory limit enforcement to the OS. UNIX operating systems that support RLIMIT_RSS for `setrlimit()` can apply the memory limit to each process.

The following operating systems do not support memory limit at the OS level:

- Windows
- Sun Solaris 2.x

Default

Not defined. LSF passes memory limit enforcement to the OS.

See also

lsb.queues

LSB_MIG2PEND

Syntax

LSB_MIG2PEND=0 | 1

Description

Applies only to migrating checkpointable or rerunnable jobs.

When defined with a value of 1, requeues migrating jobs instead of restarting or rerunning them on the first available host. Requeues the jobs in the PEND state in order of the original submission time and with the original job priority.

If you want to place the migrated jobs at the bottom of the queue without considering submission time, define both LSB_MIG2PEND=1 and LSB_REQUEUE_TO_BOTTOM=1 in `lsf.conf.`

Ignored in a MultiCluster environment.

Default

Not defined. LSF restarts or reruns migrating jobs on the first available host.

See also

LSB_REQUEUE_TO_BOTTOM

LSB_MIXED_PATH_DELIMITER

Syntax

LSB_MIXED_PATH_DELIMITER="|"

Description

Defines the delimiter between UNIX and Windows paths if LSB_MIXED_PATH_ENABLE=y. For example, `/home/tmp/J.out|c:\tmp\J.out.`

Default

A pipe "|" is the default delimiter.

See also

LSB_MIXED_PATH_ENABLE

LSB_MIXED_PATH_ENABLE

Syntax

LSB_MIXED_PATH_ENABLE=y │ **n**

Description

Allows you to specify both a UNIX and Windows path when submitting a job in a mixed cluster (both Windows and UNIX hosts).

The format is always **unix_path_cmd|windows_path_cmd**.

Applies to the following options of bsub:

- -o, -oo
- -e, -eo
- -i, -is
- -cwd
- -E, -Ep
- CMD
- queue level PRE_EXEC, POST_EXEC
- application level PRE_EXEC, POST_EXEC

For example:

```
bsub -o "/home/tmp/job%J.out|c:\tmp\job%J.out" -e "/home/tmp/err%
J.out|c:\tmp\err%J.out" -E "sleep 9|sleep 8" -Ep "sleep 7|sleep 6"
-cwd "/home/tmp|c:\tmp" "sleep 121|sleep 122"
```

The delimiter is configurable: LSB_MIXED_PATH_DELIMITER.

Default

Not defined. LSF jobs submitted .

See also

LSB_MIXED_PATH_DELIMITER

LSB_MOD_ALL_JOBS

Syntax

LSB_MOD_ALL_JOBS=y │ **Y**

Description

If set, enables bmod to modify resource limits and location of job output files for running jobs.

After a job has been dispatched, the following modifications can be made:

- CPU limit (-c [*hour*:]*minute*[/*host_name* | /*host_model*] | -cn)
- Memory limit (-M *mem_limit* | -Mn)

- Rerunnable jobs (-r | -rn)
- Resource requirements (-R "*res_req*" except -R "cu[*cu_string*]")
- Run limit (-W *run_limit*[/*host_name* | /*host_model*] | -Wn)
- Standard output file name (-o *output_file* | -on)
- Standard error file name (-e *error_file* | -en)
- Overwrite standard output (stdout) file name up to 4094 characters for UNIX or 255 characters for Windows (-oo *output_file*)
- Overwrite standard error (stderr) file name up to 4094 characters for UNIX or 255 characters for Windows (-eo *error_file*)

To modify the CPU limit or the memory limit of running jobs, the parameters LSB_JOB_CPULIMIT=Y and LSB_JOB_MEMLIMIT=Y must be defined in lsf.conf.

Important:

Always run badmin mbdrestart after modifying LSB_MOD_ALL_JOBS.

Default

Not defined

See also

LSB_JOB_CPULIMIT, LSB_JOB_MEMLIMIT

LSB_NCPU_ENFORCE

Description

When set to 1, enables parallel fairshare and considers the number of CPUs when calculating dynamic priority for queue-level user-based fairshare. LSB_NCPU_ENFORCE does not apply to host-partition user-based fairshare. For host-partition user-based fairshare, the number of CPUs is automatically considered.

Default

Not defined

LSB_NQS_PORT

Syntax

LSB_NQS_PORT=*port_number*

Description

Required for LSF to work with NQS.

TCP service port to use for communication with NQS.

Where defined

This parameter can alternatively be set as an environment variable or in the services database such as /etc/services.

Example

LSB_NQS_PORT=607

Default

Not defined

LSB_NUM_NIOS_CALLBACK_THREADS

Syntax

LSB_NUM_NIOS_CALLBACK_THREADS=*integer*

Description

Specifies the number of callback threads to use for batch queries.

If your cluster runs a large amount of blocking mode (bsub -K) and interactive jobs (bsub -I), response to batch queries can become very slow. If you run large number of bsub -I or bsub -K jobs, you can define the threads to the number of processors on the master host.

Default

Not defined

LSB_PSET_BIND_DEFAULT

Syntax

LSB_PSET_BIND_DEFAULT=y | Y

Description

If set, Platform LSF HPC binds a job that is not explicitly associated with an HP-UX pset to the default pset 0. If LSB_PSET_BIND_DEFAULT is not set, LSF HPC must still attach the job to a pset, and so binds the job to the same pset used by the LSF HPC daemons.

Use LSB_PSET_BIND_DEFAULT to improve LSF daemon performance by automatically unbinding a job with no pset options from the pset used by the LSF daemons, and binding it to the default pset.

Default

Not defined

LSB_QUERY_PORT

Syntax

LSB_QUERY_PORT=*port_number*

Description

Optional. Applies only to UNIX platforms that support thread programming.

This parameter is recommended for busy clusters with many jobs and frequent query requests to increase mbatchd performance when you use the bjobs command.

This may indirectly increase overall mbatchd performance.

The port_number is the TCP/IP port number to be used by mbatchd to only service query requests from the LSF system. mbatchd checks the query port during initialization.

If LSB_QUERY_PORT *is not* defined:

- mbatchd uses the port specified by LSB_MBD_PORT in lsf.conf, or, if LSB_MBD_PORT is not defined, looks into the system services database for port numbers to communicate with other hosts in the cluster.
- For each query request it receives, mbatchd forks one child mbatchd to service the request. Each child mbatchd processes one request and then exits.

If LSB_QUERY_PORT is defined:

- mbatchd prepares this port for connection.The default behavior of mbatchd changes, a child mbatchd is forked, and the child mbatchd creates threads to process requests.
- mbatchd responds to requests by forking one child mbatchd. As soon as mbatchd has forked a child mbatchd, the child mbatchd takes over and listens on the port to process more query requests. For each request, the child mbatchd creates a thread to process it.

The interval used by mbatchd for forking new child mbatchds is specified by the parameter MBD_REFRESH_TIME in lsb.params.

The child mbatchd continues to listen to the port number specified by LSB_QUERY_PORT and creates threads to service requests until the job changes status, a new job is submitted, or the time specified in MBD_REFRESH_TIME in lsb.params has passed (see MBD_REFRESH_TIME in lsb.params for more details). When any of these happens, the parent mbatchd sends a message to the child mbatchd to exit.

LSB_QUERY_PORT must be defined when NEWJOB_REFRESH=Y in lsb.params to enable a child mbatchd to get up to date information about new jobs from the parent mbatchd.

Operating system support

> **Tip:**
>
> See the Online Support area of the Platform Computing Web site at www.platform.com for the latest information about operating systems that support multithreaded mbatchd.

Default

Not defined

See also

MBD_REFRESH_TIME and NEWJOB_REFRESH in lsb.params

LSB_REQUEUE_TO_BOTTOM

Syntax

LSB_REQUEUE_TO_BOTTOM=0 | **1**

Description

When defined with a value of 1, requeues automatically requeued jobs to the bottom of the queue instead of to the top. Also requeues migrating jobs to the bottom of the queue if LSB_MIG2PEND is also defined with a value of 1.

Ignored in a MultiCluster environment.

Default

Not defined. LSF requeues jobs in order of original submission time and job priority.

See also

LSB_MIG2PEND, REQUEUE_EXIT_VALUES in lsb.queues

LSB_RLA_HOST_LIST

Syntax

LSB_RLA_HOST_LIST="*host_name* ...**"**

Description

By default, the LSF scheduler can contact the LSF HPC topology adapter (RLA) running on any host for Linux/QsNet RMS allocation requests. LSB_RLA_HOST_LIST defines a list of hosts to restrict which RLAs the LSF scheduler contacts.

If LSB_RLA_HOST_LIST is configured, you must list at least one host per RMS partition for the RMS partition to be considered for job scheduling.

Listed hosts must be defined in `lsf.cluster.`*cluster_name*.

Host names are separated by spaces.

Default

Not defined

LSB_RLA_PORT

Syntax

LSB_RLA_PORT=*port_number*

Description

TCP port used for communication between the LSF HPC topology adapter (RLA) and the LSF HPC scheduler plugin.

Default

6883

LSB_RLA_UPDATE

Syntax

LSB_RLA_UPDATE=*time_seconds*

Description

Specifies how often the LSF HPC scheduler refreshes free node information from the LSF HPC topology adapter (RLA).

Default

600 seconds

LSB_RLA_WORKDIR

Syntax

LSB_RLA_WORKDIR=*directory*

Description

Directory to store the LSF HPC topology adapter (RLA) status file. Allows RLA to recover its original state when it restarts. When RLA first starts, it creates the directory defined by LSB_RLA_WORKDIR if it does not exist, then creates subdirectories for each host.

You should avoid using /tmp or any other directory that is automatically cleaned up by the system. Unless your installation has restrictions on the LSB_SHAREDIR directory, you should use the default for LSB_RLA_WORKDIR.

Default

LSB_SHAREDIR/*cluster_name*/rla_workdir

LSB_RMSACCT_DELAY

Syntax

LSB_RMSACCT_DELAY=*time_seconds*

Description

If set, RES waits the specified number of seconds before exiting to allow LSF and RMS job statistics to synchronize.

If LSB_RMSACCT_DELAY=0, RES waits forever until the database is up to date.

Default

Not defined. RES does not wait at all.

LSB_RMS_MAXNUMNODES

Syntax

LSB_RMS_MAXNUMNODES=*integer*

Description

Maximum number of nodes in a system. Specifies a maximum value for the nodes argument to the topology scheduler options specified in:

- -extsched option of bsub
- DEFAULT_EXTSCHED and MANDATORY_EXTSCHED in lsb.queues

Default

1024

LSB_RMS_MAXNUMRAILS

Syntax

LSB_RMS_MAXNUMRAILS=_integer_

Description

Maximum number of rails in a system. Specifies a maximum value for the rails argument to the topology scheduler options specified in:

- -extsched option of bsub
- DEFAULT_EXTSCHED and MANDATORY_EXTSCHED in lsb.queues

Default

32

LSB_RMS_MAXPTILE

Syntax

LSB_RMS_MAXPTILE=_integer_

Description

Maximum number of CPUs per node in a system. Specifies a maximum value for the ptile argument to the topology scheduler options specified in:

- -extsched option of bsub
- DEFAULT_EXTSCHED and MANDATORY_EXTSCHED in lsb.queues

Default

32

LSB_SLURM_BESTFIT

Syntax

LSB_SLURM_BESTFIT=y | Y

Description

Enables best-fit node allocation for SLURM jobs.

By default, LSF applies a *first-fit* allocation policy to select from the nodes available for the job. The allocations are made left to right for all parallel jobs, and right to left for all serial jobs (all other job requirements being equal).

In a heterogeneous SLURM cluster, a *best-fit* allocation may be preferable for clusters where a mix of serial and parallel jobs run. In this context, best fit means: "the nodes that minimally satisfy the requirements." Nodes with the maximum number of CPUs are chosen first. For parallel and serial jobs, the nodes with minimal memory, minimal tmp space, and minimal weight are chosen.

Default

Not defined

LSB_SBD_PORT

See LSF_LIM_PORT, LSF_RES_PORT, LSB_MBD_PORT, LSB_SBD_PORT.

LSB_SET_TMPDIR

Syntax

LSB_SET_TMPDIR=y | **n**

If y, LSF sets the TMPDIR environment variable, overwriting the current value with `/tmp/`*job_ID*`.tmpdir`.

Default

n

LSB_SHAREDIR

Syntax

LSB_SHAREDIR=_directory_

Description

Directory in which the job history and accounting logs are kept for each cluster. These files are necessary for correct operation of the system. Like the organization under LSB_CONFDIR, there is one subdirectory for each cluster.

The LSB_SHAREDIR directory must be owned by the LSF administrator. It must be accessible from all hosts that can potentially become the master host, and must allow read and write access from the master host.

The LSB_SHAREDIR directory typically resides on a reliable file server.

Default

LSF_INDEP/work

See also

LSB_LOCALDIR

LSB_SHORT_HOSTLIST

Syntax

LSB_SHORT_HOSTLIST=1

Description

Displays an abbreviated list of hosts in bj obs and bhi st for a parallel job where multiple processes of a job are running on a host. Multiple processes are displayed in the following format:

`processes*hostA`

For example, if a parallel job is running 5 processes on hostA, the information is displayed in the following manner:

`5*hostA`

Setting this parameter may improve mbatchd restart performance and accelerate event replay.

Default

Not defined

LSB_SIGSTOP

Syntax

LSB_SIGSTOP=_signal_name_ | _signal_value_

Description

Specifies the signal sent by the SUSPEND action in LSF. You can specify a signal name or a number.

If LSB_SIGSTOP is set to anything other than SIGSTOP, the SIGTSTP signal that is normally sent by the SUSPEND action is not sent.

If this parameter is not defined, by default the SUSPEND action in LSF sends the following signals to a job:

- Parallel or interactive jobs: 1. SIGTSTP is sent first to allow user programs to catch the signal and clean up. 2. SIGSTOP is sent 10 seconds after SIGTSTP. SIGSTOP cannot be caught by user programs.
- Other jobs: SIGSTOP is sent. SIGSTOP cannot be caught by user programs. The same set of signals is not supported on all UNIX systems. To display a list of the symbolic names of the signals (without the SIG prefix) supported on your system, use the kill -l command.

Example

`LSB_SIGSTOP=SIGKILL`

In this example, the SUSPEND action sends the three default signals sent by the TERMINATE action (SIGINT, SIGTERM, and SIGKILL) 10 seconds apart.

Default

Not defined. Default SUSPEND action in LSF is sent.

LSB_STDOUT_DIRECT

Syntax

LSB_STDOUT_DIRECT=y | Y

Description

When set, and used with the -o or -e options of bsub, redirects standard output or standard error from the job directly to a file as the job runs.

If LSB_STDOUT_DIRECT is not set and you use the bsub - o option, the standard output of a job is written to a temporary file and copied to the file you specify *after* the job finishes.

LSB_STDOUT_DIRECT is not supported on Windows.

Default

Not defined

LSB_STOP_IGNORE_IT

Usage

LSB_STOP_IGNORE_IT= Y | y

Description

Allows a solitary job to be stopped regardless of the idle time (IT) of the host that the job is running on. By default, if only one job is running on a host, the host idle time must be zero in order to stop the job.

Default

Not defined

LSB_SUB_COMMANDNAME

Syntax

LSB_SUB_COMMANDNAME=y | Y

Description

If set, enables esub to use the variable LSB_SUB_COMMAND_LINE in the esub job parameter file specified by the $LSB_SUB_PARM_FILE environment variable.

The LSB_SUB_COMMAND_LINE variable carries the value of the bsub command argument, and is used when esub runs.

Example

esub contains:

```
#!/bin/sh . $LSB_SUB_PARM_FILE exec 1>&2 if
[ $LSB_SUB_COMMAND_LINE="netscape" ]; then echo "netscape is not allowed to
run in batch mode" exit $LSB_SUB_ABORT_VALUE fi
```

LSB_SUB_COMMAND_LINE is defined in $LSB_SUB_PARM_FILE as:

`LSB_SUB_COMMAND_LINE=netscape`

A job submitted with:

`bsub netscape ...`

Causes esub to echo the message:

`netscape is not allowed to run in batch mode`

Default

Not defined

See also

LSB_SUB_COMMAND_LINE and LSB_SUB_PARM_FILE environment variables

LSB_TIME_CMD

Syntax

LSB_TIME_CMD=*timimg_level*

Description

The timing level for checking how long batch commands run.

Time usage is logged in milliseconds; specify a positive integer.

Example: `LSB_TIME_CMD=1`

Default

Not defined

See also

LSB_TIME_MBD, LSB_TIME_SBD, LSF_TIME_LIM, LSF_TIME_RES

LSB_TIME_MBD

Syntax

LSB_TIME_MBD=*timing_level*

Description

The timing level for checking how long mbatchd routines run.

Time usage is logged in milliseconds; specify a positive integer.

Example: LSB_TIME_MBD=1

Default

Not defined

See also

LSB_TIME_CMD, LSB_TIME_SBD, LSF_TIME_LIM, LSF_TIME_RES

LSB_TIME_RESERVE_NUMJOBS

Syntax

LSB_TIME_RESERVE_NUMJOBS=_maximum_reservation_jobs_

Description

Enables time-based slot reservation. The value must be positive integer.

LSB_TIME_RESERVE_NUMJOBS controls maximum number of jobs using time-based slot reservation. For example, if LSB_TIME_RESERVE_NUMJOBS=4, only the top 4 jobs get their future allocation information.

Use LSB_TIME_RESERVE_NUMJOBS=1 to allow only the highest priority job to get accurate start time prediction.

Recommended value

3 or 4 is the recommended setting. Larger values are not as useful because after the first pending job starts, the estimated start time of remaining jobs may be changed.

Default

Not defined

LSB_TIME_SBD

Syntax

LSB_TIME_SBD=_timing_level_

Description

The timing level for checking how long sbatchd routines run.

Time usage is logged in milliseconds; specify a positive integer.

Example: LSB_TIME_SBD=1

Default

Not defined

See also

LSB_TIME_CMD, LSB_TIME_MBD, LSF_TIME_LIM, LSF_TIME_RES

LSB_TIME_SCH

Syntax

LSB_TIME_SCH=_timing_level_

Description

The timing level for checking how long mbschd routines run.

Time usage is logged in milliseconds; specify a positive integer.

Example: LSB_TIME_SCH=1

Default

Not defined

LSB_UTMP

Syntax

LSB_UTMP=y | Y

Description

If set, enables registration of user and account information for interactive batch jobs submitted with bsub -Ip or bsub -Is. To disable utmp file registration, set LSB_UTMP to any value other than y or Y; for example, LSB_UTMP=N.

LSF registers interactive batch jobs the job by adding a entries to the utmp file on the execution host when the job starts. After the job finishes, LSF removes the entries for the job from the utmp file.

Limitations

Registration of utmp file entries is supported on the following platforms:

- SGI IRIX (6.4 and later)
- Solaris (all versions)
- HP-UX (all versions)
- Linux (all versions)

utmp file registration is not supported in a MultiCluster environment.

Because interactive batch jobs submitted with bsub -I are not associated with a pseudo-terminal, utmp file registration is not supported for these jobs.

Default

Not defined

LSF_AFS_CELLNAME

Syntax

LSF_AFS_CELLNAME=_AFS_cell_name_

Description

Must be defined to AFS cell name if the AFS file system is in use.

Example:
```
LSF_AFS_CELLNAME=cern.ch
```

Default

Not defined

LSF_AM_OPTIONS

Syntax

LSF_AM_OPTIONS=AMFIRST | AMNEVER

Description

Determines the order of file path resolution when setting the user's home directory.

This variable is rarely used but sometimes LSF does not properly change the directory to the user's home directory when the user's home directory is automounted. Setting LSF_AM_OPTIONS forces LSF to change directory to $HOME before attempting to automount the user's home.

When this parameter is not defined or set to AMFIRST, LSF, sets the user's $HOME directory from the automount path. If it cannot do so, LSF sets the user's $HOME directory from the passwd file.

When this parameter is set to AMNEVER, LSF, never uses automount to set the path to the user's home. LSF sets the user's $HOME directory directly from the passwd file.

Valid values

The two values are AMFIRST and AMNEVER

Default

Same as AMFIRST

LSF_API_CONNTIMEOUT

Syntax

LSF_API_CONNTIMEOUT=*time_seconds*

Description

Timeout when connecting to LIM.

EGO parameter

EGO_LIM_CONNTIMEOUT

Default

5

See also

LSF_API_RECVTIMEOUT

LSF_API_RECVTIMEOUT

Syntax

LSF_API_RECVTIMEOUT=_time_seconds_

Description

Timeout when receiving a reply from LIM.

EGO parameter

EGO_LIM_RECVTIMEOUT

Default

20

See also

LSF_API_CONNTIMEOUT

LSF_AUTH

Syntax

LSF_AUTH=eauth | **ident**

Description

Enables either external authentication or authentication by means of identification daemons. This parameter is required for any cluster that contains Windows hosts, and is optional for UNIX-only clusters. After defining or changing the value of LSF_AUTH, you must shut down and restart the LSF daemons on all server hosts to apply the new authentication method.

eauth

For site-specific customized external authentication. Provides the highest level of security of all LSF authentication methods.

ident

For authentication using the RFC 931/1413/1414 protocol to verify the identity of the remote client. If you want to use ident authentication, you must download and install the ident protocol, available from the public domain, and register ident as required by your operating system.

For UNIX-only clusters, privileged ports authentication (setuid) can be configured by commenting out or deleting the LSF_AUTH parameter. If you choose privileged ports authentication, LSF commands must be installed as setuid programs owned by root. If the commands are installed in an NFS-mounted shared file system, the file system must be mounted with setuid execution allowed, that is, without the nosuid option.

Restriction:

To enable privileged ports authentication, LSF_AUTH must not be defined; setuid is not a valid value for LSF_AUTH.

Default

eauth

During LSF installation, a default eauth executable is installed in the directory specified by the parameter LSF_SERVERDIR in the lsf.conf file. The default executable provides an example of how the eauth protocol works. You should write your own eauth executable to meet the security requirements of your cluster.

LSF_ASPLUGIN

Syntax

LSF_ASPLUGIN=*path*

Description

Points to the SGI Array Services library libarray.so. The parameter only takes effect on 64-bit x-86 Linux 2.6, glibc 2.3.

Default

/usr/lib64/libarray.so

LSF_AUTH_DAEMONS

Syntax

LSF_AUTH_DAEMONS=y | **Y**

Description

Enables LSF daemon authentication when external authentication is enabled (LSF_AUTH=eauth in the file lsf.conf). Daemons invoke eauth to authenticate each other as specified by the eauth executable.

Default

Not defined.

LSF_BINDIR

Syntax

LSF_BINDIR=*directory*

Description

Directory in which all LSF user commands are installed.

Default

LSF_MACHDEP/bin

LSF_BIND_JOB

Syntax

LSF_BIND_JOB=NONE | **BALANCE** | **PACK** | **ANY** | **USER** | **USER_CPU_LIST**

Description

Specifies the processor binding policy for sequential and parallel job processes that run on a single host.

On Linux execution hosts that support this feature, job processes are hard bound to selected processors.

If processor binding feature is not configured with the BIND_JOB parameter in an application profile in lsb.applications, the lsf.conf configuration setting takes effect. The application profile configuration for processor binding overrides the lsf.conf configuration.

For backwards compatibility:

- LSF_BIND_JOB=Y is interpreted as LSF_BIND_JOB=BALANCE
- LSF_BIND_JOB=N is interpreted as LSF_BIND_JOB=NONE

Supported platforms

Linux with kernel version 2.6 or higher

Default

Not defined. Processor binding is disabled.

LSF_BMPLUGIN

Syntax

LSF_BMPLUGIN=_path_

Description

Points to the bitmask library libbitmask.so. The parameter only takes effect on 64-bit x-86 Linux 2.6, glibc 2.3.

Default

/usr/lib64/libbitmask.so

LSF_CMD_LOGDIR

Syntax

LSF_CMD_LOGDIR=_path_

Description

The path to the log files used for debugging LSF commands.

This parameter can also be set from the command line.

Default

/tmp

See also

LSB_CMD_LOG_MASK, LSB_CMD_LOGDIR, LSB_DEBUG, LSB_DEBUG_CMD, LSB_TIME_CMD, LSF_CMD_LOG_MASK, LSF_LOG_MASK, LSF_LOGDIR, LSF_TIME_CMD

LSF_CMD_LOG_MASK

Syntax

LSF_CMD_LOG_MASK=*log_level*

Description

Specifies the logging level of error messages from LSF commands.

For example:
```
LSF_CMD_LOG_MASK=LOG_DEBUG
```

To specify the logging level of error messages, use LSB_CMD_LOG_MASK. To specify the logging level of error messages for LSF daemons, use LSF_LOG_MASK.

LSF commands log error messages in different levels so that you can choose to log all messages, or only log messages that are deemed critical. The level specified by LSF_CMD_LOG_MASK determines which messages are recorded and which are discarded. All messages logged at the specified level or higher are recorded, while lower level messages are discarded.

For debugging purposes, the level LOG_DEBUG contains the fewest number of debugging messages and is used for basic debugging. The level LOG_DEBUG3 records all debugging messages, and can cause log files to grow very large; it is not often used. Most debugging is done at the level LOG_DEBUG2.

The commands log to the sysl og facility unless LSF_CMD_LOGDIR is set.

Valid values

The log levels from highest to lowest are:

- LOG_EMERG
- LOG_ALERT
- LOG_CRIT
- LOG_ERR
- LOG_WARNING
- LOG_NOTICE
- LOG_INFO
- LOG_DEBUG
- LOG_DEBUG1
- LOG_DEBUG2
- LOG_DEBUG3

Default

LOG_WARNING

See also

LSB_CMD_LOG_MASK, LSB_CMD_LOGDIR, LSB_DEBUG, LSB_DEBUG_CMD, LSB_TIME_CMD, LSB_CMD_LOGDIR, LSF_LOG_MASK, LSF_LOGDIR, LSF_TIME_CMD

LSF_CONF_RETRY_INT

Syntax

LSF_CONF_RETRY_INT=*time_seconds*

Description

The number of seconds to wait between unsuccessful attempts at opening a configuration file (only valid for LIM). This allows LIM to tolerate temporary access failures.

EGO parameter

EGO_CONF_RETRY_INT

Default

30

See also

LSF_CONF_RETRY_MAX

LSF_CONF_RETRY_MAX

Syntax

LSF_CONF_RETRY_MAX=*integer*

Description

The maximum number of retry attempts by LIM to open a configuration file. This allows LIM to tolerate temporary access failures. For example, to allow one more attempt after the first attempt has failed, specify a value of 1.

EGO parameter

EGO_CONF_RETRY_MAX

Default

0

See also

LSF_CONF_RETRY_INT

LSF_CONFDIR

Syntax

LSF_CONFDIR=*directory*

Description

Directory in which all LSF configuration files are installed. These files are shared throughout the system and should be readable from any host. This directory can contain configuration files for more than one cluster.

The files in the LSF_CONFDIR directory must be owned by the primary LSF administrator, and readable by all LSF server hosts.

Default

LSF_INDEP/conf

See also

LSB_CONFDIR

LSF_CPUSETLIB

Syntax

LSF_CPUSETLIB=*path*

Description

Points to the SGI cpuset library libcpuset.so. The parameter only takes effect on 64-bit x-86 Linux 2.6, glibc 2.3.

Default

/usr/lib64/libcpuset.so

LSF_CRASH_LOG

Syntax

LSF_CRASH_LOG=Y | N

Description

On Linux hosts only, enables logging when or if a daemon crashes. Relies on the Linux debugger (gdb). Two log files are created, one for the root daemons (res, lim, sbd, and mbatchd) in /tmp/lsf_root_daemons_crash.log and one for administrative daemons (mbschd) in /tmp/lsf_admin_daemons_crash.log.

File permissions for both files are 600.

If enabling, you must restart the daemons for the change to take effect.

Default

N (no log files are created for daemon crashes)

LSF_DAEMONS_CPUS

Syntax

LSF_DAEMONS_CPUS="*mbatchd_cpu_list***:***mbschd_cpu_list***"**

mbatchd_cpu_list

Defines the list of master host CPUS where the mbatchd daemon processes can run (hard CPU affinity). Format the list as a white-space delimited list of CPU numbers.

mbschd_cpu_list

Defines the list of master host CPUS where the mbschd daemon processes can run. Format the list as a white-space delimited list of CPU numbers.

Description

By default, mbatchd and mbschd can run on any CPUs. If LSF_DAEMONS_CPUS is set, they only run on a specified list of CPUs. An empty list means LSF daemons can run on any CPUs. Use spaces to separate multiple CPUs.

The operating system can assign other processes to run on the same CPU; however, if utilization of the bound CPU is lower than utilization of the unbound CPUs.

Related parameters

To improve scheduling and dispatch performance of all LSF daemons, you should use LSF_DAEMONS_CPUS together with EGO_DAEMONS_CPUS (in ego. conf or lsf. conf), which controls LIM CPU allocation, and MBD_QUERY_CPUS, which binds mbactchd query processes to specific CPUs so that higher priority daemon processes can run more efficiently. To get best performance, CPU allocation for all four daemons should be assigned their own CPUs. For example, on a 4 CPU SMP host, the following configuration gives the best performance:

```
EGO_DAEMONS_CPUS=0 LSF_DAEMONS_CPUS=1: 2 MBD_QUERY_CPUS=3
```

Examples

If you specify
LSF_DAEMONS_CPUS="1:2"

the mbatchd processes run only on CPU number 1 on the master host, and mbschd run on only on CPU number 2.

If you specify
LSF_DAEMONS_CPUS="1 2:1 2"

both mbatchd and mbschd run CPU 1 and CPU 2.

Important

You can specify CPU affinity only for master hosts that use one of the following operating systems:

- Linux 2.6 or higher
- Solaris 8 or higher

EGO parameter

LSF_DAEMONS_CPUS=*lim_cpu_list*: run the EGO LIM daemon on the specified CPUs.

Default

Not defined

See also

MBD_QUERY_CPUS in lsb.params

LSF_DAEMON_WRAP

Syntax

LSF_DAEMON_WRAP=y | Y

Description

Applies to Kerberos, DCE/DFS and AFS environments; if you are using LSF with DCE, AFS, or Kerberos, set this parameter to y or Y.

When this parameter is set to y or Y, mbatchd, sbatchd, and RES run the executable daemons.wrap located in LSF_SERVERDIR.

Default

Not defined. LSF does not run the daemons.wrap executable.

LSF_DEBUG_CMD

Syntax

LSB_DEBUG_CMD=*log_class*

Description

Sets the debugging log class for LSF commands and APIs.

Specifies the log class filtering to be applied to LSF commands or the API. Only messages belonging to the specified log class are recorded.

LSF_DEBUG_CMD sets the log class and is used in combination with LSF_CMD_LOG_MASK, which sets the log level. For example:

```
LSF_CMD_LOG_MASK=LOG_DEBUG LSF_DEBUG_CMD="LC_TRACE LC_EXEC"
```

Debugging is turned on when you define both parameters.

The daemons log to the syslog facility unless LSF_CMD_LOGDIR is defined.

To specify multiple log classes, use a space-separated list enclosed by quotation marks. For example:

```
LSF_DEBUG_CMD="LC_TRACE LC_EXEC"
```

Can also be defined from the command line.

Valid values

Valid log classes are:

- LC_AFS - Log AFS messages
- LC_AUTH - Log authentication messages
- LC_CHKPNT - Log checkpointing messages
- LC_COMM - Log communication messages
- LC_DCE - Log messages pertaining to DCE support
- LC_EEVENTD - Log eeventd messages
- LC_ELIM - Log ELIM messages
- LC_EXEC - Log significant steps for job execution
- LC_FAIR - Log fairshare policy messages
- LC_FILE - Log file transfer messages
- LC_HANG - Mark where a program might hang
- LC_JARRAY - Log job array messages
- LC_JLIMIT - Log job slot limit messages
- LC_LICENSE - Log license management messages (LC_LICENCE is also supported for backward compatibility)
- LC_LOADINDX - Log load index messages
- LC_M_LOG - Log multievent logging messages
- LC_MPI - Log MPI messages
- LC_MULTI - Log messages pertaining to MultiCluster
- LC_PEND - Log messages related to job pending reasons
- LC_PERFM - Log performance messages
- LC_PIM - Log PIM messages
- LC_PREEMPT - Log preemption policy messages
- LC_RESREQ - Log resource requirement messages
- LC_SIGNAL - Log messages pertaining to signals
- LC_SYS - Log system call messages
- LC_TRACE - Log significant program walk steps
- LC_XDR - Log everything transferred by XDR

Default

Not defined

See also

LSF_CMD_LOG_MASK, LSF_CMD_LOGDIR, LSF_DEBUG_LIM, LSF_DEBUG_RES, LSF_LIM_PORT, LSF_RES_PORT, LSB_MBD_PORT, LSB_SBD_PORT, LSF_LOGDIR, LSF_LIM_DEBUG, LSF_RES_DEBUG

LSF_DEBUG_LIM

Syntax

LSF_DEBUG_LIM=*log_class*

Description

Sets the log class for debugging LIM.

Specifies the log class filtering to be applied to LIM. Only messages belonging to the specified log class are recorded.

The LSF_DEBUG_LIM sets the log class and is used in combination with EGO_LOG_MASK in `ego, conf`, which sets the log level.

For example, in `ego. conf`:
```
EGO_LOG_MASK=LOG_DEBUG
```

and in `l sf. conf`:
```
LSF_DEBUG_LI M=LC_TRACE
```

Important:

LSF_LOG_MASK in lsf.conf no longer specifies LIM logging level in LSF Version 7. For LIM, you must use EGO_LOG_MASK in ego.conf to control message logging for LIM. The default value for EGO_LOG_MASK is LOG_WARNING.

You need to restart the daemons after setting LSF_DEBUG_LIM for your changes to take effect.

If you use the command `l sadmi n  l i mdebug` to temporarily change this parameter without changing `l sf. conf`, you do not need to restart the daemons.

To specify multiple log classes, use a space-separated list enclosed in quotation marks. For example:
```
LSF_DEBUG_LI M="LC_TRACE  LC_EXEC"
```

This parameter can also be defined from the command line.

Valid values

Valid log classes are:

- LC_AFS - Log AFS messages
- LC_AUTH - Log authentication messages
- LC_CHKPNT - log checkpointing messages
- LC_COMM - Log communication messages
- LC_DCE - Log messages pertaining to DCE support
- LC_EXEC - Log significant steps for job execution
- LC_FILE - Log file transfer messages
- LC_HANG - Mark where a program might hang
- LC_JGRP - Log job group messages
- LC_LICENSE - Log license management messages (LC_LICENCE is also supported for backward compatibility)
- LC_LICSCHED - Log LSF License Scheduler messages
- LC_MEMORY - Log memory limit messages
- LC_MULTI - Log messages pertaining to MultiCluster
- LC_PIM - Log PIM messages
- LC_RESOURCE - Log resource broker messages
- LC_SIGNAL - Log messages pertaining to signals
- LC_TRACE - Log significant program walk steps
- LC_XDR - Log everything transferred by XDR

EGO parameter

EGO_DEBUG_LIM

Default

Not defined

See also

LSF_DEBUG_RES, LSF_CMD_LOGDIR, LSF_CMD_LOG_MASK, LSF_LOG_MASK, LSF_LOGDIR

LSF_DEBUG_RES

Syntax

LSF_DEBUG_RES=*log_class*

Description

Sets the log class for debugging RES.

Specifies the log class filtering to be applied to RES. Only messages belonging to the specified log class are recorded.

LSF_DEBUG_RES sets the log class and is used in combination with LSF_LOG_MASK, which sets the log level. For example:

```
LSF_LOG_MASK=LOG_DEBUG  LSF_DEBUG_RES=LC_TRACE
```

To specify multiple log classes, use a space-separated list enclosed in quotation marks. For example:

```
LSF_DEBUG_RES="LC_TRACE  LC_EXEC"
```

You need to restart the daemons after setting LSF_DEBUG_RES for your changes to take effect.

If you use the command lsadmin resdebug to temporarily change this parameter without changing lsf.conf, you do not need to restart the daemons.

Valid values

For a list of valid log classes see LSF_DEBUG_LIM

Default

Not defined

See also

LSF_DEBUG_LIM, LSF_CMD_LOGDIR, LSF_CMD_LOG_MASK, LSF_LOG_MASK, LSF_LOGDIR

LSF_DHCP_ENV

Syntax

LSF_DHCP_ENV=y

Description

If defined, enables dynamic IP addressing for all LSF client hosts in the cluster.

Dynamic IP addressing is not supported across clusters in a MultiCluster environment.

If you set LSF_DHCP_ENV, you must also specify LSF_DYNAMIC_HOST_WAIT_TIME in order for hosts to rejoin a cluster after their IP address changes.

> **Tip:**
>
> After defining or changing this parameter, you must run `lsadmin reconfig` and `badmin mbdrestart` to restart all LSF daemons.

EGO parameter

EGO_DHCP_ENV

Default

Not defined

See also

LSF_DYNAMIC_HOST_WAIT_TIME

LSF_DISABLE_LSRUN

Syntax

LSF_DISABLE_LSRUN=y | Y

Description

When defined, RES refuses remote connections from `lsrun` and `lsgrun` unless the user is either an LSF administrator or root. For remote execution by root, LSF_ROOT_REX must be defined.

Other remote execution commands, such as `ch` and `lsmake` are not affected.

Default

Not defined

LSF_DISPATCHER_LOGDIR

Syntax

LSF_DISPATCHER_LOGDIR=_path_

Description

Specifies the path to the log files for slot allocation decisions for queue-based fairshare.

If defined, LSF writes the results of its queue-based fairshare slot calculation to the specified directory. Each line in the file consists of a timestamp for the slot allocation and the number of slots allocated to each queue under its control. LSF logs in this file every minute. The format of this file is suitable for plotting with `gnuplot`.

Example

```
# clients managed by LSF
# Roma # Verona # Genova # Pisa # Venezia # Bologna
15/3        19:4:50    0 0 0 0 0 0
15/3        19:5:51    8 5 2 5 2 0
15/3        19:6:51    8 5 2 5 5 1
15/3        19:7:53    8 5 2 5 5 5
15/3        19:8:54    8 5 2 5 5 0
15/3        19:9:55    8 5 0 5 4 2
```

The queue names are in the header line of the file. The columns correspond to the allocations per each queue.

Default

Not defined

LSF_DUALSTACK_PREFER_IPV6

Syntax

LSF_DUALSTACK_PREFER_IPV6=Y | y

Description

Define this parameter when you want to ensure that clients and servers on dual-stack hosts use IPv6 addresses only. Setting this parameter configures LSF to sort the dynamically created address lookup list in order of AF_INET6 (IPv6) elements first, followed by AF_INET (IPv4) elements, and then others.

Restriction:

IPv4-only and IPv6-only hosts cannot belong to the same cluster.
In a MultiCluster environment, you cannot mix IPv4-only and
IPv6-only clusters.

Follow these guidelines for using IPv6 addresses within your cluster:

- Define this parameter only if your cluster

  - Includes only dual-stack hosts, or a mix of dual-stack and IPv6-only hosts, *and*
  - Does not include IPv4-only hosts or IPv4 servers running on dual-stack hosts (servers prior to LSF version 7)

Important:

Do not define this parameter for any other cluster configuration.

- Within a MultiCluster environment, do not define this parameter if any cluster contains IPv4-only hosts or IPv4 servers (prior to LSF version 7) running on dual-stack hosts.
- Applications must be engineered to work with the cluster IP configuration.
- If you use IPv6 addresses within your cluster, ensure that you have configured the dual-stack hosts correctly. For more detailed information, see *Administering Platform LSF*.
- Define the parameter LSF_ENABLE_SUPPORT_IPV6 in lsf.conf.

Default

Not defined. LSF sorts the dynamically created address lookup list in order of AF_INET (IPv4) elements first, followed by AF_INET6 (IPv6) elements, and then others. Clients and servers on dual-stack hosts use the first address lookup structure in the list (IPv4).

See also

LSF_ENABLE_SUPPORT_IPV6

LSF_DYNAMIC_HOST_TIMEOUT

Syntax

LSF_DYNAMIC_HOST_TIMEOUT=*time_hours*

LSF_DYNAMIC_HOST_TIMEOUT=*time_minutes***m|M**

Description

Enables automatic removal of dynamic hosts from the cluster and specifies the timeout value (minimum 10 minutes). To improve performance in very large clusters, you should disable this feature and remove unwanted hosts from the host cache file manually.

Specifies the length of time a dynamic host is unavailable before the master host removes it from the cluster. Each time LSF removes a dynamic host, mbatchd automatically reconfigures itself.

Valid value

The timeout value must be greater than or equal to 10 minutes.

Example

```
LSF_DYNAMIC_HOST_TIMEOUT=60
```

A dynamic host is removed from the cluster when it is unavailable for 60 hours.

```
LSF_DYNAMIC_HOST_TIMEOUT=60m
```

A dynamic host is removed from the cluster when it is unavailable for 60 minutes.

EGO parameter

EGO_DYNAMIC_HOST_TIMEOUT

Default

Not defined. Unavailable hosts are never removed from the cluster.

LSF_DYNAMIC_HOST_WAIT_TIME

Syntax

LSF_DYNAMIC_HOST_WAIT_TIME=*time_seconds*

Description

Defines the length of time in seconds that a dynamic host waits communicating with the master LIM to either add the host to the cluster or to shut down any running daemons if the host is not added successfully.

> **Note:**
>
> To enable dynamically added hosts, the following parameters must be defined:
>
> - LSF_MASTER_LIST in lsf.conf
> - LSF_DYNAMIC_HOST_WAIT_TIME in lsf.conf, or EGO_DYNAMIC_HOST_WAIT_TIME in ego.conf
> - LSF_HOST_ADDR_RANGE in lsf.cluster.*cluster_name*

> **Note:**
>
> To enable daemons to be shut down automatically for hosts that attempted to join the cluster but were rejected within the LSF_DYNAMIC_HOST_WAIT_TIME period:
>
> - EGO_ENABLE_AUTO_DAEMON_SHUTDOWN in lsf.conf or in ego.conf.

Recommended value

An integer greater than zero, up to 60 seconds for every 1000 hosts in the cluster, for a maximum of 15 minutes. Selecting a smaller value results in a quicker response time for hosts at the expense of an increased load on the master LIM.

Example

```
LSF_DYNAMIC_HOST_WAIT_TIME=60
```

A host waits 60 seconds from startup to send a request for the master LIM to add it to the cluster or to shut down any daemons if it is not added to the cluster.

EGO parameter

EGO_DYNAMIC_HOST_WAIT_TIME

Default

60

LSF_EGO_DAEMON_CONTROL

Syntax

LSF_EGO_DAEMON_CONTROL="Y" | "N"

Description

Enables EGO Service Controller to control LSF res and sbatchd startup. Set the value to "Y" if you want EGO Service Controller to start res and sbatchd, and restart them if they fail.

To configure this parameter at installation, set EGO_DAEMON_CONTROL in
install.config so that res and sbatchd start automatically as EGO services.

If LSF_ENABLE_EGO="N", this parameter is ignored and EGO Service Controller is not
started.

If you manually set EGO_DAEMON_CONTROL=Y after installation, you *must* configure
LSF res and sbatchd startup to AUTOMATIC in the EGO configuration files res.xml and
sbatchd.xml under EGO_ESRVDIR/esc/conf/services.

To avoid conflicts with existing LSF startup scripts, do not set this parameter to "Y" if you use
a script (for example in /etc/rc or /etc/inittab) to start LSF daemons. If this parameter
is not defined in install.config file, it takes default value of "N".

> **Important:**
>
> After installation, LSF_EGO_DAEMON_CONTROL alone *does
> not* change the start type for the sbatchd and res EGO services
> to AUTOMATIC in res.xml and sbatchd.xml under
> EGO_ESRVDIR/esc/conf/services. You must edit these files
> and set the <sc:StartType> parameter to AUTOMATIC.

Example

LSF_EGO_DAEMON_CONTROL="N"

Default

N (res and sbatchd are started manually or through operating system rc facility)

LSF_EGO_ENVDIR

Syntax

LSF_EGO_ENVDIR=*directory*

Description

Directory where all Platform EGO configuration files are installed. These files are shared
throughout the system and should be readable from any host.

If LSF_ENABLE_EGO="N", this parameter is ignored and ego.conf is not loaded.

Default

LSF_CONFDIR/ego/*cluster_name*/kernel. If not defined, or commented out, /etc is
assumed.

LSF_ENABLE_CSA

Syntax

LSF_ENABLE_CSA=y | Y

Description

If set, enables LSF to write records for LSF jobs to SGI IRIX Comprehensive System Accounting facility (CSA).

CSA writes an accounting record for each process in the `pacct` file, which is usually located in the `/var/adm/acct/day` directory. IRIX system administrators then use the `csabuild` command to organize and present the records on a job by job basis.

When LSF_ENABLE_CSA is set, for each job run on the IRIX system, LSF writes an LSF-specific accounting record to CSA when the job starts, and when the job finishes. LSF daemon accounting in CSA starts and stops with the LSF daemon.

To disable IRIX CSA accounting, remove LSF_ENABLE_CSA from `lsf.conf`.

See the IRIX resource administration documentation for information about CSA.

Setting up IRIX CSA

1. Define the LSF_ENABLE_CSA parameter in `lsf.conf`:
   ```
   ... LSF_ENABLE_CSA=Y ...
   ```
2. Set the following parameters in `/etc/csa.conf` to on:

   * CSA_START
   * WKMG_START
3. Run the `csaswitch` command to turn on the configuration changes in `/etc/csa.conf`.

   Note:

   See the IRIX resource administration documentation for information about the `csaswitch` command.

Information written to the pacct file

LSF writes the following records to the `pacct` file when a job starts and when it exits:

* Job record type (job start or job exit)
* Current system clock time
* Service provider (LSF)
* Submission time of the job (at job start only)
* User ID of the job owner
* Array Session Handle (ASH) of the job
* IRIX job ID
* IRIX project ID
* LSF job name if it exists
* Submission host name
* LSF queue name
* LSF external job ID
* LSF job array index
* LSF job exit code (at job exit only)
* NCPUS : number of CPUs the LSF job has been using

Default

Not defined

LSF_ENABLE_DUALCORE

Syntax

LSF_ENABLE_DUALCORE=y | **n**

Description

Enables job scheduling based on dual-core information for a host. If yes (Y), LSF scheduling policies use the detected number of cores as the number of physical processors on the host instead of the number of dual-core chips for job scheduling. For a dual-core host, lshosts shows the number of cores under ncpus instead of the number of chips.

IF LSF_ENABLE_DUALCORE=n, then lshosts shows the number of processor chips under ncpus.

EGO parameter

EGO_ENABLE_DUALCORE

Default

N

LSF_ENABLE_EGO

Syntax

LSF_ENABLE_EGO="Y" | **"N"**

Description

Enables Platform EGO functionality in the LSF cluster.

If you set LSF_ENABLE_EGO="Y", you must set or uncomment LSF_EGO_ENVDIR in lsf.conf.

If you set LSF_ENABLE_EGO="N" you must remove or comment out LSF_EGO_ENVDIR in lsf.conf.

Set the value to "N" if you do not want to take advantage of the following LSF features that depend on EGO:

- LSF daemon control by EGO Service Controller
- EGO-enabled SLA scheduling
- Platform Management Console (PMC)
- LSF reporting

Important:

After changing the value of LSF_ENABLE_EGO, you must shut down and restart the cluster.

Default

Y (EGO is enabled in the LSF cluster)

LSF_ENABLE_EXTSCHEDULER

Syntax

LSF_ENABLE_EXTSCHEDULER=y | Y

Description

If set, enables mbatchd external scheduling for LSF HPC.

Default

Not defined

LSF_ENABLE_SUPPORT_IPV6

Syntax

LSF_ENABLE_SUPPORT_IPV6=y | Y

Description

If set, enables the use of IPv6 addresses in addition to IPv4.

Default

Not defined

See also

LSF_DUALSTACK_PREFER_IPV6

LSF_ENVDIR

Syntax

LSF_ENVDIR=*directory*

Description

Directory containing the lsf.conf file.

By default, lsf.conf is installed by creating a shared copy in LSF_CONFDIR and adding a symbolic link from /etc/lsf.conf to the shared copy. If LSF_ENVDIR is set, the symbolic link is installed in LSF_ENVDIR/lsf.conf.

The lsf.conf file is a global environment configuration file for all LSF services and applications. The LSF default installation places the file in LSF_CONFDIR.

Default

/etc

LSF_EVENT_PROGRAM

Syntax

LSF_EVENT_PROGRAM=*event_program_name*

Description

Specifies the name of the LSF event program to use.

If a full path name is not provided, the default location of this program is LSF_SERVERDIR.

If a program that does not exist is specified, event generation does not work.

If this parameter is not defined, the default name is `genevent` on UNIX, and `genevent.exe` on Windows.

Default

Not defined

LSF_EVENT_RECEIVER

Syntax

LSF_EVENT_RECEIVER=*event_receiver_program_name*

Description

Specifies the LSF event receiver and enables event generation.

Any string may be used as the LSF event receiver; this information is not used by LSF to enable the feature but is only passed as an argument to the event program.

If LSF_EVENT_PROGRAM specifies a program that does not exist, event generation does not work.

Default

Not defined. Event generation is disabled

LSF_GET_CONF

Syntax

LSF_GET_CONF=lim

Description

Synchronizes a local host's cluster configuration with the master host's cluster configuration. Specifies that a slave host must request cluster configuration details from the LIM of a host on the SERVER_HOST list. Use when a slave host does not share a filesystem with master hosts, and therefore cannot access cluster configuration.

Default

Not defined.

LSF_HOST_CACHE_NTTL

Syntax

LSF_HOST_CACHE_NTTL=_time_seconds_

Description

Negative-time-to-live value in seconds. Specifies the length of time the system caches a failed DNS lookup result. If you set this value to zero (0), LSF does not cache the result.

> **Note:**
>
> Setting this parameter does not affect the positive-time-to-live value set by the parameter LSF_HOST_CACHE_PTTL.

Valid values

Positive integer. Recommended value less than or equal to 60 seconds (1 minute).

Default

20 seconds

See also

LSF_HOST_CACHE_PTTL

LSF_HOST_CACHE_PTTL

Syntax

LSF_HOST_CACHE_PTTL=_time_seconds_

Description

Positive-time-to-live value in seconds. Specifies the length of time the system caches a successful DNS lookup result. If you set this value to zero (0), LSF does not cache the result.

> **Note:**
>
> Setting this parameter does not affect the negative-time-to-live value set by the parameter LSF_HOST_CACHE_NTTL.

Valid values

Positive integer. Recommended value equal to or greater than 3600 seconds (1 hour).

Default

86400 seconds (24 hours)

See also

LSF_HOST_CACHE_NTTL

LSF_HPC_EXTENSIONS

Syntax

LSF_HPC_EXTENSIONS="*extension_name* ...**"**

Description

Enables Platform LSF HPC extensions.

Valid values

The following extension names are supported:

CUMULATIVE_RUSAGE : When a parallel job script runs multiple commands, resource usage is collected for jobs in the job script, rather than being overwritten when each command is executed.

DISP_RES_USAGE_LIMITS : `bjobs` displays resource usage limits configured in the queue as well as job-level limits.

LSB_HCLOSE_BY_RES : If `res` is down, host is closed with a message

```
Host is closed because RES is not available.
```

The status of the closed host is closed_Adm. No new jobs are dispatched to this host, but currently running jobs are not suspended.

RESERVE_BY_STARTTIME : LSF selects the reservation that gives the job the earliest predicted start time.

By default, if multiple host groups are available for reservation, LSF chooses the largest possible reservation based on number of slots.

SHORT_EVENTFILE : Compresses long host name lists when event records are written to `lsb.events` and `lsb.acct` for large parallel jobs. The short host string has the format:

```
number_of_hosts*real_host_name
```

Tip:

When SHORT_EVENTFILE is enabled, older daemons and commands (pre-LSF Version 7) cannot recognize the lsb.acct and lsb.events file format.

For example, if the original host list record is

```
6 "hostA" "hostA" "hostA" "hostA" "hostB" "hostC"
```

redundant host names are removed and the short host list record becomes

```
3 "4*hostA" "hostB" "hostC"
```

When LSF_HPC_EXTENSIONS="SHORT_EVENTFILE" is set, and LSF reads the host list from `lsb.events` or `lsb.acct`, the compressed host list is expanded into a normal host list.

SHORT_EVENTFILE affects the following events and fields:

- JOB_START in `lsb.events` when a normal job is dispatched

  - numExHosts (%d)
  - execHosts (%s)
- JOB_CHUNK in `lsb.events` when a job is inserted into a job chunk

- • numExHosts (%d)
- • execHosts (%s)
- • JOB_FORWARD in `lsb.events` when a job is forwarded to a MultiCluster leased host
  - • numReserHosts (%d)
  - • reserHosts (%s)
- • JOB_FINISH record in `lsb.acct`
  - • numExHosts (%d)
  - • execHosts (%s)

SHORT_PIDLIST : Shortens the output from `bjobs` to omit all but the first process ID (PID) for a job. `bjobs` displays only the first ID and a count of the process group IDs (PGIDs) and process IDs for the job.

Without SHORT_PIDLIST, `bjobs -l` displays all the PGIDs and PIDs for the job. With SHORT_PIDLIST set, `bjobs -l` displays a count of the PGIDS and PIDs.

TASK_MEMLIMIT : Enables enforcement of a memory limit (`bsub -M`, `bmod -M`, or MEMLIMIT in `lsb.queues`) for individual tasks in a parallel job. If any parallel task exceeds the memory limit, LSF terminates the entire job.

TASK_SWAPLIMIT: Enables enforcement of a virtual memory (swap) limit (`bsub -v`, `bmod -v`, or SWAPLIMIT in `lsb.queues`) for individual tasks in a parallel job. If any parallel task exceeds the swap limit, LSF terminates the entire job.

Example JOB_START events in lsb.events:

For a job submitted with
bsub -n 64 -R "span[ptile=32]" sleep 100

Without SHORT_EVENTFILE, a JOB_START event like the following would be logged in `lsb.events`:

```
"JOB_START" "7.0" 1058989891 710 4 0 0 10.3 64 "hostA" "hostA" "hostA" "hostA" "hostA" "hostA"
"hostA" "hostA" "hostA" "hostA" "hostA" "hostA" "hostA" "hostA" "hostA" "hostA" "hostA" "hostA"
"hostA" "hostA" "hostA" "u050" "hostA" "hostA" "hostA" "hostA" "hostA" "hostA" "hostA" "hostA"
"hostA" "hostA" "hostB" "hostB" "hostB" "hostB" "hostB" "hostB" "hostB" "hostB" "hostB" "hostB"
"hostB" "hostB" "hostB" "hostB" "hostB" "hostB" "hostB" "hostB" "hostB" "hostB" "hostB" "hostB"
"hostB" "hostB" "hostB" "hostB" "hostB" "hostB" "hostB" "hostB" "hostB" "hostB" "" "" 0 "" 0
```

With SHORT_EVENTFILE, a JOB_START event would be logged in `lsb.events` with the number of execution hosts (numExHosts field) changed from 64 to 2 and the execution host list (execHosts field) shortened to "32*hostA" and "32*hostB":

```
"JOB_START" "7.0" 1058998174 812 4 0 0 10.3 2 "32*hostA" "32*hostB" "" "" 0 "" 0 ""
```

Example JOB_FINISH records in lsb.acct:

For a job submitted with
bsub -n 64 -R "span[ptile=32]" sleep 100

Without SHORT_EVENTFILE, a JOB_FINISH event like the following would be logged in `lsb.acct`:

```
"JOB_FINISH" "7.0" 1058990001 710 33054 33816578 64 1058989880 0 0 1058989891 "user1" "normal"
"span[ptile=32]" "" "" "hostA" "/scratch/user1/work" "" "" "1058989880.710" 0 64 "hostA"
"hostA" "hostA" "hostA" "hostA" "hostA" "hostA" "hostA" "hostA" "hostA" "hostA" "hostA"
"hostA" "hostA" "hostA" "hostA" "hostA" "hostA" "hostA" "hostA" "hostA" "hostA" "hostA"
"hostA" "hostA" "hostA" "hostA" "hostA" "hostA" "hostA" "hostB" "hostB" "hostB" "hostB" "hostB"
"hostB" "hostB" "hostB" "hostB" "hostB" "hostB" "hostB" "hostB" "hostB" "hostB" "hostB" "hostB"
"hostB" "hostB" "hostB" "hostB" "hostB" "hostB" "hostB" "hostB" "hostB" "hostB" "hostB" "hostB"
"hostB" "hostB" "hostB" 64 10.3 "" "sleep 100" 0.079999 0.270000 0 0 -1 0 0 0 0 0 0 -1 0 0 0 0
0 -1 "" "default" 0 64 "" "" 0 4304 6024 "" "" ""
```

With SHORT_EVENTFILE, a JOB_FINISH event like the following would be logged in
lsb.acct with the number of execution hosts (numExHosts field) changed from 64 to 2 and
the execution host list (execHosts field) shortened to "32*hostA" and "32*hostB":

```
"JOB_FINISH" "7.0" 1058998282 812 33054 33816578 64 1058998163 0 0 1058998174 "user1" "normal"
"span[ptile=32]" "" "" "hostA" "/scratch/user1/work" "" "" "" "1058998163.812" 0 2 "32*hostA"
"32*hostB" 64 10.3 "" "sleep 100" 0.039999 0.259999 0 0 -1 0 0 0 0 0 0 -1 0 0 0 0 -1 "" "default"
0 64 "" "" 0 4304 6024 "" "" "" "" 0
```

Example bjobs -l output without SHORT_PIDLIST:

bjobs -l displays all the PGIDs and PIDs for the job:

bjobs -l

```
Job <109>, User <user3>, Project <default>, Status <RUN>, Queue <normal>, Inte
                        ractive mode, Command <./myjob.sh>
Mon Jul 21 20:54:44: Submitted from host <hostA>, CWD <$HOME/LSF/jobs;

 RUNLIMIT
 10.0 min of hostA

 STACKLIMIT CORELIMIT MEMLIMIT
   5256 K    10000 K    5000 K
Mon Jul 21 20:54:51: Started on <hostA>;
Mon Jul 21 20:55:03: Resource usage collected.
                     MEM: 2 Mbytes;  SWAP: 15 Mbytes
                     PGID: 256871;  PIDs: 256871
                     PGID: 257325;  PIDs: 257325 257500 257482 257501 257523
                     257525 257531
SCHEDULING PARAMETERS:
           r15s   r1m  r15m   ut      pg     io   ls   it    tmp    swp    mem
 loadSched   -     -     -     -       -      -    -    -     -      -      -
 loadStop    -     -     -     -       -      -    -    -     -      -      -
                    cpuspeed    bandwidth
 loadSched           -           -
 loadStop            -           -
<< Job <109> is done successfully. >>
```

Example bjobs -l ouput with SHORT_PIDLIST:

bjobs -l displays a count of the PGIDS and PIDs:

bjobs -l

Job <109>, User <user3>, Project <default>, Status <RUN>, Queue <normal>, Inte

ractive mode, Command <./myjob.sh>

Mon Jul 21 20:54:44: Submitted from host <hostA>, CWD <$HOME/LSF/jobs;

RUNLIMIT

10.0 min of hostA

STACKLIMIT CORELIMIT MEMLIMIT

5256 K   10000 K   5000 K

Mon Jul 21 20:54:51: Started on <hostA>;

Mon Jul 21 20:55:03: Resource usage collected.

MEM: 2 Mbytes; SWAP: 15 Mbytes

PGID(s): 256871:1 PID, 257325:7 PIDs

SCHEDULING PARAMETERS:

	r15s	r1m	r15m	ut	pg	io	ls	it	tmp	swp	mem
loadSched	-	-	-	-	-	-	-	-	-	-	-
loadStop	-	-	-	-	-	-	-	-	-	-	-

	cpuspeed	bandwidth
loadSched	-	-
loadStop	-	-

Default

Not defined

LSF_HPC_NCPU_COND

Syntax

LSF_HPC_NCPU_COND=and | or

Description

Defines how any two LSF_HPC_NCPU_* thresholds are combined.

Default

or

LSF_HPC_NCPU_INCREMENT

Syntax

LSF_HPC_NCPU_INCREMENT=_increment_

Description

Defines the upper limit for the number of CPUs that are changed since the last checking cycle.

Default

0

LSF_HPC_NCPU_INCR_CYCLES

Syntax

LSF_HPC_NCPU_INCR_CYCLES=*increment_cycles*

Description

Minimum number of consecutive cycles where the number of CPUs changed does not exceed LSF_HPC_NCPU_INCREMENT. LSF checks total usable CPUs every 2 minutes.

Default

1

LSF_HPC_NCPU_THRESHOLD

Syntax

LSF_HPC_NCPU_THRESHOLD=*threshold*

Description

The percentage of total usable CPUs in the LSF partition of a SLURM cluster.

Default

80

LSF_HPC_PJL_LOADENV_TIMEOUT

Syntax

LSF_HPC_PJL_LOADENV_TIMEOUT=*time_seconds*

Description

Timeout value in seconds for PJL to load or unload the environment. For example, set LSF_HPC_PJL_LOADENV_TIMEOUT to the number of seconds needed for IBM POE to load or unload adapter windows.

At job startup, the PJL times out if the first task fails to register with PAM within the specified timeout value. At job shutdown, the PJL times out if it fails to exit after the last Taskstarter termination report within the specified timeout value.

Default

LSF_HPC_PJL_LOADENV_TIMEOUT=300

LSF_ID_PORT

Syntax

LSF_ID_PORT=*port_number*

Description

The network port number used to communicate with the authentication daemon when LSF_AUTH is set to ident.

Default

Not defined

LSF_INCLUDEDIR

Syntax

LSF_INCLUDEDIR=*directory*

Description

Directory under which the LSF API header files lsf.h and lsbatch.h are installed.

Default

LSF_INDEP/include

See also

LSF_INDEP

LSF_INDEP

Syntax

LSF_INDEP=*directory*

Description

Specifies the default top-level directory for all machine-independent LSF files.

This includes man pages, configuration files, working directories, and examples. For example, defining LSF_INDEP as /usr/share/lsf/mnt places man pages in /usr/share/lsf/mnt/man, configuration files in /usr/share/lsf/mnt/conf, and so on.

The files in LSF_INDEP can be shared by all machines in the cluster.

As shown in the following list, LSF_INDEP is incorporated into other LSF environment variables.

- LSB_SHAREDIR=$LSF_INDEP/work
- LSF_CONFDIR=$LSF_INDEP/conf
- LSF_INCLUDEDIR=$LSF_INDEP/include
- LSF_MANDIR=$LSF_INDEP/man
- XLSF_APPDIR=$LSF_INDEP/misc

lsf.conf

Default

/usr/share/lsf/mnt

See also

LSF_MACHDEP, LSB_SHAREDIR, LSF_CONFDIR, LSF_INCLUDEDIR, LSF_MANDIR, XLSF_APPDIR

LSF_INTERACTIVE_STDERR

Syntax

LSF_INTERACTIVE_STDERR=y | n

Description

Separates stderr from stdout for interactive tasks and interactive batch jobs.

This is useful to redirect output to a file with regular operators instead of the bsub -e *err_file* and -o *out_file* options.

This parameter can also be enabled or disabled as an environment variable.

> **Caution:**
>
> If you enable this parameter globally in lsf.conf, check any custom scripts that manipulate stderr and stdout.

When this parameter is not defined or set to n, the following are written to stdout on the submission host for interactive tasks and interactive batch jobs:

- Job standard output messages
- Job standard error messages

The following are written to stderr on the submission host for interactive tasks and interactive batch jobs:

- LSF messages
- NIOS standard messages
- NIOS debug messages (if LSF_NIOS_DEBUG=1 in lsf.conf)

When this parameter is set to y, the following are written to stdout on the submission host for interactive tasks and interactive batch jobs:

- Job standard output messages

The following are written to stderr on the submission host:

- Job standard error messages
- LSF messages
- NIOS standard messages
- NIOS debug messages (if LSF_NIOS_DEBUG=1 in lsf.conf)

Default

Not defined

502 Platform LSF Configuration Reference

Notes

When this parameter is set, the change affects interactive tasks and interactive batch jobs run with the following commands:

*   `bsub -I`
*   `bsub -Ip`
*   `bsub -Is`
*   `lsrun`
*   `lsgrun`
*   `lsmake` (Platform LSF Make)
*   `bsub pam` (Platform LSF HPC)

Limitations

*   Pseudo-terminal: Do not use this parameter if your application depends on `stderr` as a terminal. This is because LSF must use a non-pseudo-terminal connection to separate `stderr` from `stdout`.
*   Synchronization: Do not use this parameter if you depend on messages in `stderr` and `stdout` to be synchronized and jobs in your environment are continuously submitted. A continuous stream of messages causes `stderr` and `stdout` to not be synchronized. This can be emphasized with parallel jobs. This situation is similar to that of `rsh`.
*   NIOS standard and debug messages: NIOS standard messages, and debug messages (when LSF_NIOS_DEBUG=1 in `lsf.conf` or as an environment variable) are written to `stderr`. NIOS standard messages are in the format <<message>>, which makes it easier to remove them if you wish. To redirect NIOS debug messages to a file, define LSF_CMD_LOGDIR in `lsf.conf` or as an environment variable.

See also

LSF_NIOS_DEBUG, LSF_CMD_LOGDIR

LSF_LD_SECURITY

Syntax

LSF_LD_SECURITY=y | n

Description

LSF_LD_SECURITY: When set, jobs submitted using `bsub -Is` or `bsub -Ip` cause the environment variables LD_PRELOAD and LD_LIBRARY_PATH to be removed from the job environment during job initialization to ensure enhanced security against users obtaining root privileges.

Two new environment variables are created (LSF_LD_LIBRARY_PATH and LSF_LD_PRELOAD) to allow LD_PRELOAD and LD_LIBRARY_PATH to be put back before the job runs.

Default

N

LSF_LIBDIR

Syntax

LSF_LIBDIR=*directory*

Description

Specifies the directory in which the LSF libraries are installed. Library files are shared by all hosts of the same type.

Default

LSF_MACHDEP/l i b

LSF_LIC_SCHED_HOSTS

Syntax

LSF_LIC_SCHED_HOSTS="*candidate_host_list*"

candidate_host_list is a space-separated list of hosts that are candidate LSF License Scheduler hosts.

Description

The candidate License Scheduler host list is read by LIM on each host to check if the host is a candidate License Scheduler master host. If the host is on the list, LIM starts the License Scheduler daemon (bld) on the host.

LSF_LIC_SCHED_PREEMPT_REQUEUE

Syntax

LSF_LIC_SCHED_PREEMPT_REQUEUE=y | **n**

Description

Set this parameter to requeue a job whose license is preempted by LSF License Scheduler. The job is killed and requeued instead of suspended.

If you set LSF_LIC_SCHED_PREEMPT_REQUEUE, do not set LSF_LIC_SCHED_PREEMPT_SLOT_RELEASE. If both these parameters are set, LSF_LIC_SCHED_PREEMPT_SLOT_RELEASE is ignored.

Default

N

See also

LSF_LIC_SCHED_PREEMPT_SLOT_RELEASE, LSF_LIC_SCHED_PREEMPT_STOP

LSF_LIC_SCHED_PREEMPT_SLOT_RELEASE

Syntax

LSF_LIC_SCHED_PREEMPT_SLOT_RELEASE=y | **n**

Description

Set this parameter to release the slot of a job that is suspended when its license is preempted by LSF License Scheduler.

If you set LSF_LIC_SCHED_PREEMPT_SLOT_RELEASE, do not set LSF_LIC_SCHED_PREEMPT_REQUEUE. If both these parameters are set, LSF_LIC_SCHED_PREEMPT_SLOT_RELEASE is ignored.

Default

Y

See also

LSF_LIC_SCHED_PREEMPT_REQUEUE, LSF_LIC_SCHED_PREEMPT_STOP

LSF_LIC_SCHED_PREEMPT_STOP

Syntax

LSF_LIC_SCHED_PREEMPT_STOP=y | **n**

Description

Set this parameter to use job controls to stop a job that is preempted. When this parameter is set, a UNIX SIGSTOP signal is sent to suspend a job instead of a UNIX SIGTSTP.

To send a SIGSTOP signal instead of SIGTSTP, the following parameter in lsb.queues must also be set:

JOB_CONTROLS=SUSPEND[*SIGSTOP*]

Default

N

See also

LSF_LIC_SCHED_PREEMPT_SLOT_RELEASE, LSF_LIC_SCHED_PREEMPT_REQUEUE

LSF_LIC_SCHED_STRICT_PROJECT_NAME

Syntax

LSF_LIC_SCHED_STRICT_PROJECT_NAME=y | **n**

Description

Enforces strict checking of the License Scheduler project name upon job submission or job modification (bsub or bmod). If the project named is misspelled (case sensitivity applies), the job is rejected.

If this parameter is not set or it is set to n, and if there is an error in the project name, the default project is used.

Default

N

LSF_LICENSE_ACCT_PATH

Syntax

LSF_LICENSE_ACCT_PATH=*directory*

Description

Specifies the location for the license accounting files. These include the license accounting files for LSF Family products.

Use this parameter to define the location of all the license accounting files. By defining this parameter, you can store the license accounting files for the LSF Family of products in the same directory for convenience.

Default

Not defined. The license accounting files are stored in the default log directory for the particular product. For example, LSF stores its license audit file in the LSF system log file directory.

See also

- LSF_LOGDIR
- lsf.cluster_name.license.acct
- bld.license.acct

LSF_LICENSE_FILE

Syntax

LSF_LICENSE_FILE="*file_name* ... | *port_number@host_name*
[**:***port_number@host_name* ...]**"**

Description

Specifies one or more demo or FLEXlm permanent license files used by LSF.

The value for LSF_LICENSE_FILE can be either of the following:

- The full path name to the license file.

  - UNIX example:
    ```
    LSF_LICENSE_FILE=/usr/share/lsf/cluster1/conf/license.dat
    ```

- Windows examples:

  `LSF_LICENSE_FILE= C:\licenses\license.dat`

  `LSF_LICENSE_FILE=\\HostA\licenses\license.dat`

- For a permanent license, the name of the license server host and TCP port number used by the `lmgrd` daemon, in the format *port@host_name*. For example:

  `LSF_LICENSE_FILE="1700@hostD"`

- For a license with redundant servers, use a comma to separate the *port@host_names*. The port number must be the same as that specified in the SERVER line of the license file. For example:

  UNIX:

  `LSF_LICENSE_FILE="port@hostA:port@hostB:port@hostC"`

  Windows:

  `LSF_LICENSE_FILE="port@hostA;port@hostB;port@hostC"`

- For a license with distributed servers, use a pipe to separate the *port@host_names*. The port number must be the same as that specified in the SERVER line of the license file. For example:

  `LSF_LICENSE_FILE="port@hostA|port@hostB|port@hostC"`

Multiple license files should be quoted and must be separated by a pipe character (|).

Windows example:

`LSF_LICENSE_FILE="C:\licenses\license1|C:\licenses\license2|D:\mydir\license3"`

Multiple files may be kept in the same directory, but each one must reference a different license server. When checking out a license, LSF searches the servers in the order in which they are listed, so it checks the second server when there are no more licenses available from the first server.

If this parameter is not defined, LSF assumes the default location.

Default

If you installed LSF with a default installation, the license file is installed in the LSF configuration directory (`LSF_CONFDIR/license.dat`).

If you installed LSF with a custom installation, you specify the license installation directory. The default is the LSF configuration directory (`LSF_SERVERDIR` for the custom installation).

If you installed FLEXlm separately from LSF to manage other software licenses, the default FLEXlm installation puts the license file in the following location:

- UNIX: `/usr/share/flexlm/licenses/license.dat`
- Windows: `C:\flexlm\license.dat`

LSF_LICENSE_MAINTENANCE_INTERVAL

Syntax

LSF_LICENSE_MAINTENANCE_INTERVAL=*time_seconds*

Description

Specifies how often LSF checks the LSF licenses when starting or restarting the cluster. A small number could delay LSF. Valid values are from 5 to 300.

When this paramater is not set, the default value is used.

Recommended value

Set LSF_LICENSE_MAINTENANCE_INTERVAL depending on your cluster size, system buffer size, license server, and cluster communication speed:

- If you have network delays or a small system buffer (less than 32 KB), set LSF_LICENSE_MAINTENANCE to the high end of the valid values (300).
- For a small cluster (fewer than 1000 hosts), specify LSF_LICENSE_MAINTENANCE_INTERVAL with 5-60 second value.
- For a large cluster (greater than 4000 hosts) with limited licenses, use the maximum value: 300 seconds.
- If you have slow cluster communication (for example, if you use a Web-based intranet), use the maximum value: 300 seconds.

Default

5 seconds

LSF_LICENSE_NOTIFICATION_INTERVAL

Syntax

LSF_LICENSE_MAINTENANCE_INTERVAL=*time_hours*

Description

Specifies how often notification email is sent to the primary cluster administrator about overuse of LSF Family product licenses and LSF License Scheduler tokens.

Recommended value

To avoid getting the same audit information more than once, set LSF_LICENSE_NOTIFICATION_INTERVAL greater than 24 hours.

Example notification email

```
Subject: LSF license overuse LSF Administrator: Your cluster has experienced
license overuse. Platform Product License Name: LSF_MANAGER CLASS E license
usage: 0 in total; 8 in use (8 overused).  Overuse Hosts:  hostA Use lim -t
and lshosts -l or see /usr/opt/lsf7.0/log/lsf.cluster_7.0.license.acct file
for details. Please contact Platform Support at support@platform.com for
information about getting additional licenses.
```

Default

24 hours

See also

- LSF_LICENSE_ACCT_PATH
- LSF_LOGDIR
- lsf.cluster_name.license.acct
- bld.license.acct

LSF_LIM_API_NTRIES

Syntax

LSF_LIM_API_NTRIES=*integer*

Description

Defines the number of times LSF commands will retry to communicate with the LIM API when LIM is not available. LSF_LIM_API_NTRIES is ignored by LSF and EGO daemons and EGO commands. The LSF_LIM_API_NTRIES environment variable. overrides the value of LSF_LIM_API_NTRIES in `lsf.conf`.

Valid values

1 to 65535

Default

Not defined. LIM API exits without retrying.

LSF_LIM_DEBUG

Syntax

LSF_LIM_DEBUG=1 | **2**

Description

Sets LSF to debug mode.

If LSF_LIM_DEBUG is defined, LIM operates in single user mode. No security checking is performed, so LIM should not run as root.

LIM does not look in the services database for the LIM service port number. Instead, it uses port number 36000 unless LSF_LIM_PORT has been defined.

Specify 1 for this parameter unless you are testing LSF.

Valid values

LSF_LIM_DEBUG=1

LIM runs in the background with no associated control terminal.

LSF_LIM_DEBUG=2

LIM runs in the foreground and prints error messages to `tty`.

EGO parameter

EGO_LIM_DEBUG

Default

Not defined

See also

LSF_RES_DEBUG, LSF_CMD_LOGDIR, LSF_CMD_LOG_MASK, LSF_LOG_MASK, LSF_LOGDIR

LSF_LIM_IGNORE_CHECKSUM

Syntax

LSF_LIM_IGNORE_CHECKSUM=y | Y

Description

Configure LSF_LIM_IGNORE_CHECKSUM=Y to ignore warning messages logged to lim log files on non-master hosts.

When LSF_MASTER_LIST is set, lsadmin reconfig only restarts master candidate hosts (for example, after adding or removing hosts from the cluster). This can cause superfluous warning messages like the following to be logged in the lim log files for non-master hosts because lim on these hosts are not restarted after configuration change:

```
Aug 26 13:47:35 2006 9746 4 7.0 xdr_loadvector: Sender <10.225.36.46:9999>
has a different configuration
```

Default

Not defined.

See also

LSF_MASTER_LIST

LSF_LIM_PORT, LSF_RES_PORT, LSB_MBD_PORT, LSB_SBD_PORT

Syntax

LSF_LIM_PORT=*port_number*

Description

TCP service ports to use for communication with the LSF daemons.

If port parameters are not defined, LSF obtains the port numbers by looking up the LSF service names in the /etc/services file or the NIS (UNIX). If it is not possible to modify the services database, you can define these port parameters to set the port numbers.

EGO parameter

EGO_LIM_PORT

Default

On UNIX, the default is to get port numbers from the services database.

On Windows, these parameters are mandatory.

Default port number values are:

- LSF_LIM_PORT=7869
- LSF_RES_PORT=6878
- LSB_MBD_PORT=6881
- LSB_SBD_PORT=6882

LSF_LOAD_USER_PROFILE

Syntax

LSF_LOAD_USER_PROFILE=local | roaming

Description

When running jobs on Windows hosts, you can specify whether a user profile should be loaded. Use this parameter if you have jobs that need to access user-specific resources associated with a user profile.

Local and roaming user profiles are Windows features. For more information about them, check Microsoft documentation.

- Local: LSF loads the Windows user profile from the local execution machine (the host on which the job runs).

  > **Note:**
  >
  > If the user has logged onto the machine before, the profile of that user is used. If not, the profile for the default user is used

- Roaming: LSF loads a roaming user profile if it has been set up. If not, the local user profile is loaded instead.

Default

Not defined. No user profiles are loaded when jobs run on Windows hosts.

LSF_LOCAL_RESOURCES

Syntax

LSF_LOCAL_RESOURCES="*resource* ...**"**

Description

Defines instances of local resources residing on the slave host.

- For numeric resources, defined name-value pairs:
  ```
  "[resourcemap value*resource_name]"
  ```
- For Boolean resources, the value is the resource name in the form:
  ```
  "[resource resource_name]"
  ```

When the slave host calls the master host to add itself, it also reports its local resources. The local resources to be added must be defined in lsf.shared.

If the same resource is already defined in lsf.shared as default or all, it cannot be added as a local resource. The shared resource overrides the local one.

> **Tip:**

LSF_LOCAL_RESOURCES is usually set in the
slave.config file during installation. If
LSF_LOCAL_RESOURCES are already defined in a local
lsf.conf on the slave host, lsfinstall *does not* add
resources you define in LSF_LOCAL_RESOURCES in
slave.config. You should not have duplicate
LSF_LOCAL_RESOURCES entries in lsf.conf. If local
resources are defined more than once, only the last definition is
valid.

Important:

Resources must already be mapped to hosts in the ResourceMap
section of lsf.cluster.*cluster_name*. If the ResourceMap section
does not exist, local resources are not added.

Example

```
LSF_LOCAL_RESOURCES="[resourcemap 1*verilog] [resource linux]"
```

EGO parameter

EGO_LOCAL_RESOURCES

Default

Not defined

LSF_LOG_MASK

Syntax

LSF_LOG_MASK=*message_log_level*

Description

Specifies the logging level of error messages for LSF daemons, except LIM, which is controlled
by Platform EGO.

For example:

```
LSF_LOG_MASK=LOG_DEBUG
```

If EGO is enabled in the LSF cluster, and EGO_LOG_MASK is not defined, LSF uses the value
of LSF_LOG_MASK for LIM, PIM, and MELIM. EGO vemkd and pem components continue
to use the EGO default values. If EGO_LOG_MASK is defined, and EGO is enabled, then EGO
value is taken.

To specify the logging level of error messages for LSF commands, use
LSF_CMD_LOG_MASK. To specify the logging level of error messages for LSF batch
commands, use LSB_CMD_LOG_MASK.

On UNIX, this is similar to syslog. All messages logged at the specified level or higher are
recorded; lower level messages are discarded. The LSF_LOG_MASK value can be any log
priority symbol that is defined in syslog.h (see syslog).

The log levels in order from highest to lowest are:

- LOG_EMERG
- LOG_ALERT
- LOG_CRIT
- LOG_ERR
- LOG_WARNING
- LOG_NOTICE
- LOG_INFO
- LOG_DEBUG
- LOG_DEBUG1
- LOG_DEBUG2
- LOG_DEBUG3

The most important LSF log messages are at the LOG_ERR or LOG_WARNING level. Messages at the LOG_INFO and LOG_DEBUG level are only useful for debugging.

Although message log level implements similar functionality to UNIX syslog, there is no dependency on UNIX syslog. It works even if messages are being logged to files instead of syslog.

LSF logs error messages in different levels so that you can choose to log all messages, or only log messages that are deemed critical. The level specified by LSF_LOG_MASK determines which messages are recorded and which are discarded. All messages logged at the specified level or higher are recorded, while lower level messages are discarded.

For debugging purposes, the level LOG_DEBUG contains the fewest number of debugging messages and is used for basic debugging. The level LOG_DEBUG3 records all debugging messages, and can cause log files to grow very large; it is not often used. Most debugging is done at the level LOG_DEBUG2.

In versions earlier than LSF 4.0, you needed to restart the daemons after setting LSF_LOG_MASK in order for your changes to take effect.

LSF 4.0 implements dynamic debugging, which means you do not need to restart the daemons after setting a debugging environment variable.

EGO parameter

EGO_LOG_MASK

Default

LOG_WARNING

See also

LSB_CMD_LOG_MASK, LSB_CMD_LOGDIR, LSB_DEBUG, LSB_DEBUG_CMD, LSB_DEBUG_NQS, LSB_TIME_CMD, LSF_CMD_LOGDIR, LSF_CMD_LOG_MASK, LSF_DEBUG_LIM, LSB_DEBUG_MBD, LSF_DEBUG_RES, LSB_DEBUG_SBD, LSB_DEBUG_SCH, LSF_LOG_MASK, LSF_LOGDIR, LSF_TIME_CMD

LSF_LOG_MASK_WIN

Syntax

LSF_LOG_MASK_WIN=*message_log_level*

Description

Allows you to reduce the information logged to the LSF Windows event log files. Messages of lower severity than the specified level are discarded.

For all LSF files, the types of messages saved depends on LSF_LOG_MASK, so the threshold for the Windows event logs is either LSF_LOG_MASK or LSF_LOG_MASK_WIN, whichever is higher. LSF_LOG_MASK_WIN is ignored if LSF_LOG_MASK is set to a higher level.

The LSF event log files for Windows are:

- `lim.log.`*host_name*
- `res.log.`*host_name*
- `sbatchd.log.`*host_name*
- `mbatchd.log.`*host_name*
- `pim.log.`*host_name*

The log levels you can specify for this parameter, in order from highest to lowest, are:

- LOG_ERR
- LOG_WARNING
- LOG_INFO
- LOG_NONE (LSF does not log Windows events)

Default

LOG_ERR

See also

LSF_LOG_MASK

LSF_LOGDIR

Syntax

LSF_LOGDIR=*directory*

Description

Defines the LSF system log file directory. Error messages from all servers are logged into files in this directory. To effectively use debugging, set LSF_LOGDIR to a directory such as `/tmp`. This can be done in your own environment from the shell or in `lsf.conf`.

Windows

LSF_LOGDIR is required on Windows if you wish to enable logging.

You must also define LSF_LOGDIR_USE_WIN_REG=n.

If you define LSF_LOGDIR without defining LSF_LOGDIR_USE_WIN_REG=n, LSF logs error messages into files in the default local directory specified in one of the following Windows registry keys:

- On Windows 2000, Windows XP, and Windows 2003:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Platform Computing Corporation\LSF
\cluster_name\LSF_LOGDIR
```

- On Windows XP x64 and Windows 2003 x64:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Platform Computing Corporation\LSF
\cluster_name\LSF_LOGDIR
```

If a server is unable to write in the LSF system log file directory, LSF attempts to write to the following directories in the following order:

- LSF_TMPDIR if defined
- %TMP% if defined
- %TEMP% if defined
- System directory, for example, c:\winnt

UNIX

If a server is unable to write in this directory, the error logs are created in /tmp on UNIX.

If LSF_LOGDIR is not defined, syslog is used to log everything to the system log using the LOG_DAEMON facility. The syslog facility is available by default on most UNIX systems. The /etc/syslog.conf file controls the way messages are logged and the files they are logged to. See the man pages for the syslogd daemon and the syslog function for more information.

Default

Not defined. On UNIX, log messages go to syslog. On Windows, no logging is performed.

See also

LSB_CMD_LOG_MASK, LSB_CMD_LOGDIR, LSB_DEBUG, LSB_DEBUG_CMD, LSB_TIME_CMD, LSF_CMD_LOGDIR, LSF_CMD_LOG_MASK, LSF_LOG_MASK, LSF_LOGDIR_USE_WIN_REG, LSF_TIME_CMD

Files

- lim.log.*host_name*
- res.log.*host_name*
- sbatchd.log.*host_name*
- sbatchdc.log.host_name (when LSF_DAEMON_WRAP=Y)
- mbatchd.log.*host_name*
- eeventd.log.*host_name*
- pim.log.*host_name*

LSF_LOGDIR_USE_WIN_REG

Syntax

LSF_LOGDIR_USE_WIN_REG=n │ N

Description

Windows only.

If set, LSF logs error messages into files in the directory specified by LSF_LOGDIR in lsf.conf.

Use this parameter to enable LSF to save log files in a different location from the default local directory specified in the Windows registry.

If not set, or if set to any value other than N or n, LSF logs error messages into files in the default local directory specified in one of the following Windows registry keys:

- On Windows 2000, Windows XP, and Windows 2003:

  ```
  HKEY_LOCAL_MACHINE\SOFTWARE\Platform Computing Corporation\LSF
  \cluster_name\LSF_LOGDIR
  ```
- On Windows XP x64 and Windows 2003 x64:

  ```
  HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Platform Computing Corporation\LSF
  \cluster_name\LSF_LOGDIR
  ```

Default

Not set.

LSF uses the default local directory specified in the Windows registry.

See also

LSF_LOGDIR

LSF_LOGFILE_OWNER

Syntax

LSF_LOGFILE_OWNER="*user_name*"

Description

Specifies an owner for the LSF log files other than the default, the owner of `lsf.conf`. To specify a Windows user account, include the domain name in uppercase letters (*DOMAIN_NAME\user_name*).

Default

Not set. The LSF Administrator with root privileges is the owner of LSF log files.

LSF_LSLOGIN_SSH

Syntax

LSF_LSLOGIN_SSH=YES | yes

Description

Enables SSH to secure communication between hosts and during job submission.

SSH is used when running any of the following:

- Remote log on to a lightly loaded host (lslogin)
- An interactive job (bsub -IS | -ISp | ISs)
- An X-window job (bsub -IX)
- An externally submitted job that is interactive or X-window (esub)

Default

Not set. LSF uses rlogin to authenticate users.

LSF_MACHDEP

Syntax

LSF_MACHDEP=*directory*

Description

Specifies the directory in which machine-dependent files are installed. These files cannot be shared across different types of machines.

In clusters with a single host type, LSF_MACHDEP is usually the same as LSF_INDEP. The machine dependent files are the user commands, daemons, and libraries. You should not need to modify this parameter.

As shown in the following list, LSF_MACHDEP is incorporated into other LSF variables.

- LSF_BINDIR=$LSF_MACHDEP/bin
- LSF_LIBDIR=$LSF_MACHDEP/lib
- LSF_SERVERDIR=$LSF_MACHDEP/etc
- XLSF_UIDDIR=$LSF_MACHDEP/lib/uid

Default

`/usr/share/lsf`

See also

LSF_INDEP

LSF_MANDIR

Syntax

LSF_MANDIR=*directory*

Description

Directory under which all man pages are installed.

The man pages are placed in the man1, man3, man5, and man8 subdirectories of the LSF_MANDIR directory. This is created by the LSF installation process, and you should not need to modify this parameter.

Man pages are installed in a format suitable for BSD-style man commands.

For most versions of UNIX and Linux, you should add the directory LSF_MANDIR to your MANPATH environment variable. If your system has a man command that does not understand MANPATH, you should either install the man pages in the /usr/man directory or get one of the freely available man programs.

Default

LSF_INDEP/man

LSF_MASTER_LIST

Syntax

LSF_MASTER_LIST="*host_name ...***"**

Description

Required. Defines a list of hosts that are candidates to become the master host for the cluster.

Listed hosts must be defined in `lsf.cluster.`*`cluster_name`*.

Host names are separated by spaces.

> **Tip:**
>
> On UNIX and Linux, master host candidates should share LSF configuration and binaries. On Windows, configuration files are shared, but not binaries.

Starting in LSF 7, LSF_MASTER_LIST *must* be defined in `lsf.conf`.

If EGO is enabled, LSF_MASTER_LIST can only be defined `lsf.conf`. EGO_MASTER_LIST can only be defined in `ego.conf`. EGO_MASTER_LIST cannot be defined in `lsf.conf`. LSF_MASTER_LIST cannot be defined `ego.conf`.

LIM reads EGO_MASTER_LIST wherever it is defined. If both LSF_MASTER_LIST and EGO_MASTER_LIST are defined, the value of EGO_MASTER_LIST in `ego.conf` is taken. To avoid errors, you should make sure that the value of LSF_MASTER_LIST matches the value of EGO_MASTER_LIST, or define only EGO_MASTER_LIST.

If EGO is disabled, `ego.conf` not loaded and the value of LSF_MASTER_LIST defined in `lsf.conf` is taken.

When you run `lsadmin reconfig` to reconfigure the cluster, only the master LIM candidates read `lsf.shared` and `lsf.cluster.`*`cluster_name`* to get updated information. The elected master LIM sends configuration information to slave LIMs.

If you have a large number of non-master hosts, you should configure LSF_LIM_IGNORE_CHECKSUM=Y to ignore warning messages like the following logged to lim log files on non-master hosts.

```
Aug 26 13:47:35 2006 9746 4 7.0 xdr_loadvector: Sender <10.225.36.46:9999>
has a different configuration
```

Interaction with LSF_SERVER_HOSTS

You can use the same list of hosts, or a subset of the master host list defined in LSF_MASTER_LIST, in LSF_SERVER_HOSTS. If you include the primary master host in LSF_SERVER_HOSTS, you should define it as the last host of the list.

If LSF_ADD_CLIENTS is defined in `install.config` at installation, `lsfinstall` automatically appends the hosts in LSF_MASTER_LIST to the list of hosts in LSF_SERVER_HOSTS so that the primary master host is last. For example:

```
LSF_MASTER_LIST="lsfmaster hostE"
```

```
LSF_SERVER_HOSTS="hostB hostC hostD hostE lsfmaster"
```

The value of LSF_SERVER_HOSTS is not changed during upgrade.

EGO parameter

EGO_MASTER_LIST

Default

Defined at installation

See also

LSF_LIM_IGNORE_CHECKSUM

LSF_MASTER_NSLOOKUP_TIMEOUT

Syntax

LSF_MASTER_NSLOOKUP_TIMEOUT=*time_milliseconds*

Description

Timeout in milliseconds that the master LIM waits for DNS host name lookup.

If LIM spends a lot of time calling DNS to look up a host name, LIM appears to hang.

This parameter is used by master LIM only. Only the master LIM detects this parameter and enable the DNS lookup timeout.

Default

Not defined. No timeout for DNS lookup

See also

LSF_LIM_IGNORE_CHECKSUM

LSF_MAX_TRY_ADD_HOST

Syntax

LSF_MAX_TRY_ADD_HOST=*integer*

Description

When a slave LIM on a dynamically added host sends an add host request to the master LIM, but master LIM cannot add the host for some reason. the slave LIM tries again. LSF_MAX_TRY_ADD_HOST specifies how many times the slave LIM retries the add host request before giving up.

Default

20

LSF_MC_NON_PRIVILEGED_PORTS

Syntax

LSF_MC_NON_PRIVILEGED_PORTS=y | Y

Description

MultiCluster only. If this parameter is enabled in one cluster, it must be enabled in all clusters.

Specify Y to make LSF daemons use non-privileged ports for communication across clusters.

Compatibility

This disables privileged port daemon authentication, which is a security feature. If security is a concern, you should use `eauth` for LSF daemon authentication (see LSF_AUTH_DAEMONS in `lsf.conf`).

Default

Not defined. LSF daemons use privileged port authentication

LSF_MONITOR_LICENSE_TOOL

Syntax

LSF_MONITOR_LICENSE_TOOL=y | Y

Description

Specify Y to enable data collection by lim for the command option `lsadmin lsflic`.

Default

Not defined. lim ignores requests from `lsadmin`, closing the channel.

LSF_MISC

Syntax

LSF_MISC=*directory*

Description

Directory in which miscellaneous machine independent files, such as example source programs and scripts, are installed.

Default

LSF_CONFDIR/misc

LSF_NIOS_DEBUG

Syntax

LSF_NIOS_DEBUG=1

Description

Turns on NIOS debugging for interactive jobs.

If LSF_NIOS_DEBUG=1, NIOS debug messages are written to standard error.

This parameter can also be defined as an environment variable.

When LSF_NIOS_DEBUG and LSF_CMD_LOGDIR are defined, NIOS debug messages are logged in nios.log.*host_name*. in the location specified by LSF_CMD_LOGDIR.

If LSF_NIOS_DEBUG is defined, and the directory defined by LSF_CMD_LOGDIR is inaccessible, NIOS debug messages are logged to /tmp/nios.log.*host_name* instead of stderr.

On Windows, NIOS debug messages are also logged to the temporary directory.

Default

Not defined

See also

LSF_CMD_LOGDIR, LSF_CMD_LOG_MASK, LSF_LOG_MASK, LSF_LOGDIR

LSF_NIOS_JOBSTATUS_INTERVAL

Syntax

LSF_NIOS_JOBSTATUS_INTERVAL=*time_minutes*

Description

Applies only to interactive batch jobs.

Time interval at which NIOS polls mbatchd to check if a job is still running. Used to retrieve a job's exit status in the case of an abnormal exit of NIOS, due to a network failure for example.

Use this parameter if you run interactive jobs and you have scripts that depend on an exit code being returned.

When this parameter is not defined and a network connection is lost, mbatchd cannot communicate with NIOS and the return code of a job is not retrieved.

When this parameter is defined, before exiting, NIOS polls mbatchd on the interval defined by LSF_NIOS_JOBSTATUS_INTERVAL to check if a job is still running. NIOS continues to poll mbatchd until it receives an exit code or mbatchd responds that the job does not exist (if the job has already been cleaned from memory for example).

If an exit code cannot be retrieved, NIOS generates an error message and the code -11.

Valid values

Any integer greater than zero

Default

Not defined

Notes

Set this parameter to large intervals such as 15 minutes or more so that performance is not negatively affected if interactive jobs are pending for too long. NIOS always calls mbatchd on the defined interval to confirm that a job is still pending and this may add load to mbatchd.

See also

Environment variable LSF_NIOS_PEND_TIMEOUT

LSF_NIOS_MAX_TASKS

Syntax

LSF_NIOS_MAX_TASKS=*integer*

Description

Specifies the maximum number of NIOS tasks.

Default

Not defined

LSF_NIOS_RES_HEARTBEAT

Syntax

LSF_NIOS_RES_HEARTBEAT=*time_minutes*

Description

Applies only to interactive non-parallel batch jobs.

Defines how long NIOS waits before sending a message to RES to determine if the connection is still open.

Use this parameter to ensure NIOS exits when a network failure occurs instead of waiting indefinitely for notification that a job has been completed. When a network connection is lost, RES cannot communicate with NIOS and as a result, NIOS does not exit.

When this parameter is defined, if there has been no communication between RES and NIOS for the defined period of time, NIOS sends a message to RES to see if the connection is still open. If the connection is no longer available, NIOS exits.

Valid values

Any integer greater than zero

Default

Not defined

Notes

The time you set this parameter to depends how long you want to allow NIOS to wait before exiting. Typically, it can be a number of hours or days. Too low a number may add load to the system.

LSF_NON_PRIVILEGED_PORTS

Syntax

LSF_NON_PRIVILEGED_PORTS=y | Y

Description

Disables privileged ports usage.

By default, LSF daemons and clients running under root account use privileged ports to communicate with each other. Without LSF_NON_PRIVILEGED_PORTS defined, and if LSF_AUTH is not defined in l sf. conf, LSF daemons check privileged port of request message to do authentication.

If LSF_NON_PRIVILEGED_PORTS=Y is defined, LSF clients (LSF commands and daemons) do not use privileged ports to communicate with daemons and LSF daemons do not check privileged ports of incoming requests to do authentication.

LSF_PAM_APPL_CHKPNT

Syntax

LSF_PAM_APPL_CHKPNT=Y | N

Description

When set to Y, allows PAM to function together with application checkpointing support.

Default

Y

LSF_PAM_CLEAN_JOB_DELAY

Syntax

LSF_PAM_CLEAN_JOB_DELAY=*time_seconds*

Description

The number of seconds LSF waits before killing a parallel job with failed tasks. Specifying LSF_PAM_CLEAN_JOB_DELAY implies that if any parallel tasks fail, the entire job should exit without running the other tasks in the job. The job is killed if any task exits with a non-zero exit code.

Specify a value greater than or equal to zero (0).

Applies only to PAM jobs.

Default

Undefined: LSF kills the job immediately

LSF_PAM_HOSTLIST_USE

Syntax

LSF_PAM_HOSTLIST_USE=unique

Description

Used to start applications that use both OpenMP and MPI.

Valid values

unique

Default

Not defined

Notes

At job submission, LSF reserves the correct number of processors and PAM starts only 1 process per host. For example, to reserve 32 processors and run on 4 processes per host, resulting in the use of 8 hosts:

bsub -n 32 -R "span[ptile=4]" pam yourOpenMPJob

Where defined

This parameter can alternatively be set as an environment variable. For example:

setenv LSF_PAM_HOSTLIST_USE unique

LSF_PAM_PLUGINDIR

Syntax

LSF_PAM_PLUGINDIR=*path*

Description

The path to libpamvcl.so. Used with Platform LSF HPC.

Default

Path to LSF_LIBDIR

LSF_PAM_USE_ASH

Syntax

LSF_PAM_USE_ASH=y | **Y**

Description

Enables LSF to use the SGI IRIX Array Session Handles (ASH) to propagate signals to the parallel jobs.

See the IRIX system documentation and the array_session(5) man page for more information about array sessions.

Default

Not defined

LSF_PIM_INFODIR

Syntax

LSF_PIM_INFODIR=*path*

Description

The path to where PIM writes the `pim.info.host_name` file.

Specifies the path to where the process information is stored. The process information resides in the file `pim.info.host_name`. The PIM also reads this file when it starts so that it can accumulate the resource usage of dead processes for existing process groups.

EGO parameter

EGO_PIM_INFODIR

Default

Not defined. The system uses `/tmp`.

LSF_PIM_SLEEPTIME

Syntax

LSF_PIM_SLEEPTIME=*time_seconds*

Description

The reporting period for PIM.

PIM updates the process information every 15 minutes unless an application queries this information. If an application requests the information, PIM updates the process information every LSF_PIM_SLEEPTIME seconds. If the information is not queried by any application for more than 5 minutes, the PIM reverts back to the 15 minute update period.

EGO parameter

EGO_PIM_SLEEPTIME

Default

15

LSF_PIM_SLEEPTIME_UPDATE

Syntax

LSF_PIM_SLEEPTIME_UPDATE=y | n

Description

UNIX only.

Use this parameter to improve job throughput and reduce a job's start time if there are many jobs running simultaneously on a host. This parameter reduces communication traffic between sbatchd and PIM on the same host.

When this parameter is not defined or set to n, sbatchd queries PIM as needed for job process information.

When this parameter is defined, sbatchd does not query PIM immediately as it needs information; sbatchd only queries PIM every LSF_PIM_SLEEPTIME seconds.

Limitations

When this parameter is defined:

- sbatchd may be intermittently unable to retrieve process information for jobs whose run time is smaller than LSF_PIM_SLEEPTIME.
- It may take longer to view resource usage with bj obs -l.

EGO parameter

EGO_PIM_SLEEPTIME_UPDATE

Default

Not defined

LSF_POE_TIMEOUT_BIND

Syntax

LSF_POE_TIMEOUT_BIND=*time_seconds*

Description

Specifies the time in seconds for the poe_w wrapper to keep trying to set up a server socket to listen on.

poe_w is the wrapper for the IBM poe driver program.

LSF_POE_TIMEOUT_BIND can also be set as an environment variable for poe_w to read.

Default

120 seconds

LSF_POE_TIMEOUT_SELECT

Syntax

LSF_POE_TIMEOUT_SELECT=*time_seconds*

Description

Specifies the time in seconds for the poe_w wrapper to wait for connections from the pmd_w wrapper. pmd_w is the wrapper for pmd (IBM PE Partition Manager Daemon).

LSF_POE_TIMEOUT_SELECT can also be set as an environment variable for poe_w to read.

Default

160 seconds

LSF_RES_ACCT

Syntax

LSF_RES_ACCT=*time_milliseconds* | 0

Description

If this parameter is defined, RES logs information for completed and failed tasks by default (see `lsf.acct`).

The value for LSF_RES_ACCT is specified in terms of consumed CPU time (milliseconds). Only tasks that have consumed more than the specified CPU time are logged.

If this parameter is defined as LSF_RES_ACCT=0, then all tasks are logged.

For those tasks that consume the specified amount of CPU time, RES generates a record and appends the record to the task log file `lsf.acct.host_name`. This file is located in the LSF_RES_ACCTDIR directory.

If this parameter is not defined, the LSF administrator must use the `lsadmin` command (see `lsadmin`) to turn task logging on after RES has started.

Default

Not defined

See also

LSF_RES_ACCTDIR

LSF_RES_ACCTDIR

Syntax

LSF_RES_ACCTDIR=*directory*

Description

The directory in which the RES task log file `lsf.acct.`*host_name* is stored.

If LSF_RES_ACCTDIR is not defined, the log file is stored in the `/tmp` directory.

Default

(UNIX) `/tmp`

(Windows) `C:\temp`

See also

LSF_RES_ACCT

LSF_RES_ACTIVE_TIME

Syntax

LSF_RES_ACTIVE_TIME=*time_seconds*

Description

Time in seconds before LIM reports that RES is down.

Minimum value

10 seconds

Default

90 seconds

LSF_RES_CLIENT_TIMEOUT

Syntax

LSF_RES_CLIENT_TIMEOUT=*time_minutes*

Description

Specifies in minutes how long an application RES waits for a new task before exiting.

> **Caution:**
>
> If you use the LSF API to run remote tasks and you define this
> parameter with timeout. the remote execution of the new task fails
> (for example, ls_rtask()).

Default

The parameter is not set; the application RES waits indefinitely for new task to come until client tells it to quit.

LSF_RES_CONNECT_RETRY

Syntax

LSF_RES_CONNECT_RETRY=*integer* / **0**

Description

The number of attempts by RES to reconnect to NIOS.

If LSF_RES_CONNECT_RETRY is not defined, the default value is used.

Default

0

See also

LSF_NIOS_RES_HEARTBEAT

LSF_RES_DEBUG

Syntax

LSF_RES_DEBUG=1 / **2**

Description

Sets RES to debug mode.

If LSF_RES_DEBUG is defined, the Remote Execution Server (RES) operates in single user mode. No security checking is performed, so RES should not run as root. RES does not look in the services database for the RES service port number. Instead, it uses port number 36002 unless LSF_RES_PORT has been defined.

Specify 1 for this parameter unless you are testing RES.

Valid values

LSF_RES_DEBUG=1

RES runs in the background with no associated control terminal.

LSF_RES_DEBUG=2

RES runs in the foreground and prints error messages to `tty`.

Default

Not defined

See also

LSF_LIM_DEBUG, LSF_CMD_LOGDIR, LSF_CMD_LOG_MASK, LSF_LOG_MASK, LSF_LOGDIR

LSF_RES_PORT

See LSF_LIM_PORT, LSF_RES_PORT, LSB_MBD_PORT, LSB_SBD_PORT.

LSF_RES_RLIMIT_UNLIM

Syntax

LSF_RES_RLIMIT_UNLIM=cpu | **fsize** | **data** | **stack** | **core** | **vmem**

Description

By default, RES sets the hard limits for a remote task to be the same as the hard limits of the local process. This parameter specifies those hard limits which are to be set to unlimited, instead of inheriting those of the local process.

Valid values are cpu, fsize, data, stack, core, and vmem, for CPU, file size, data size, stack, core size, and virtual memory limits, respectively.

Example

The following example sets the CPU, core size, and stack hard limits to be unlimited for all remote tasks:

```
LSF_RES_RLIMIT_UNLIM="cpu core stack"
```

Default

Not defined

LSF_RES_TIMEOUT

Syntax

LSF_RES_TIMEOUT=*time_seconds*

Description

Timeout when communicating with RES.

Default

15

LSF_ROOT_REX

Syntax

LSF_ROOT_REX=local

Description

UNIX only.

Allows root remote execution privileges (subject to identification checking) on remote hosts, for both interactive and batch jobs. Causes RES to accept requests from the superuser (root) on remote hosts, subject to identification checking.

If LSF_ROOT_REX is not defined, remote execution requests from user root are refused.

Theory

Sites that have separate root accounts on different hosts within the cluster should not define LSF_ROOT_REX. Otherwise, this setting should be based on local security policies.

The lsf.conf file is host-type specific and not shared across different platforms. You must make sure that lsf.conf for all your host types are changed consistently.

Default

Not defined. Root execution is not allowed.

See also

LSF_TIME_CMD, LSF_AUTH

LSF_RSH

Syntax

LSF_RSH=*command* [*command_options*]

Description

Specifies shell commands to use when the following LSF commands require remote execution:

- badmin hstartup
- bpeek

- `lsadmin limstartup`
- `lsadmin resstartup`
- `lsfrestart`
- `lsfshutdown`
- `lsfstartup`
- `lsrcpu`

By default, `rsh` is used for these commands. Use LSF_RSH to enable support for `ssh`.

EGO parameter

EGO_RSH

Default

Not defined

Example

To use an `ssh` command before trying `rsh` for LSF commands, specify:

```
LSF_RSH="ssh -o 'PasswordAuthentication no' -o 'StrictHostKeyChecking no'"
```

`ssh` options such as `PasswordAuthentication` and `StrictHostKeyChecking` can also be configured in the global `SSH_ETC/ssh_config` file or `$HOME/.ssh/config`.

See also

`ssh`, `ssh_config`

LSF_SECUREDIR

Syntax

LSF_SECUREDIR=_path_

Description

Windows only; mandatory if using `lsf.sudoers`.

Path to the directory that contains the file `lsf.sudoers` (shared on an NTFS file system).

LSF_SERVER_HOSTS

Syntax

LSF_SERVER_HOSTS="_host_name_ ...**"**

Description

Defines one or more server hosts that the client should contact to find a Load Information Manager (LIM). 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.

Important:

LSF_SERVER_HOSTS is required for non-shared slave hosts.

Use this parameter to ensure that commands execute successfully when no LIM is running on the local host, or when the local LIM has just started. The client contacts the LIM on one of the LSF_SERVER_HOSTS and execute the command, provided that at least one of the hosts defined in the list has a LIM that is up and running.

If LSF_SERVER_HOSTS is not defined, the client tries to contact the LIM on the local host.

The host names in LSF_SERVER_HOSTS must be enclosed in quotes and separated by white space. For example:

```
LSF_SERVER_HOSTS="hostA hostD hostB"
```

The parameter string can include up to 4094 characters for UNIX or 255 characters for Windows.

Interaction with LSF_MASTER_LIST

Starting in LSF 7, LSF_MASTER_LIST must be defined in `lsf.conf`. You can use the same list of hosts, or a subset of the master host list, in LSF_SERVER_HOSTS. If you include the primary master host in LSF_SERVER_HOSTS, you should define it as the last host of the list.

If LSF_ADD_CLIENTS is defined in `install.config` at installation, `lsfinstall` automatically appends the hosts in LSF_MASTER_LIST to the list of hosts in LSF_SERVER_HOSTS so that the primary master host is last. For example:

```
LSF_MASTER_LIST="lsfmaster hostE"
```

```
LSF_SERVER_HOSTS="hostB hostC hostD hostE lsfmaster"
```

```
LSF_ADD_CLIENTS="clientHostA"
```

The value of LSF_SERVER_HOSTS is not changed during upgrade.

Default

Not defined

See also

LSF_MASTER_LIST

LSF_SERVERDIR

Syntax

LSF_SERVERDIR=*directory*

Description

Directory in which all server binaries and shell scripts are installed.

These include lim, res, nios, sbatchd, mbatchd, and mbschd. If you use `elim`, `eauth`, `eexec`, `esub`, etc, they are also installed in this directory.

Default

LSF_MACHDEP/etc

See also

LSB_ECHKPNT_METHOD_DIR

LSF_SHELL_AT_USERS

Syntax

LSF_SHELL_AT_USERS="*user_name user_name ...***"**

Description

Applies to l stcsh only. Specifies users who are allowed to use @ for host redirection. Users not specified with this parameter cannot use host redirection in l stcsh. To specify a Windows user account, include the domain name in uppercase letters (*DOMAIN_NAME\user_name*).

If this parameter is not defined, all users are allowed to use @ for host redirection in l stcsh.

Default

Not defined

LSF_SHIFT_JIS_INPUT

Syntax

LSF_SHIFT_JIS_INPUT=y | n

Description

Enables LSF to accept Shift-JIS character encoding for job information (for example, user names, queue names, job names, job group names, project names, commands and arguments, esub parameters, external messages, etc.)

Default

n

LSF_SLURM_DISABLE_CLEANUP

Syntax

LSF_SLURM_DISABLE_CLEANUP=y | Y

Description

Disables cleanup of non-LSF jobs running in a SLURM LSF partition on a SLURM cluster.

By default, only LSF jobs are allowed to run within a SLURM LSF partition. LSF periodically cleans up any jobs submitted outside of LSF. This clean up period is defined through LSB_RLA_UPDATE.

For example, the following srun job is not submitted through LSF, so it is terminated:

srun -n 4 -p lsf sleep 100000

```
srun: error: n13: task[0-1]: Terminated
```

```
srun: Terminating job
```

If LSF_SLURM_DISABLE_CLEANUP=Y is set, this job would be allowed to run.

Default

Not defined

LSF_SLURM_TMPDIR

Syntax

LSF_SLURM_TMPDIR=*path*

Description

Specifies the LSF HPC tmp directory for SLURM clusters. The default LSF_TMPDIR /tmp cannot be shared across nodes, so LSF_SLURM_TMPDIR must specify a path that is accessible on all SLURM nodes.

Default

`/hptc_cluster/lsf/tmp`

LSF_STRICT_CHECKING

Syntax

LSF_STRICT_CHECKING=Y

Description

If set, enables more strict checking of communications between LSF daemons and between LSF commands and daemons when LSF is used in an untrusted environment, such as a public network like the Internet.

If you enable this parameter, you must enable it in the entire cluster, as it affects all communications within LSF. If it is used in a MultiCluster environment, it must be enabled in all clusters, or none. Ensure that all binaries and libraries are upgraded to LSF Version 7, including LSF_BINDIR, LSF_SERVERDIR and LSF_LIBDIR directories, if you enable this parameter.

If your site uses any programs that use the LSF base and batch APIs, or LSF MPI (Message Passing Interface), they need to be recompiled using the LSF Version 7 APIs before they can work properly with this option enabled.

> **Important:**
>
> You must shut down the entire cluster before enabling or disabling this parameter.
>
> If LSF_STRICT_CHECKING is defined, and your cluster has slave hosts that are dynamically added, LSF_STRICT_CHECKING must be configured in the local lsf.conf on all slave hosts.

Valid value

Set to Y to enable this feature.

Default

Not defined. LSF is secure in trusted environments.

LSF_STRICT_RESREQ

Syntax

LSF_STRICT_RESREQ=Y | N

Description

When LSF_STRICT_RESREQ=Y, the resource requirement selection string must conform to the stricter resource requirement syntax described in *Administering Platform LSF*. The strict resource requirement syntax only applies to the select section. It does not apply to the other resource requirement sections (order, rusage, same, span, or cu).

When LSF_STRICT_RESREQ=Y in lsf.conf, LSF rejects resource requirement strings where an rusage section contains a non-consumable resource.

When LSF_STRICT_RESREQ=N, the default resource requirement selection string evaluation is performed.

Default

N

LSF_STRIP_DOMAIN

Syntax

LSF_STRIP_DOMAIN=domain_suffix[**:**domain_suffix ...]

Description

(Optional) If all of the hosts in your cluster can be reached using short host names, you can configure LSF to use the short host names by specifying the portion of the domain name to remove. If your hosts are in more than one domain or have more than one domain name, you can specify more than one domain suffix to remove, separated by a colon (:).

For example, given this definition of LSF_STRIP_DOMAIN,

```
LSF_STRIP_DOMAIN=.foo.com:.bar.com
```

LSF accepts hostA, hostA.foo.com, and hostA.bar.com as names for host hostA, and uses the name hostA in all output. The leading period '.' is required.

Example:

```
LSF_STRIP_DOMAIN=.platform.com:.generic.com
```

In the above example, LSF accepts hostA, hostA.platform.com, and hostA.generic.com as names for hostA, and uses the name hostA in all output.

Setting this parameter only affects host names displayed through LSF, it does not affect DNS host lookup.

EGO parameter

EGO_STRIP_DOMAIN

Default

Not defined

LSF_TIME_CMD

Syntax

LSF_TIME_CMD=*timimg_level*

Description

The timing level for checking how long LSF commands run. Time usage is logged in milliseconds. Specify a positive integer.

Default

Not defined

See also

LSB_TIME_MBD, LSB_TIME_SBD, LSB_TIME_CMD, LSF_TIME_LIM, LSF_TIME_RES

LSF_TIME_LIM

Syntax

LSF_TIME_LIM=*timing_level*

Description

The timing level for checking how long LIM routines run.

Time usage is logged in milliseconds. Specify a positive integer.

EGO parameter

EGO_TIME_LIM

Default

Not defined

See also

LSB_TIME_CMD, LSB_TIME_MBD, LSB_TIME_SBD, LSF_TIME_RES

LSF_TIME_RES

Syntax

LSF_TIME_RES=*timing_level*

Description

The timing level for checking how long RES routines run.

Time usage is logged in milliseconds. Specify a positive integer.

LSF_TIME_RES is not supported on Windows.

Default

Not defined

See also

LSB_TIME_CMD, LSB_TIME_MBD, LSB_TIME_SBD, LSF_TIME_LIM

LSF_TMPDIR

Syntax

LSF_TMPDIR=*directory*

Description

Specifies the path and directory for temporary job output.

When LSF_TMPDIR is defined in `lsf.conf`, LSF creates a temporary directory under the directory specified by LSF_TMPDIR on the execution host when a job is started and sets the temporary directory environment variable (TMPDIR) for the job.

The name of the temporary directory has the following format:

```
$LSF_TMPDIR/job_ID.tmpdir
```

On UNIX, the directory has the permission 0700 and is owned by the execution user.

After adding LSF_TMPDIR to `lsf.conf`, use `badmin hrestart all` to reconfigure your cluster.

If LSB_SET_TMPDIR= Y, the environment variable TMPDIR will be set equal to the path specified by LSF_TMPDIR.

If the path specified by LSF_TMPDIR does not exist, the value of TMPDIR is set to the default path `/tmp/job_ID.tmpdir`.

Valid values

Specify any valid path up to a maximum length of 256 characters. The 256 character maximum path length includes the temporary directories and files that the system creates as jobs run. The path that you specify for LSF_TMPDIR should be as short as possible to avoid exceeding this limit.

UNIX

Specify an absolute path. For example:

```
LSF_TMPDIR=/usr/share/lsf_tmp
```

Windows

Specify a UNC path or a path with a drive letter. For example:

```
LSF_TMPDIR=\\HostA\temp\lsf_tmp
```

```
LSF_TMPDIR=D:\temp\lsf_tmp
```

Temporary directory for tasks launched by blaunch

By default, LSF creates a temporary directory for a job only on the first execution host. If LSF_TMPDIR is set in lsf.conf, the path of the job temporary directory on the first execution host is set to LSF_TMPDIR/job_ID.tmpdir.

If LSB_SET_TMPDIR=Y, the environment variable TMPDIR will be set equal to the path specified by LSF_TMPDIR.

Tasks launched through the blaunch distributed application framework make use of the LSF temporary directory specified by LSF_TMPDIR:

*   When the environment variable TMPDIR is set on the first execution host, the blaunch framework propagates this environment variable to all execution hosts when launching remote tasks
*   The job RES or the task RES creates the directory specified by TMPDIR if it does not already exist before starting the job
*   The directory created by the job RES or task RES has permission 0700 and is owned by the execution user
*   If the TMPDIR directory was created by the task RES, LSF deletes the temporary directory and its contents when the task is complete
*   If the TMPDIR directory was created by the job RES, LSF will delete the temporary directory and its contents when the job is done
*   If the TMPDIR directory is on a shared file system, it is assumed to be shared by all the hosts allocated to the blaunch job, so LSF *does not* remove TMPDIR directories created by the job RES or task RES

Default

By default, LSF_TMPDIR is not enabled. If LSF_TMPDIR is not specified in lsf.conf, this parameter is defined as follows:

*   On UNIX: $TMPDIR/job_ID.tmpdir or /tmp/job_ID.tmpdir
*   On Windows: %TMP%, %TEMP, or %SystemRoot%

LSF_ULDB_DOMAIN

Syntax

LSF_ULDB_DOMAIN="domain_name ...**"**

Description

LSF_ULDB_DOMAIN specifies the name of the LSF domain in the ULDB domain directive. A domain definition of name *domain_name* must be configured in the SGI IRIX jlimit.in input file.

Used with IRIX User Limits Database (ULDB). Configures LSF so that jobs submitted to a host with the IRIX job limits option installed are subject to the job limits configured in the IRIX User Limits Database (ULDB).

The ULDB contains job limit information that system administrators use to control access to a host on a per user basis. The job limits in the ULDB override the system default values for both job limits and process limits. When a ULDB domain is configured, the limits are enforced as IRIX job limits.

If the ULDB domain specified in LSF_ULDB_DOMAIN is not valid or does not exist, LSF uses the limits defined in the domain named batch. If the batch domain does not exist, then the system default limits are set.

When an LSF job is submitted, an IRIX job is created, and the job limits in the ULDB are applied.

Next, LSF resource usage limits are enforced for the IRIX job under which the LSF job is running. LSF limits override the corresponding IRIX job limits. The ULDB limits are used for any LSF limits that are not defined. If the job reaches the IRIX job limits, the action defined in the IRIX system is used.

IRIX job limits in the ULDB apply only to batch jobs.

See the IRIX resource administration documentation for information about configuring ULDB domains in the jlimit.in file.

LSF resource usage limits controlled by ULDB

- PROCESSLIMIT: Corresponds to IRIX JLIMIT_NUMPROC; fork() fails, but the existing processes continue to run
- MEMLIMIT : Corresponds to JLIMIT_RSS; Resident pages above the limit become prime swap candidates
- DATALIMIT : Corresponds to LIMIT_DATA; malloc() calls in the job fail with errno set to ENOMEM
- CPULIMIT: Corresponds to JLIMIT_CPU; IRIX sends SIGXCPU signal to job, then after the grace period expires, sends SIGINT, SIGTERM, and SIGKILL
- FILELIMIT: No corresponding IRIX limit; use process limit RLIMIT_FSIZE
- STACKLIMIT : No corresponding IRIX limit; use process limit RLIMIT_STACK
- CORELIMIT: No corresponding IRIX limit; use process limit RLIMIT_CORE
- SWAPLIMIT: Corresponds to JLIMIT_VMEM; use process limit RLIMIT_VMEM

Increasing the default MEMLIMIT for ULDB

In some pre-defined LSF queues, such as normal, the default MEMLIMIT is set to 5000 (5 MB). However, if ULDB is enabled (LSF_ULDB_DOMAIN is defined) the MEMLIMIT should be set greater than 8000 in lsb.queues.

Default

Not defined

LSF_UNIT_FOR_LIMITS

Syntax

LSF_UNIT_FOR_LIMITS=*unit*

Description

Enables scaling of large units in resource usage limits.

When set, LSF_UNIT_FOR_LIMITS applies cluster-wide to limits at the job-level (bsub), queue-level (lsb.queues), and application level (lsb.applications).

The limit unit specified by LSF_UNIT_FOR_LIMITS also applies to limits modified with bmod, and the display of resource usage limits in query commands (bacct, bapp, bhist, bhosts, bjobs, bqueues, lsload, and lshosts).

Important:

Before changing the units of your resource usage limits, you should completely drain the cluster of all workload. There should be no running, pending, or finished jobs in the system.

In a MultiCluster environment, you should configure the same unit for all clusters.

Example

A job is submitted with bsub -M 100 and LSF_UNIT_FOR_LIMITS=MB; the memory limit for the job is 100 MB rather than the default 100 KB.

Valid values

unit indicates the unit for the resource usage limit, one of:

- KB (kilobytes)
- MB (megabytes)
- GB (gigabytes)
- TB (terabytes)
- PB (petabytes)
- EB (exabytes)

Default

KB

LSF_USE_HOSTEQUIV

Syntax

LSF_USE_HOSTEQUIV=y | Y

Description

(UNIX only; optional)

If LSF_USE_HOSTEQUIV is defined, RES and mbatchd call the ruserok() function to decide if a user is allowed to run remote jobs.

The ruserok() function checks in the /etc/hosts.equiv file and the user's $HOME/.rhosts file to decide if the user has permission to execute remote jobs.

If LSF_USE_HOSTEQUIV is not defined, all normal users in the cluster can execute remote jobs on any host.

If LSF_ROOT_REX is set, root can also execute remote jobs with the same permission test as for normal users.

Default

Not defined

See also

LSF_ROOT_REX

LSF_USER_DOMAIN

Syntax

LSF_USER_DOMAIN=*domain_name:domain_name:domain_name... .*

Description

Enables the UNIX/Windows user account mapping feature, which allows cross-platform job submission and execution in a mixed UNIX/Windows environment. LSF_USER_DOMAIN specifies one or more Windows domains that LSF either strips from the user account name when a job runs on a UNIX host, or adds to the user account name when a job runs on a Windows host.

Important:

Configure LSF_USER_DOMAIN immediately after you install LSF; changing this parameter in an existing cluster requires that you verify and possibly reconfigure service accounts, user group memberships, and user passwords.

Specify one or more Windows domains, separated by a colon (:). You can enter an unlimited number of Windows domains. A period (.) specifies a local account, not a domain.

Examples

LSF_USER_DOMAIN=BUSINESS

LSF_USER_DOMAIN=BUSINESS:ENGINEERING:SUPPORT

Default

The default depends on your LSF installation:

- If you upgrade a cluster to LSF version 7, the default is the existing value of LSF_USER_DOMAIN, if defined
- For a new cluster, this parameter is not defined, and UNIX/Windows user account mapping is not enabled

LSF_VPLUGIN

Syntax

LSF_VPLUGIN=*path*

Description

The full path to the vendor MPI library l i bxmpi . so. Used with Platform LSF HPC.

For PAM to access the SGI MPI `libxmpi.so` library, the file permission mode must be 755 (`-rwxr-xr-x`).

Examples

- HP MPI: `LSF_VPLUGIN=/opt/mpi/lib/pa1.1/libmpirm.sl`
- SGI MPI: `LSF_VPLUGIN=/usr/lib32/libxmpi.so`
- SGI Linux (64-bit x-86 Linux 2.6, glibc 2.3.): `LSF_VPLUGIN=/usr/lib32/libxmpi.so: /usr/lib/libxmpi.so: /usr/lib64/libxmpi.so`

Default

Not defined

MC_PLUGIN_REMOTE_RESOURCE

Syntax

MC_PLUGIN_REMOTE_RESOURCE=y

Description

MultiCluster job forwarding model only. By default, the submission cluster does not consider remote resources. Define MC_PLUGIN_REMOTE_RESOURCE=y in the submission cluster to allow consideration of remote resources.

> **Note:**
>
> When MC_PLUGIN_REMOTE_RESOURCE is defined, only the following resource requirements are supported: -R "type==type_name", -R "same[type]" and -R "defined (resource_name)"

Default

Not defined. The submission cluster does not consider remote resources.

XLSF_APPDIR

Syntax

XLSF_APPDIR=*directory*

Description

(UNIX only; optional) Directory in which X application default files for LSF products are installed.

The LSF commands that use X look in this directory to find the application defaults. Users do not need to set environment variables to use the Platform LSF X applications. The application default files are platform-independent.

Default

`LSF_INDEP/misc`

XLSF_UIDDIR

Syntax

XLSF_UIDDIR=*directory*

Description

(UNIX only) Directory in which Motif User Interface Definition files are stored.

These files are platform-specific.

Default

LSF_LIBDIR/uid

lsf.licensescheduler

The `lsf.licensescheduler` file contains Platform LSF License Scheduler configuration information. All sections except ProjectGroup are required.

The command `blparams` displays configuration information from this file.

Changing lsf.licensescheduler configuration

After making any changes to `lsf.licensescheduler`, run the following commands:

- `bladmin reconfig` to reconfigure `bld`
- `badmin mbdrestart` to restart mbatchd

Parameters section

Description

Required. Defines License Scheduler configuration parameters.

Parameters section structure

The Parameters section begins and ends with the lines `Begin Parameters` and `End Parameters`. Each subsequent line describes one configuration parameter. All parameters are mandatory.

```
Begin Parameters
ADMIN=lsadmin
HOSTS=hostA hostB hostC
LMSTAT_PATH=/etc/flexlm/bin
LM_STAT_INTERVAL=30
PORT=9581
End Parameters
```

Parameters

- ADMIN
- AUTH
- DISTRIBUTION_POLICY_VIOLATION_ACTION
- ENABLE_INTERACTIVE
- HOSTS
- LIB_RECVTIMEOUT
- LM_REMOVE_INTERVAL
- LM_STAT_INTERVAL
- LMSTAT_PATH
- LS_DEBUG_BLD
- LS_LOG_MASK
- LS_MAX_TASKMAN_SESSIONS
- LS_PREEMPT_PEER
- PORT

- BLC_HEARTBEAT_FACTOR

ADMIN

Syntax

ADMIN=*user_name* ...

Description

Defines the License Scheduler administrator using a valid UNIX user account. You can specify multiple accounts.

AUTH

Syntax

AUTH=Y

Description

Enables License Scheduler user authentication for projects for taskman jobs.

DISTRIBUTION_POLICY_VIOLATION_ACTION

Syntax

DISTRIBUTION_POLICY_VIOLATION_ACTION=(PERIOD *reporting_period* CMD *reporting_command***)**

reporting_period

Specify the keyword PERIOD with a positive integer representing the interval (a multiple of LM_STAT_INTERVAL periods) at which License Scheduler checks for distribution policy violations.

reporting_command

Specify the keyword CMD with the directory path and command that License Scheduler runs when reporting a violation.

Description

Optional. Defines how License Scheduler handles distribution policy violations. Distribution policy violations are caused by non-LSF workloads; LSF License Scheduler explicitly follows its distribution policies.

License Scheduler reports a distribution policy violation when the total number of licenses given to the LSF workload, both free and in use, is less than the LSF workload distribution specified in WORKLOAD_DISTRIBUTION. If License Scheduler finds a distribution policy violation, it creates or overwrites the LSF_LOGDIR/
bld.violation.*service_domain_name*.log file and runs the user command specified by the CMD keyword.

Example

The LicenseServer1 service domain has a total of 80 licenses, and its workload distribution and enforcement is configured as follows:

```
Begin Parameter

 ...

DISTRIBUTION_POLICY_VIOLATION_ACTION=(PERIOD 5 CMD /bin/mycmd)

 ...

End Parameter
Begin Feature
NAME=ApplicationX
DISTRIBUTION=LicenseServer1(Lp1 1 Lp2 2)
WORKLOAD_DISTRIBUTION=LicenseServer1(LSF 8 NON_LSF 2)
End Feature
```

According to this configuration, 80% of the available licenses, or 64 licenses, are available to the LSF workload. License Scheduler checks the service domain for a violation every five scheduling cycles, and runs the /bin/mycmd command if it finds a violation.

If the current LSF workload license usage is 50 and the number of free licenses is 10, the total number of licenses assigned to the LSF workload is 60. This is a violation of the workload distribution policy because this is less than the specified LSF workload distribution of 64 licenses.

ENABLE_INTERACTIVE

Syntax

ENABLE_INTERACTIVE=Y

Description

Optional. Globally enables one share of the licenses for interactive tasks.

> **Tip:**
>
> By default, ENABLE_INTERACTIVE is not set. License Scheduler allocates licenses equally to each cluster and does not distribute licenses for interactive tasks.

HOSTS

Syntax

HOSTS=*host_name.domain_name* ...

Description

Defines License Scheduler hosts, including License Scheduler candidate hosts.

Specify a fully qualified host name such as hostX.mycompany.com. You can omit the domain name if all your License Scheduler clients run in the same DNS domain.

LIB_RECVTIMEOUT

Syntax

LIB_RECVTIMEOUT=*seconds*

Description

Specifies a timeout value in seconds for communication between LSF License Scheduler and LSF.

Default

0 seconds

LM_REMOVE_INTERVAL

Syntax

LM_REMOVE_INTERVAL=*seconds*

Description

Specifies the minimum time a job must have a license checked out before `lmremove` can remove the license. `lmremove` causes `lmgrd` and vendor daemons to close the TCP connection with the application. They then retry the license checkout.

Default

180 seconds

LM_STAT_INTERVAL

Syntax

LM_STAT_INTERVAL=*seconds*

Description

Defines a time interval between calls that License Scheduler makes to collect license usage information from FLEXlm license management.

Default

60 seconds

LMSTAT_PATH

Syntax

LMSTAT_PATH=*path*

Description

Defines the full path to the location of the FLEXlm command `lmstat`.

LS_DEBUG_BLD

Syntax

LS_DEBUG_BLD=*log_class*

Description

Sets the debugging log class for the LSF License Schedulerbld daemon.

Specifies the log class filtering to be applied to bld. Messages belonging to the specified log class are recorded. Not all debug message are controlled by log class.

LS_DEBUG_BLD sets the log class and is used in combination with MASK, which sets the log level. For example:

```
LS_LOG_MASK=LOG_DEBUG  LS_DEBUG_BLD="LC_TRACE"
```

To specify multiple log classes, use a space-separated list enclosed in quotation marks. For example:

```
LS_DEBUG_BLD="LC_TRACE"
```

You need to restart the bld daemon after setting LS_DEBUG_BLD for your changes to take effect.

If you use the command bladmin blddebug to temporarily change this parameter without changing lsf.licensescheduler, you do not need to restart the daemons.

Valid values

Valid log classes are:

- LC_AUTH - Log authentication messages
- LC_COMM - Log communication messages
- LC_FLEX - Log everything related to FLEX_STAT or FLEX_EXEC Macrovision APIs
- LC_LICENSE - Log license management messages (LC_LICENCE is also supported for backward compatibility)
- LC_PREEMPT - Log license preemption policy messages
- LC_RESREQ - Log resource requirement messages
- LC_TRACE - Log significant program walk steps
- LC_XDR - Log everything transferred by XDR

Valid values

Valid log classes are the same as for LS_DEBUG_CMD.

Default

Not defined.

LS_ENABLE_MAX_PREEMPT

Syntax

LS_ENABLE_MAX_PREEMPT=Y

Description

Enables maximum preemption time checking for taskman jobs.

When LS_ENABLE_MAX_PREEMPT is disabled, preemption times for taskman job are not checked regardless of the value of parameters LS_MAX_TASKMAN_PREEMPT in lsf.licensescheduler and MAX_JOB_PREEMPT in lsb.queues, lsb.applications, or lsb.params.

Default

N

LS_LOG_MASK

Syntax

LS_LOG_MASK=_message_log_level_

Description

Specifies the logging level of error messages for LSF License Scheduler daemons. If LS_LOG_MASK is not defined in lsf.licensescheduler, the value of LSF_LOG_MASK in lsf.conf is used. If neither LS_LOG_MASK nor LSF_LOG_MASK is defined, the default is LOG_WARNING.

For example:

```
LS_LOG_MASK=LOG_DEBUG
```

The log levels in order from highest to lowest are:

• LOG_WARNING
• LOG_DEBUG
• LOG_DEBUG1
• LOG_DEBUG2
• LOG_DEBUG3

The most important License Scheduler log messages are at the LOG_WARNING level. Messages at the LOG_DEBUG level are only useful for debugging.

Although message log level implements similar functionality to UNIX syslog, there is no dependency on UNIX syslog. It works even if messages are being logged to files instead of syslog.

License Scheduler logs error messages in different levels so that you can choose to log all messages, or only log messages that are deemed critical. The level specified by LS_LOG_MASK determines which messages are recorded and which are discarded. All messages logged at the specified level or higher are recorded, while lower level messages are discarded.

For debugging purposes, the level LOG_DEBUG contains the fewest number of debugging messages and is used for basic debugging. The level LOG_DEBUG3 records all debugging messages, and can cause log files to grow very large; it is not often used. Most debugging is done at the level LOG_DEBUG2.

Default

LOG_WARNING

LS_MAX_TASKMAN_PREEMPT

Syntax

LS_MAX_TASKMAN_PREEMPT=*integer*

Description

Defines the maximum number of times `taskman` jobs can be preempted.

Maximum preemption time checking for all jobs is enabled by LS_ENABLE_MAX_PREEMPT.

Default

unlimited

LS_MAX_TASKMAN_SESSIONS

Syntax

LS_MAX_TASKMAN_SESSIONS=*integer*

Description

Defines the maximum number of `taskman` jobs that run simultaneously. This prevents system-wide performance issues that occur if there are a large number of `taskman` jobs running in License Scheduler.

LS_PREEMPT_PEER

Syntax

LS_PREEMPT_PEER=Y

Description

Enables bottom-up license token preemption in hierarchical project group configuration. License Scheduler attempts to preempt tokens from the closest projects in the hierarchy first. This balances token ownership from the bottom up.

Default

Not defined. Token preemption in hierarchical project groups is top down.

PORT

Syntax

PORT=*integer*

Description

Defines the TCP listening port used by License Scheduler hosts, including candidate License Scheduler hosts. Specify any non-privileged port number.

BLC_HEARTBEAT_FACTOR

Syntax

BLC_HEARTBEAT_FACTOR=*integer*

Description

Enables bld to detect blcollect failure. Defines the number of times that bld receives no response from a license collector daemon (blcollect) before bld resets the values for that collector to zero. Each license usage reported to bld by the collector is treated as a heartbeat.

Default

3

Clusters section

Description

Required. Lists the clusters that can use License Scheduler.

When configuring clusters for a WAN, the Clusters section of the master cluster must define its slave clusters.

Clusters section structure

The Clusters section begins and ends with the lines Begin Clusters and End Clusters. The second line is the column heading, CLUSTERS. Subsequent lines list participating clusters, one name per line:

```
Begin Clusters

CLUSTERS

cluster1

cluster2

End Clusters
```

CLUSTERS

Defines the name of each participating LSF cluster. Specify using one name per line.

ServiceDomain section

Description

Required. Defines License Scheduler service domains as groups of physical license server hosts that serve a specific network.

ServiceDomain section structure

Define a section for each License Scheduler service domain.

This example shows the structure of the section:

```
Begin ServiceDomain
NAME=DesignCenterB
LIC_SERVERS=((1888@hostD)(1888@hostE))
LIC_COLLECTOR=CenterB
End ServiceDomain
```

Parameters

- NAME
- LIC_SERVERS
- LIC_COLLECTOR
- LM_STAT_INTERVAL

NAME

Defines the name of the service domain.

LIC_SERVERS

Syntax

LIC_SERVERS=([(*host_name* | *port_number@host_name* |(*port_number@host_name port_number@host_name port_number@host_name*))] ...**)**

Description

Defines the FLEXlm license server hosts that make up the License Scheduler service domain. For each FLEXlm license server host, specify the number of the port that FLEXlm uses, then the at symbol (@), then the name of the host. If FLEXlm uses the default port on a host, you can specify the host name without the port number. Put one set of parentheses around the list, and one more set of parentheses around each host, unless you have redundant servers (three hosts sharing one license file). If you have redundant servers, the parentheses enclose all three hosts.

Examples

- One FLEXlm license server host:
  ```
  LIC_SERVERS=((1700@hostA))
  ```
- Multiple FLEXlm license server hosts with unique license.dat files:
  ```
  LIC_SERVERS=((1700@hostA)(1700@hostB)(1700@hostC))
  ```
- Redundant FLEXlm license server hosts sharing the same license.dat file:
  ```
  LIC_SERVERS=((1700@hostD 1700@hostE 1700@hostF))
  ```

LIC_COLLECTOR

Syntax

LIC_COLLECTOR=*licence_collector_name*

Description

Optional. Defines a name for the license collector daemon (blcollect) to use in each service domain. blcollect collects license usage information from FLEXlm and passes it to the

License Scheduler daemon (bld). It improves performance by allowing you to distribute license information queries on multiple hosts.

You can only specify one collector per service domain, but you can specify one collector to serve multiple service domains. Each time you run bl collect, you must specify the name of the collector for the service domain. You can use any name you want.

Default

Undefined. The License Scheduler daemon uses one license collector daemon for the entire cluster.

LM_STAT_INTERVAL

Syntax

LM_STAT_INTERVAL=*seconds*

Description

Defines a time interval between calls that License Scheduler makes to collect license usage information from FLEXlm license management.

The value specified for a service domain overrides the global value defined in the Parameters section. Each service domain definition can specify a different value for this parameter.

Default

Undefined: License Scheduler applies the global value.

Feature section

Description

Required. Defines license distribution policies.

Feature section structure

Define a section for each feature managed by License Scheduler.

```
Begin Feature
NAME=vcs
FLEX_NAME=vcs
DISTRIBUTION=lanserver1 (Lp1 1 Lp2 4/6)
lanserver2 (Lp3 1 Lp4 10/8)
wanserver (Lp1 1 Lp2 1 Lp3 1 Lp4 1)
End Feature
```

Parameters

- NAME
- FLEX_NAME
- DISTRIBUTION
- ALLOCATION
- GROUP

- GROUP_DISTRIBUTION
- LOCAL_TO
- LS_FEATURE_PERCENTAGE
- NON_SHARED_DISTRIBUTION
- PREEMPT_RESERVE
- SERVICE_DOMAINS
- WORKLOAD_DISTRIBUTION
- ENABLE_DYNAMIC_RUSAGE
- DYNAMIC
- LM_REMOVE_INTERVAL
- ENABLE_MINJOB_PREEMPTION

NAME

Required. Defines the token name—the name used by License Scheduler and LSF to identify the license feature.

Normally, license token names should be the same as the FLEXlm Licensing feature names, as they represent the same license. However, LSF does not support names that start with a number, or names containing a dash or hyphen character (-), which may be used in the FLEXlm Licensing feature name.

FLEX_NAME

Optional. Defines the feature name—the name used by FLEXlm to identify the type of license. You only need to specify this parameter if the License Scheduler token name is not identical to the FLEXlm feature name.

FLEX_NAME allows the NAME parameter to be an alias of the FLEXlm feature name. For feature names that start with a number or contain a dash (-), you must set both NAME and FLEX_NAME, where FLEX_NAME is the actual FLEXlm Licensing feature name, and NAME is an arbitrary license token name you choose.

For example

```
Begin Feature
FLEX_NAME=201-AppZ
NAME=AppZ201
DISTRIBUTION=LanServer1(Lp1 1 Lp2 1)
End Feature
```

DISTRIBUTION

Syntax

DISTRIBUTION=[*service_domain_name*([*project_name number_shares*[/*number_licenses_owned*]] ... [default])] ...

service_domain_name

Specify a License Scheduler service domain (described in the ServiceDomain section) that distributes the licenses.

project_name

Specify a License Scheduler project (described in the Projects section) that is allowed to use the licenses.

number_shares

Specify a positive integer representing the number of shares assigned to the project.

The number of shares assigned to a project is only meaningful when you compare it to the number assigned to other projects, or to the total number assigned by the service domain. The total number of shares is the sum of the shares assigned to each project.

number_licenses_owned

Optional. Specify a slash (/) and a positive integer representing the number of licenses that the project owns.

default

A reserved keyword that represents the default License Scheduler project if the job submission does not specify a project (bsub -Lp).

Default includes all projects that have not been defined in the PROJECTS section of lsf.licensescheduler. Jobs that belong to projects that are defined in lsf.licensescheduler do not get a share of the tokens when the project is not explicitly defined in the distribution.

Description

Required if GROUP_DISTRIBUTION is not defined. Defines the distribution policies for the license. The name of each service domain is followed by its distribution policy, in parentheses. The distribution policy determines how the licenses available in each service domain are distributed among the clients.

The distribution policy is a space-separated list with each project name followed by its share assignment. The share assignment determines what fraction of available licenses is assigned to each project, in the event of competition between projects. Optionally, the share assignment is followed by a slash and the number of licenses owned by that project. License ownership enables a preemption policy. (In the event of competition between projects, projects that own licenses preempt jobs. Licenses are returned to the owner immediately.)

GROUP_DISTRIBUTION and DISTRIBUTION are mutually exclusive. If they are both defined in the same feature, the License Scheduler daemon returns an error and ignores this feature.

Examples

```
DISTRIBUTION=wanserver (Lp1 1 Lp2 1 Lp3 1 Lp4 1)
```

In this example, the service domain named wanserver shares licenses equally among four License Scheduler projects. If all projects are competing for a total of eight licenses, each project is entitled to two licenses at all times. If all projects are competing for only two licenses in total, each project is entitled to a license half the time.

```
DISTRIBUTION=lanserver1 (Lp1 1 Lp2 2/6)
```

In this example, the service domain named lanserver1 allows Lp1 to use one third of the available licenses and Lp2 can use two thirds of the licenses. However, Lp2 is always entitled to six licenses, and can preempt another project to get the licenses immediately if they are needed. If the projects are competing for a total of 12 licenses, Lp2 is entitled to eight licenses (six on demand, and two more as soon as they are free). If the projects are competing for only

six licenses in total, Lp2 is entitled to all of them, and Lp1 can only use licenses when Lp2 does not need them.

ALLOCATION

Syntax

ALLOCATION=*project_name* (*cluster_name* [*number_shares*] ... **)**] ...

cluster_name

Specify LSF cluster names that licenses are to be allocated to.

project_name

Specify a License Scheduler project (described in the PROJECTS section) that is allowed to use the licenses.

number_shares

Specify a positive integer representing the number of shares assigned to the cluster.

The number of shares assigned to a cluster is only meaningful when you compare it to the number assigned to other clusters. The total number of shares is the sum of the shares assigned to each cluster.

Description

Defines the allocation of license features across clusters and between LSF jobs and non-LSF interactive jobs.

ALLOCATION ignores the global setting of the ENABLE_INTERACTIVE parameter because ALLOCATION is configured for the license feature.

You can configure the allocation of license shares to:

- Change the share number between clusters for a feature
- Limit the scope of license usage and change the share number between LSF jobs and interactive tasks for a feature

Tip:

To manage interactive (non-LSF) tasks in License Scheduler projects, you require the LSF Task Manager, taskman. The Task Manager utility is supported by License Scheduler. For more information about taskman, contact Platform.

Default

Undefined. If ENABLE_INTERACTIVE is not set, each cluster receives one share, and interactive tasks receive no shares.

Each example contains two clusters and 12 licenses of a specific feature.

Example 1

ALLOCATION is not configured. The ENABLE_INTERACTIVE parameter is not set.

```
Begin Parameters
 ...
ENABLE_INTERACTIVE=n
 ...
End Parameters
Begin Feature
NAME=ApplicationX
DISTRIBUTION=LicenseServer1 (Lp1 1)
End Feature
```

Six licenses are allocated to each cluster. No licenses are allocated to interactive tasks.

Example 2

ALLOCATION is not configured. The ENABLE_INTERACTIVE parameter is set.

```
Begin Parameters
 ...
ENABLE_INTERACTIVE=y
 ...
End Parameters
Begin Feature
NAME=ApplicationX
DISTRIBUTION=LicenseServer1 (Lp1 1)
End Feature
```

Four licenses are allocated to each cluster. Four licenses are allocated to interactive tasks.

Example 3

In the following example, the ENABLE_INTERACTIVE parameter does not affect the ALLOCATION configuration of the feature.

ALLOCATION is configured. The ENABLE_INTERACTIVE parameter is set.

```
Begin Parameters
 ...
ENABLE_INTERACTIVE=y
 ...
End Parameters
Begin Feature
NAME=ApplicationY
DISTRIBUTION=LicenseServer1 (Lp2 1)
ALLOCATION=Lp2(cluster1 1 cluster2 0 interactive 1)
End Feature
```

The ENABLE_INTERACTIVE setting is ignored. Licenses are shared equally between cluster1 and interactive tasks. Six licenses of ApplicationY are allocated to cluster1. Six licenses are allocated to interactive tasks.

Example 4

In the following example, the ENABLE_INTERACTIVE parameter does not affect the ALLOCATION configuration of the feature.

ALLOCATION is configured. The ENABLE_INTERACTIVE parameter is not set.

```
Begin Parameters

 ...

ENABLE_INTERACTIVE=n

...

End Parameters
Begin Feature
NAME=ApplicationZ
DISTRIBUTION=LicenseServer1 (Lp1 1)
ALLOCATION=Lp1(cluster1 0 cluster2 1 interactive 2)
End Feature
```

The ENABLE_INTERACTIVE setting is ignored. Four licenses of ApplicationZ are allocated to cluster2. Eight licenses are allocated to interactive tasks.

GROUP

Syntax

GROUP=[*group_name*(*project_name...*)] ...

group_name

Specify a name for a group of projects.

project_name

Specify a License Scheduler project (described in the PROJECTS section) that is allowed to use the licenses. The project must appear in the DISTRIBUTION.

A project should only belong to one group.

Description

Optional. Defines groups of projects and specifies the name of each group. The groups defined here are used for group preemption and replace single projects with group projects.

This parameter is ignored if GROUP_DISTRIBUTION is also defined.

GROUP_DISTRIBUTION

Syntax

GROUP_DISTRIBUTION=*top_level_hierarchy_name*

top_level_hierarchy_name

Specify the name of the top level hierarchical group.

Description

Required if DISTRIBUTION is not defined. Defines the name of the hierarchical group containing the distribution policy attached to this feature.

GROUP_DISTRIBUTION and DISTRIBUTION are mutually exclusive. If they are both defined in the same feature, the License Scheduler daemon returns an error and ignores this feature.

If GROUP is also defined, it is ignored in favour of GROUP_DISTRIBUTION.

Example

The following example shows the GROUP_DISTRIBUTION parameter hierarchical scheduling for the top-level hierarchical group named groups. The SERVICE_DOMAINS parameter defines a list of service domains that provide tokens for the group.

```
Begin Feature

NAME = myjob2

GROUP_DISTRIBUTION = groups

SERVICE_DOMAINS = LanServer wanServer

End Feature
```

LOCAL_TO
Syntax

LOCAL_TO=*cluster_name* | *location_name***(***cluster_name* [*cluster_name* ...]**)**

Description

Configures token locality for the license feature. You must configure different feature sections for same feature based on their locality. By default, If LOCAL_TO is not defined, the feature is available to all clients and is not restricted by geographical location. When LOCAL_TO is configured, for a feature, License Scheduler treats license features served to different locations as different token names, and distributes the tokens to projects according the distribution and allocation policies for the feature.

LOCAL_TO allows you to limit features from different service domains to specific clusters, so License Scheduler only grants tokens of a feature to jobs from clusters that are entitled to them.

For example, if your license servers restrict the serving of license tokens to specific geographical locations, use LOCAL_TO to specify the locality of a license token if any feature cannot be shared across all the locations. This avoids having to define different distribution and allocation policies for different service domains, and allows hierarchical group configurations.

License Scheduler manages features with different localities are different resources. Use bl info and bl stat to see the different resource information for the features depending on their cluster locality.

License features with different localities must be defined in different feature sections. The same Service Domain can appear only once in the configuration for a given license feature.

A configuration like LOCAL_TO=Site1(clusterA clusterB) configures the feature for more than one cluster.

A configuration like LOCAL_TO=clusterA configures locality for only one cluster. This is the same as LOCAL_TO=clusterA(clusterA).

Cluster names must be the names of clusters defined in the Clusters section of lsf.licensescheduler.

Examples

```
Begin Feature
NAME = hspice
DISTRIBUTION = SD1 (Lp1 1 Lp2 1)
LOCAL_TO = siteUS(clusterA clusterB)
End Feature
```

```
Begin Feature
NAME = hspice
DISTRIBUTION = SD2 (Lp1 1 Lp2 1)
LOCAL_TO = clusterA
End Feature
```

```
Begin Feature
NAME = hspice
DISTRIBUTION = SD3 (Lp1 1 Lp2 1) SD4 (Lp1 1 Lp2 1)
End Feature
```

Or use the hierarchical group configuration (GROUP_DISTRIBUTION):

```
Begin Feature
NAME = hspice
GROUP_DISTRIBUTION = group1
SERVICE_DOMAINS = SD1
LOCAL_TO = siteUS(clusterA clusterB)
End Feature
```

```
Begin Feature
NAME = hspice
GROUP_DISTRIBUTION = group1
SERVICE_DOMAINS = SD2
LOCAL_TO = clusterA
End Feature
```

```
Begin Feature
NAME = hspice
GROUP_DISTRIBUTION = group1
SERVICE_DOMAINS = SD3 SD4
End Feature
```

Default

Not defined. The feature is available to all clusters and interactive jobs, and is not restricted by cluster.

LS_FEATURE_PERCENTAGE

Syntax

LS_FEATURE_PERCENTAGE=Y | N

Description

Configures license ownership in percentages instead of absolute numbers. When not combined with hierarchical projects, affects DISTRIBUTED and NON_SHARED_DISTRIBUTION values only. When using hierarchical projects, percentage is applied to OWNERSHIP, LIMITS, and NON_SHARED values.

Example 1

```
Begin Feature

LS_FEATURE_PERCENTAGE = Y

DISTRIBUTION = LanServer (p1 1 p2 1 p3 1/20)

...

End Feature
```

The service domain LanServer shares licenses equally among three License Scheduler projects. P3 is always entitled to 20% of the total licenses, and can preempt another project to get the licenses immediately if they are needed.

Example 2

With LS_FEATURE_PERCENTAGE=Y in feature section and using hierarchical project groups:

```
Begin ProjectGroup
```

GROUP	SHARES		OWNERSHIP	LIMITS	NON_SHARED
(R (A p4))	(1	1)	()	()	()
(A (B p3))	(1	1)	(- 10)	(- 20)	()
(B (p1 p2))	(1	1)	(30 -)	()	(- 5)

```
End ProjectGroup
```

Project p1 owns 30% of the total licenses, and project p3 owns 10% of total licenses. P3's LIMITS is 20% of total licenses, and p2's NON_SHARED is 5%.

Default

N (Ownership is not configured with percentages but with absolute numbers.)

NON_SHARED_DISTRIBUTION

Syntax

NON_SHARED_DISTRIBUTION=*service_domain_name* ([*project_name number_non_shared_licenses*] ...) ...

service_domain_name

Specify a License Scheduler service domain (described in the ServiceDomain section) that distributes the licenses.

project_name

Specify a License Scheduler project (described in the Projects section) that is allowed to use the licenses.

number_non_shared_licenses

Specify a positive integer representing the number of non-shared licenses that the project owns.

Description

Optional. Defines non-shared licenses. Non-shared licenses are not shared with other license projects. They are available only to that project.

Use blinfo -a to display NON_SHARED_DISTRIBUTION information.

Example

```
Begin Feature
NAME=f1   # total 15 on LanServer and 15 on WanServer
FLEX_NAME=VCS-RUNTIME
DISTRIBUTION=LanServer(Lp1 4 Lp2 1) WanServer (Lp1 1 Lp2 1/3)
NON_SHARED_DISTRIBUTION=LanServer(Lp1 10) WanServer (Lp1 5 Lp2 3)
PREEMPT_RESERVE=Y
End Feature
```

In this example:

- 10 non-shared licenses are defined for the Lp1 project on LanServer
- 5 non-shared licenses are defined for the Lp1 project on WanServer
- 3 non-shared licenses are defined for the Lp2 project on WanServer

The remaining licenses are distributed as follows:

- LanServer: The remaining 5 (15-10=5) licenses on LanServer is distributed to the Lp1 and Lp2 projects with a 4:1 ratio.
- WanServer: The remaining 7 (15-5-3=7) licenses on WanServer is distributed to the Lp1 and Lp2 projects with a 1:1 ratio. If Lp2 uses fewer than 6 (3 privately owned+ 3 owned) licenses, then a job in the Lp2 can preempt Lp1 jobs.

PREEMPT_LSF

Syntax

PREEMPT_LSF=Y

Description

Optional. With the flex grid interface integration installed, enables on-demand preemption of LSF jobs for important non-managed workload. This guarantees that important non-managed jobs do not fail because of lack of licenses.

Default

LSF workload is not preemtable

PREEMPT_RESERVE

Syntax

PREEMPT_RESERVE=Y

Description

Optional. Enables License Scheduler to preempt either licenses that are reserved or already in use by other projects. The number of jobs must be greater than the number of licenses owned.

Default

Y: reserved licenses are preemtable

SERVICE_DOMAINS

Syntax

SERVICE_DOMAINS=*service_domain_name* ...

service_domain_name

Specify the name of the service domain.

Description

Required if GROUP_DISTRIBUTION is defined. Specifies the service domains that provide tokens for this feature.

WORKLOAD_DISTRIBUTION

Syntax

WORKLOAD_DISTRIBUTION=[*service_domain_name***(LSF** *lsf_distribution* [*/ enforced_distribution*] **NON_LSF** *non_lsf_distribution***)**] ...

service_domain_name

Specify a License Scheduler service domain (described in the ServiceDomain section) that distributes the licenses.

lsf_distribution

Specify the share of licenses dedicated to LSF workloads. The share of licenses dedicated to LSF workloads is a ratio of *lsf_distribution:non_lsf_distribution*.

enforced_distribution

Optional. Specify a slash (/) and a positive integer representing the enforced number of licenses.

non_lsf_distribution

Specify the share of licenses dedicated to non-LSF workloads. The share of licenses dedicated to non-LSF workloads is a ratio of *non_lsf_distribution:lsf_distribution*.

Description

Optional. Defines the distribution given to each LSF and non-LSF workload within the specified service domain.

Use bl info - a to display WORKLOAD_DISTRIBUTION configuration.

Example 1

```
Begin Feature
NAME=ApplicationX
DISTRIBUTION=LicenseServer1(Lp1 1 Lp2 2)
WORKLOAD_DISTRIBUTION=LicenseServer1(LSF 8 NON_LSF 2)
End Feature
```

On the LicenseServer1 domain, the available licenses are dedicated in a ratio of 8:2 for LSF and non-LSF workloads. This means that 80% of the available licenses are dedicated to the LSF workload, and 20% of the available licenses are dedicated to the non-LSF workload.

If LicenseServer1 has a total of 80 licenses, this configuration indicates that 64 licenses are dedicated to the LSF workload, and 16 licenses are dedicated to the non-LSF workload.

Example 2

```
Begin Feature
NAME=ApplicationX
DISTRIBUTION=LicenseServer1(Lp1 1 Lp2 2)
WORKLOAD_DISTRIBUTION=LicenseServer1(LSF 8/40 NON_LSF 2)
End Feature
```

On the LicenseServer1 domain, the available licenses are dedicated in a ratio of 8:2 for LSF and non-LSF workloads, with an absolute maximum of 40 licenses dedicated to the LSF workload. This means that 80% of the available licenses, up to a maximum of 40, are dedicated to the LSF workload, and the remaining licenses are dedicated to the non-LSF workload.

If LicenseServer1 has a total of 40 licenses, this configuration indicates that 32 licenses are dedicated to the LSF workload, and eight licenses are dedicated to the non-LSF workload. However, if LicenseServer1 has a total of 80 licenses, only 40 licenses are dedicated to the LSF workload, and the remaining 40 licenses are dedicated to the non-LSF workload.

ENABLE_DYNAMIC_RUSAGE

Syntax

ENABLE_DYNAMIC_RUSAGE=Y

Description

Enforces license distribution policies for class-C license features.

When set, ENABLE_DYNAMIC_RUSAGE enables all class-C license checkouts to be considered managed checkout, instead of unmanaged (or OTHERS).

DYNAMIC

Syntax

DYNAMIC=Y

Description

If you specify DYNAMIC=Y, you must specify a duration in an rusage resource requirement for the feature. This enables License Scheduler to treat the license as a dynamic resource and prevents License Scheduler from scheduling tokens for the feature when they are not available, or reserving license tokens when they should actually be free.

LM_REMOVE_INTERVAL

Syntax

LM_REMOVE_INTERVAL=_seconds_

Description

Specifies the minimum time a job must have a license checked out before lmremove can remove the license. lmremove causes lmgrd and vendor daemons to close the TCP connection with the application. They then retry the license checkout.

The value specified for a feature overrides the global value defined in the Parameters section. Each feature definition can specify a different value for this parameter.

Default

Undefined: License Scheduler applies the global value.

ENABLE_MINJOB_PREEMPTION

Syntax

ENABLE_MINJOB_PREEMPTION=Y

Description

Minimizes the overall number of preempted jobs by enabling job list optimization. For example, for a job that requires 10 licenses, License Scheduler preempts one job that uses 10 or more licenses rather than 10 jobs that each use one license.

Default

Undefined: License Scheduler does not optimize the job list when selecting jobs to preempt.

FeatureGroup section

Description

Optional. Collects license features into groups. Put FeatureGroup sections after Feature sections in lsf.licensescheduler.

FeatureGroup section structure

The FeatureGroup section begins and ends with the lines Begin FeatureGroup and End FeatureGroup. Feature group definition consists of a unique name and a list of features contained in the feature group.

Example

```
Begin FeatureGroup
NAME = Synposys
FEATURE_LIST = ASTRO VCS_Runtime_Net Hsim Hspice
End FeatureGroup
Begin FeatureGroup
NAME = Cadence
FEATURE_LIST = Encounter NCSim  NCVerilog
End FeatureGroup
```

Parameters

- NAME
- FEATURE_LIST

NAME

Required. Defines the name of the feature group. The name must be unique.

FEATURE_LIST

Required. Lists the license features contained in the feature group.The feature names in FEATURE_LIST must already be defined in Feature sections. Feature names cannot be repeated in the FEATURE_LIST of one feature group. The FEATURE_LIST cannot be empty. Different feature groups can have the same features in their FEATURE_LIST.

ProjectGroup section

Description

Optional. Defines the hierarchical relationships of projects.

The hierarchical groups can have multiple levels of grouping. You can configure a tree-like scheduling policy, with the leaves being the license projects that jobs can belong to. Each project group in the tree has a set of values, including shares, limits, ownership and non-shared, or exclusive, licenses.

Use bl stat -G to view the hierarchical dynamic license information.

Use bl info -G to view the hierarchical configuration.

ProjectGroup section structure

Define a section for each hierarchical group managed by License Scheduler.

The keywords GROUP, SHARES, OWNERSHIP, LIMIT, and NON_SHARED are required. The keyword PRIORITY is optional. Empty brackets are allowed only for OWNERSHIP, LIMIT, and PRIORITY. SHARES must be specified.

Begin	ProjectGroupGROUP	SHARES	OWNERSHIP	LIMITS	NON_SHARED	PRIORITY
(root(A B C))	(1 1 1)	()	()	()	(3 2 -)	
(A (P1 D))	(1 1)	()	()	()	(3 5)	
(B (P4 P5))	(1 1)	()	()	()	()	
(C (P6 P7 P8))	(1 1 1)	()	()	()	(8 3 0)	
(D (P2 P3))	(1 1)	()	()	()	(2 1)	
End ProjectGroup						

Parameters

- GROUP
- SHARES
- OWNERSHIP
- LIMITS
- NON_SHARED
- PRIORITY
- DESCRIPTION

GROUP

Defines the project names in the hierarchical grouping and its relationships. Each entry specifies the name of the hierarchical group and its members.

For better readability, you should specify the projects in the order from the root to the leaves as in the example.

Specify the entry as follows:

(*group* (*member* ...))

SHARES

Required. Defines the shares assigned to the hierarchical group member projects. Specify the share for each member, separated by spaces, in the same order as listed in the GROUP column.

OWNERSHIP

Defines the level of ownership of the hierarchical group member projects. Specify the ownership for each member, separated by spaces, in the same order as listed in the GROUP column.

You can only define OWNERSHIP for hierarchical group member projects, not hierarchical groups. Do not define OWNERSHIP for the top level (root) project group. Ownership of a given internal node is the sum of the ownership of all child nodes it directly governs.

A dash (-) is equivalent to a zero, which means there are no owners of the projects. You can leave the parentheses empty () if desired.

Valid values

A positive integer between the NON_SHARED and LIMITS values defined for the specified hierarchical group.

- If defined as less than NON_SHARED, OWNERSHIP is set to NON_SHARED.
- If defined as greater than LIMITS, OWNERSHIP is set to LIMITS.

LIMITS

Defines the maximum number of licenses that can be used at any one time by the hierarchical group member projects. Specify the maximum number of licenses for each member, separated by spaces, in the same order as listed in the GROUP column.

A dash (-) is equivalent to INFINIT_INT, which means there is no maximum limit and the project group can use as many licenses as possible.

You can leave the parentheses empty () if desired.

NON_SHARED

Defines the number of licenses that the hierarchical group member projects use exclusively. Specify the number of licenses for each group or project, separated by spaces, in the same order as listed in the GROUP column.

A dash (-) is equivalent to a zero, which means there are no licenses that the hierarchical group member projects use exclusively.

Normally, the total number of non-shared licenses should be less than the total number of license tokens available. License tokens may not be available to project groups if the total non-shared licenses for all groups is greater than the number of shared tokens available.

For example, feature p4_4 is configured as follows, with a total of 4 tokens:

```
Begin Feature NAME =p4_4 # total token value is 4 GROUP_DISTRIBUTION=final
SERVICE_DOMAINS=LanServer End Feature
```

The correct configuration is:

GROUP	SHARES	OWNERSHIP	LIMITS	NON_SHARED
(final (G2 G1))	(1 1)	()	()	(2 0)
(G1 (AP2 AP1))	(1 1)	()	()	(1 1)

Valid values

Any positive integer up to the LIMITS value defined for the specified hierarchical group.

If defined as greater than LIMITS, NON_SHARED is set to LIMITS.

PRIORITY

Optional. Defines the priority assigned to the hierarchical group member projects. Specify the priority for each member, separated by spaces, in the same order as listed in the GROUP column.

"0" is the lowest priority, and a higher number specifies a higher priority. This column overrides the default behavior. Instead of preempting based on the accumulated inuse usage

of each project, the projects are preempted according to the specified priority from lowest to highest.

By default, priorities are evaluated top down in the project group hierarchy. The priority of a given node is first decided by the priority of the parent groups. When two nodes have the same priority, priority is determined by the accumulated inuse usage of each project at the time the priorities are evaluated. Specify LS_PREEMPT_PEER=Y in the Parametersr section to enable bottom-up license token preemption in hierarchical project group configuration.

A dash (-) is equivalent to a zero, which means there is no priority for the project. You can leave the parentheses empty () if desired.

Use blinfo -G to view hierarchical project group priority information.

Priority of default project

If not explicitly configured, the default project has the priority of 0. You can override this value by explicitly configuring the default project in Projects section with the chosen priority value.

DESCRIPTION

Optional. Description of the project group.

The text can include any characters, including white space. The text can be extended to multiple lines by ending the preceding line with a backslash (\). The maximum length for the text is 64 characters.

Use blinfo -G to view hierarchical project group description.

Projects section

Description

Required. Lists the License Scheduler projects.

Projects section structure

The Projects section begins and ends with the lines Begin Projects and End Projects. The second line consists of the required column heading PROJECTS and the optional column heading PRIORITY. Subsequent lines list participating projects, one name per line.

Examples

The following example lists the projects without defining the priority:

```
Begin Projects PROJECTS Lp1 Lp2 Lp3 Lp4 ... End Projects
```

The following example lists the projects and defines the priority of each project:

```
Begin Projects

PROJECTS        PRIORITY

Lp1             3

Lp2             4

Lp3             2

Lp4             1

default         0

...

End Projects
```

Parameters

- PROJECTS
- PRIORITY
- DESCRIPTION

PROJECTS

Defines the name of each participating project. Specify using one name per line.

PRIORITY

Optional. Defines the priority for each project where "0" is the lowest priority, and the higher number specifies a higher priority. This column overrides the default behavior. Instead of preempting in order the projects are listed under PROJECTS based on the accumulated inuse usage of each project, the projects are preempted according to the specified priority from lowest to highest.

When 2 projects have the same priority number configured, the first project listed has higher priority, like LSF queues.

Use blinfo -Lp to view project priority information.

Priority of default project

If not explicitly configured, the default project has the priority of 0. You can override this value by explicitly configuring the default project in Projects section with the chosen priority value.

DESCRIPTION

Optional. Description of the project.

The text can include any characters, including white space. The text can be extended to multiple lines by ending the preceding line with a backslash (\). The maximum length for the text is 64 characters.

Use blinfo -Lp to view the project description.

Automatic time-based configuration

Variable configuration is used to automatically change LSF License Scheduler license token distribution policy configuration based on time windows. You define automatic configuration changes in lsf.licensescheduler by using if-else constructs and time expressions in the Feature section. After you change the file, check the configuration with the bladmin

ckconfig command, and restart License Scheduler the cluster with the bladmin reconfig command.

The expressions are evaluated by License Scheduler every 10 minutes based on the bld start time. When an expression evaluates true, License Scheduler dynamically changes the configuration based on the associated configuration statements. Reconfiguration is done in real time without restarting bld, providing continuous system availability.

Example

```
Begin Feature
NAME = f1
#if time(5:16:30-1:8:30 20:00-8:30)
DISTRIBUTION=Lan(P1 2/5  P2 1)
#elif time(3:8:30-3:18:30)
DISTRIBUTION=Lan(P3 1)
#else
DISTRIBUTION=Lan(P1 1 P2 2/5)
#endif
End Feature
```

lsf.shared

The lsf.shared file contains common definitions that are shared by all load sharing clusters defined by lsf.cluster.*cluster_name* files. This includes lists of cluster names, host types, host models, the special resources available, and external load indices, including indices required to submit jobs using JSDL files.

This file is installed by default in the directory defined by LSF_CONFDIR.

Changing lsf.shared configuration

After making any changes to lsf.shared, run the following commands:

* lsadmin reconfig to reconfigure LIM
* badmin mbdrestart to restart mbatchd

Cluster section

(Required) Lists the cluster names recognized by the LSF system

Cluster section structure

The first line must contain the mandatory keyword ClusterName. The other keyword is optional.

The first line must contain the mandatory keyword ClusterName and the keyword Servers in a MultiCluster environment.

Each subsequent line defines one cluster.

Example Cluster section

```
Begin Cluster
ClusterName   Servers
cluster1      hostA
cluster2      hostB
End Cluster
```

ClusterName

Defines all cluster names recognized by the LSF system.

All cluster names referenced anywhere in the LSF system must be defined here. The file names of cluster-specific configuration files must end with the associated cluster name.

By default, if MultiCluster is installed, all clusters listed in this section participate in the same MultiCluster environment. However, individual clusters can restrict their MultiCluster participation by specifying a subset of clusters at the cluster level (lsf.cluster.*cluster_name* RemoteClusters section).

Servers

MultiCluster only. List of hosts in this cluster that LIMs in remote clusters can connect to and obtain information from.

For other clusters to work with this cluster, one of these hosts must be running mbatchd.

HostType section

(Required) Lists the valid host types in the cluster. All hosts that can run the same binary executable are in the same host type.

Caution:

If you remove NTX86, NTX64, or NTIA64 from the HostType section, the functionality of `lspasswd.exe` is affected. The `lspasswd` command registers a password for a Windows user account.

HostType section structure

The first line consists of the mandatory keyword TYPENAME.

Subsequent lines name valid host types.

Example HostType section

```
Begin HostType
TYPENAME
SOL64
SOLSPARC
LINUX86LINUXPPC
LINUX64
NTX86
NTX64
NTIA64
End HostType
```

TYPENAME

Host type names are usually based on a combination of the hardware name and operating system. If your site already has a system for naming host types, you can use the same names for LSF.

HostModel section

(Required) Lists models of machines and gives the relative CPU scaling factor for each model. All hosts of the same relative speed are assigned the same host model.

LSF uses the relative CPU scaling factor to normalize the CPU load indices so that jobs are more likely to be sent to faster hosts. The CPU factor affects the calculation of job execution time limits and accounting. Using large or inaccurate values for the CPU factor can cause confusing results when CPU time limits or accounting are used.

HostModel section structure

The first line consists of the mandatory keywords MODELNAME, CPUFACTOR, and ARCHITECTURE.

Subsequent lines define a model and its CPU factor.

Example HostModel section

```
Begin HostModel  MODELNAME  CPUFACTOR       ARCHITECTURE
PC400          13.0            (i86pc_400 i686_400)
PC450          13.2            (i86pc_450 i686_450)
Sparc5F         3.0            (SUNWSPARCstation5_170_sparc)
Sparc20         4.7            (SUNWSPARCstation20_151_sparc)
Ultra5S        10.3            (SUNWUltra5_270_sparcv9 SUNWUltra510_270_sparcv9)
End HostModel
```

ARCHITECTURE

(Reserved for system use only) Indicates automatically detected host models that correspond to the model names.

CPUFACTOR

Though it is not required, you would typically assign a CPU factor of 1.0 to the slowest machine model in your system and higher numbers for the others. For example, for a machine model that executes at twice the speed of your slowest model, a factor of 2.0 should be assigned.

MODELNAME

Generally, you need to identify the distinct host types in your system, such as MIPS and SPARC first, and then the machine models within each, such as SparcIPC, Sparc1, Sparc2, and Sparc10.

About automatically detected host models and types

When you first install LSF, you do not necessarily need to assign models and types to hosts in lsf.cluster.*cluster_name*. If you do not assign models and types to hosts in lsf.cluster.*cluster_name*, LIM automatically detects the model and type for the host.

If you have versions earlier than LSF 4.0, you may have host models and types already assigned to hosts. You can take advantage of automatic detection of host model and type also.

Automatic detection of host model and type is useful because you no longer need to make changes in the configuration files when you upgrade the operating system or hardware of a host and reconfigure the cluster. LSF will automatically detect the change.

Mapping to CPU factors

Automatically detected models are mapped to the short model names in lsf.shared in the ARCHITECTURE column. Model strings in the ARCHITECTURE column are only used for mapping to the short model names.

Example lsf.shared file:

```
Begin HostModel
MODELNAME     CPUFACTOR      ARCHITECTURE
SparcU5        5.0            (SUNWUltra510_270_sparcv9)
PC486          2.0            (i486_33 i486_66)
PowerPC        3.0            (PowerPC12 PowerPC16 PowerPC31)
End HostModel
```

If an automatically detected host model cannot be matched with the short model name, it is matched to the best partial match and a warning message is generated.

If a host model cannot be detected or is not supported, it is assigned the DEFAULT model name and an error message is generated.

Naming convention

Models that are automatically detected are named according to the following convention:

```
hardware_platform [_processor_speed[_processor_type]]
```

where:

- *hardware_platform* is the only mandatory component
- *processor_speed* is the optional clock speed and is used to differentiate computers within a single platform
- *processor_type* is the optional processor manufacturer used to differentiate processors with the same speed
- Underscores (_) between *hardware_platform*, *processor_speed*, *processor_type* are mandatory.

Resource section

Optional. Defines resources (must be done by the LSF administrator).

Resource section structure

The first line consists of the keywords. RESOURCENAME and DESCRIPTION are mandatory. The other keywords are optional. Subsequent lines define resources.

Example Resource section

```
Begin Resource
RESOURCENAME  TYPE     INTERVAL INCREASING CONSUMABLE DESCRIPTION  # Keywords
   patchrev   Numeric  ()       Y          ()         (Patch revision)
   specman    Numeric  ()       N          ()         (Specman)
   switch     Numeric  ()       Y          N          (Network Switch)
   rack       String   ()       ()         ()         (Server room rack)
   owner      String   ()       ()         ()         (Owner of the host)
   elimres    Numeric  10       Y          ()         (elim generated index)
   ostype     String   ()       ()         ()         (Operating system and version)
   lmhostid   String   ()       ()         ()         (FlexLM's lmhostid)
   limversion String   ()       ()         ()         (Version of LIM binary)
End Resource
```

RESOURCENAME

The name you assign to the new resource. An arbitrary character string.

- A resource name cannot begin with a number.
- A resource name cannot contain any of the following characters:

```
:  .  (  )  [  +  -  *  /  !  &  |  <  >  @  =
```

- A resource name cannot be any of the following reserved names:

```
cpu cpuf io logins ls idle maxmem maxswp maxtmp type model status it
```

```
mem ncpus define_ncpus_cores define_ncpus_procs
```

```
define_ncpus_threads ndisks pg r15m r15s r1m swap swp tmp ut
```

- To avoid conflict with inf and nan keywords in 3rd-party libraries, resource names should not begin with inf or nan (upper case or lower case). Resource requirment strings, such as `-R "infra"` or `-R "nano"` will cause an error. Use `-R "defined(infxx)"` or `-R "defined(nanxx)"`, to specify these resource names.
- Resource names are case sensitive
- Resource names can be up to 39 characters in length
- For Solaris machines, the keyword `int` is reserved and cannot be used.

TYPE

The type of resource:

- Boolean—Resources that have a value of 1 on hosts that have the resource and 0 otherwise.
- Numeric—Resources that take numerical values, such as all the load indices, number of processors on a host, or host CPU factor.
- String— Resources that take string values, such as host type, host model, host status.

Default

If TYPE is not given, the default type is Boolean.

INTERVAL

Optional. Applies to dynamic resources only.

Defines the time interval (in seconds) at which the resource is sampled by the ELIM.

If INTERVAL is defined for a numeric resource, it becomes an external load index.

Default

If INTERVAL is not given, the resource is considered static.

INCREASING

Applies to numeric resources only.

If a larger value means greater load, INCREASING should be defined as Y. If a smaller value means greater load, INCREASING should be defined as N.

CONSUMABLE

Explicitly control if a resource is consumable. Applies to static or dynamic numeric resources.

Static and dynamic numeric resources can be specified as consumable. CONSUMABLE is optional. The defaults for the consumable attribute are:

- Built-in indicies:

  - The following are consumable: r15s, r1m, r15m, ut, pg, io, ls, it, tmp, swp, mem.
  - All other built-in static resources are not consumable. (e.g., ncpus, ndisks, maxmem, maxswp, maxtmp, cpuf, type, model, status, rexpri, server, hname).

- External shared resources:

  - All numeric resources are consumable.
  - String and boolean resources are not consumable.

You should only specify consumable resources in the rusage section of a resource requirement string. Non-consumable resources are ignored in rusage sections.

A non-consumable resource should not be releasable. Non-consumable numeric resource should be able to used in order, select and same sections of a resource requirement string.

When LSF_STRICT_RESREQ=Y in lsf.conf, LSF rejects resource requirement strings where an rusage section contains a non-consumable resource.

DESCRIPTION

Brief description of the resource.

The information defined here will be returned by the ls_info() API call or printed out by the lsinfo command as an explanation of the meaning of the resource.

RELEASE

Applies to numeric shared resources only, such as floating licenses.

Controls whether LSF releases the resource when a job using the resource is suspended. When a job using a shared resource is suspended, the resource is held or released by the job depending on the configuration of this parameter.

Specify N to hold the resource, or specify Y to release the resource.

Default

Y

lsf.sudoers

About lsf.sudoers

The `lsf.sudoers` file is an optional file to configure security mechanisms. It is not installed by default.

You use `lsf.sudoers` to set the parameter LSF_EAUTH_KEY to configure a key for `eauth` to encrypt and decrypt user authentication data.

On UNIX, you also use `lsf.sudoers` to grant permission to users other than root to perform certain operations as root in LSF, or as a specified user.

These operations include:

- LSF daemon startup/shutdown
- User ID for LSF authentication
- User ID for LSF pre- and post-execution commands.
- User ID for external LSF executables

If `lsf.sudoers` does not exist, only root can perform these operations in LSF on UNIX.

On UNIX, this file is located in `/etc`.

There is one `lsf.sudoers` file per host.

On Windows, this file is located in the directory specified by the parameter LSF_SECUREDIR in `lsf.conf`.

Changing lsf.sudoers configuration

After making any changes to `lsf.sudoers`, run `badmin reconfig` to reload the configuration files.

lsf.sudoers on UNIX

In LSF, certain operations such as daemon startup can only be performed by root. The `lsf.sudoers` file grants root privileges to specific users or user groups to perform these operations.

Location

> `lsf.sudoers` must be located in `/etc` on each host.

Permissions

> `lsf.sudoers` must have permission 600 and be readable and writable only by root.

lsf.sudoers on Windows

The `lsf.sudoers` file is shared over an NTFS network, not duplicated on every Windows host.

By default, LSF installs `lsf.sudoers` in the %SYSTEMROOT% directory.

The location of `lsf.sudoers` on Windows must be specified by LSF_SECUREDIR in `lsf.conf`. You must configure the LSF_SECUREDIR parameter in `lsf.conf` if using `lsf.sudoers` on Windows.

Windows permissions

Restriction:

The owner of `lsf.sudoers` on Windows be **Administrators**. If not, eauth may not work.

The permissions on `lsf.sudoers` for Windows are:

Workgroup Environment

- Local Admins (W)
- Everyone (R)

Domain Environment

- Domain Admins (W)
- Everyone (R)

File format

The format of `lsf.sudoers` is very similar to that of `lsf.conf`.

Each entry can have one of the following forms:

- `NAME=VALUE`
- `NAME=`
- `NAME= "STRING1 STRING2 ..."`

The equal sign = must follow each `NAME` even if no value follows and there should be no space beside the equal sign.

`NAME` describes an authorized operation.

`VALUE` is a single string or multiple strings separated by spaces and enclosed in quotation marks.

Lines starting with a pound sign (#) are comments and are ignored. Do not use `#if` as this is reserved syntax for time-based configuration.

Example lsf.sudoers File

LSB_PRE_POST_EXEC_USER=user100

LSF_STARTUP_PATH=/usr/share/lsf/etc

LSF_STARTUP_USERS="user1 user10 user55"

Creating and modifying lsf.sudoers

You can create and modify `lsf.sudoers` with a text editor.

After you modify `lsf.sudoers`, you must run `badmin hrestart all` to restart all sbatchds in the cluster with the updated configuration.

Parameters

- LSB_PRE_POST_EXEC_USER
- LSF_EAUTH_KEY
- LSF_EAUTH_USER
- LSF_EEXEC_USER
- LSF_EGO_ADMIN_PASSWD
- LSF_EGO_ADMIN_USER
- LSF_LOAD_PLUGINS

- LSF_STARTUP_PATH
- LSF_STARTUP_USERS

LSB_PRE_POST_EXEC_USER

Syntax

LSB_PRE_POST_EXEC_USER=*user_name*

Description

Specifies the UNIX user account under which pre- and post-execution commands run. This parameter applies only to pre- and post-execution commands configured at the application and queue levels; pre-execution commands defined at the job level with bsub -E run under the account of the user who submits the job.

You can specify only one user account. If the pre-execution or post-execution commands perform privileged operations that require root permissions on UNIX hosts, specify a value of root.

If you configure this parameter as root, the LD_PRELOAD and LD_LIBRARY_PATH variables are removed from the pre-execution, post-execution, and eexec environments for security purposes.

Default

Not defined. Pre-execution and post-execution commands run under the user account of the user who submits the job.

LSF_EAUTH_KEY

Syntax

LSF_EAUTH_KEY=*key*

Description

Applies to UNIX, Windows, and mixed UNIX/Windows clusters.

Specifies the key that eauth uses to encrypt and decrypt user authentication data. Defining this parameter enables increased security at your site. The key must contain at least six characters and must use only printable characters.

For UNIX, you must edit the lsf.sudoers file on all hosts within the cluster and specify the same encryption key. For Windows, you must edit the shared lsf.sudoers file.

Default

Not defined. The eauth executable encrypts and decrypts authentication data using an internal key.

LSF_EAUTH_USER

Syntax

LSF_EAUTH_USER=*user_name*

Description

UNIX or Linux only.

Specifies the UNIX or Linux user account under which the external authentication executable eauth runs.

Default

Not defined. The eauth executable runs under the account of the primary LSF administrator.

LSF_EEXEC_USER

Syntax

LSF_EEXEC_USER=*user_name*

Description

UNIX or Linux only.

Specifies the UNIX or Linux user account under which the external executable eexec runs.

Default

Not defined. The eexec executable runs under root or the account of the user who submitted the job.

LSF_EGO_ADMIN_PASSWD

Syntax

LSF_EGO_ADMIN_PASSWD=*password*

Description

When the EGO Service Controller (EGOSC) is configured to control LSF daemons, enables UNIX and Windows users to bypass the additional login required to start res and sbatchd. Bypassing the EGO administrator login enables the use of scripts to automate system startup.

Specify the Admin EGO cluster administrator password as clear text. You must also define the LSF_EGO_ADMIN_USER parameter.

Default

Not defined. With EGOSC daemon control enabled, the lsadmin and badmin startup subcommands invoke the egosh *user logon* command to prompt for the Admin EGO cluster administrator credentials.

LSF_EGO_ADMIN_USER

Syntax

LSF_EGO_ADMIN_USER=Admin

Description

When the EGO Service Controller (EGOSC) is configured to control LSF daemons, enables UNIX and Windows users to bypass the additional login required to start res and sbatchd. Bypassing the EGO administrator login enables the use of scripts to automate system startup.

Specify the Admin EGO cluster administrator account. You must also define the LSF_EGO_ADMIN_PASSWD parameter.

Default

Not defined. With EGOSC daemon control enabled, the `lsadmin` and `badmin` startup subcommands invoke the `egosh` *user logon* command to prompt for the Admin EGO cluster administrator credentials.

LSF_LOAD_PLUGINS

Syntax

LSF_LOAD_PLUGINS=y | Y

Description

If defined, LSF loads plugins from LSB_LSBDIR. Used for Kerberos authentication and to enable the LSF cpuset plugin for IRIX.

Default

Not defined. LSF does not load plugins.

LSF_STARTUP_PATH

Syntax

LSF_STARTUP_PATH=*path*

Description

UNIX only. Enables the LSF daemon startup control feature when LSF_STARTUP_USERS is also defined. Define both parameters when you want to allow users other than root to start LSF daemons.

Specifies the absolute path name of the directory in which the LSF daemon binary files (lim, res, sbatchd, and mbatchd) are installed. LSF daemons are usually installed in the path specified by LSF_SERVERDIR defined in the `cshrc.lsf`, `profile.lsf` or `lsf.conf` files.

> **Important:**
>
> For security reasons, you should move the LSF daemon binary files to a directory other than LSF_SERVERDIR or LSF_BINDIR. The user accounts specified by LSF_STARTUP_USERS can start any binary in the LSF_STARTUP_PATH.

Default

Not defined. Only the root user account can start LSF daemons.

LSF_STARTUP_USERS

Syntax

LSF_STARTUP_USERS=all_admins | "*user_name...*"

Description

UNIX only. Enables the LSF daemon startup control feature when LSF_STARTUP_PATH is also defined. Define both parameters when you want to allow users other than root to start

LSF daemons. On Windows, the Platform services admin group is equivalent to LSF_STARTUP_USERS.

On UNIX hosts, by default only root can start LSF daemons. To manually start LSF daemons, a user runs the commands `lsadmin` and `badmin`, which have been installed as setuid root. LSF_STARTUP_USERS specifies a list of user accounts that can successfully run the commands `lsadmin` and `badmin` to start LSF daemons.

all_admins

- Allows all UNIX users defined as LSF administrators in the file `lsf.cluster.`*cluster_name* to start LSF daemons as root by running the `lsadmin` and `badmin` commands.
- Not recommended due to the security risk of a non-root LSF administrator adding to the list of administrators in the `lsf.cluster.`*cluster_name* file.
- Not required for Windows hosts because all users with membership in the Platform services admin group can start LSF daemons.

"*user_name*..."

- Allows the specified user accounts to start LSF daemons by running the `lsadmin` and `badmin` commands.
- Separate multiple user names with a space.
- For a single user, do not use quotation marks.

Default

Not defined. Only the root user account can start LSF daemons.

See also

LSF_STARTUP_PATH

lsf.task

Users should not have to specify a resource requirement each time they submit a job. LSF supports the concept of a task list. This chapter describes the files used to configure task lists: `lsf.task`, `lsf.task.cluster_name`, and `.lsftask`.

Changing task list configuration

After making any changes to the task list files, run the following commands:

- `lsadmin reconfig` to reconfigure LIM
- `badmin reconfig` to reload the configuration files

About task lists

A task list is a list in LSF that keeps track of the default resource requirements for different applications and task eligibility for remote execution.

The term task refers to an application name. With a task list defined, LSF automatically supplies the resource requirement of the job whenever users submit a job unless one is explicitly specified at job submission.

LSF takes the job's command name as the task name and uses that name to find the matching resource requirement for the job from the task list. If a task does not have an entry in the task list, LSF assumes the default resource requirement; that is, a host that has the same host type as the submission host will be chosen to run the job.

An application listed in a task file is considered for load sharing by its placement in either the local tasks or remote tasks list.

- A local task is typically an application or command that it does not make sense to run remotely such as `ls`.
- A remote task is an application or command that can be run on another machine in the LSF cluster. The `compress` command is an example of a remote task.

Some applications require resources other than the default. LSF can store resource requirements for specific applications in remote task list files, so that LSF automatically chooses candidate hosts that have the correct resources available.

For frequently used commands and software packages, the LSF administrator can set up cluster-wide resource requirements that apply to all users in the cluster.

Users can modify and add to these requirements by setting up additional resource requirements that apply only to their own jobs.

Cluster-wide resource requirements

The resource requirements of applications are stored in the remote task list file.

LSF automatically picks up a job's default resource requirement string from the remote task list files, unless you explicitly override the default by specifying the resource requirement string on the command line.

User-level resource requirements

You may have applications that you need to control yourself. Perhaps your administrator did not set them up for load sharing for all users, or you need a non-standard setup. You can use LSF commands to find out resource names available in your system, and tell LSF about the needs of your applications. LSF stores the resource requirements for you from then on.

You can specify resource requirements when tasks are added to the user's remote task list. If the task to be added is already in the list, its resource requirements are replaced.

lsrtasks + myjob/swap>=100 && cpu

This adds `myjob` to the remote tasks list with its resource requirements.

Task files

There are 3 task list files that can affect a job:

- `lsf.task` — system-wide defaults apply to all LSF users, even across multiple clusters if MultiCluster is installed
- `lsf.task.`*cluster_name* — cluster-wide defaults apply to all users in the cluster
- `$HOME/.lsftask` — user-level defaults apply to a single user. This file lists applications to be added to or removed from the default system lists for your jobs. Resource requirements specified in this file override those in the system lists.

The clusterwide task file is used to augment the systemwide file. The user's task file is used to augment the systemwide and clusterwide task files.

LSF combines the systemwide, clusterwide, and user-specific task lists for each user's view of the task list. In cases of conflicts, such as different resource requirements specified for the same task in different lists, the clusterwide list overrides the systemwide list, and the user-specific list overrides both.

LSF_CONFDIR/lsf.task

Systemwide task list applies to all clusters and all users.

This file is used in a MultiCluster environment.

LSF_CONFDIR/lsf.task.*cluster_name*

Clusterwide task list applies to all users in the same cluster.

$HOME/.lsftask

User task list, one per user, applies only to the specific user. This file is automatically created in the user's home directory whenever a user first updates his task lists using the `lsrtasks` or `lsltasks` commands. For details about task eligibility lists, see the `ls_task(3)` API reference man page.

Permissions

Only the LSF administrator can modify the systemwide task list (`lsf.task`) and the clusterwide task list (`lsf.task.`c*luster_name*).

A user can modify his own task list(`.lsftask`) with the `lsrtasks` and `lsltasks` commands.

Format of task files

Each file consists of two sections, Local Tasks and RemoteTasks. For example:

```
Begin LocalTasks
ps
hostname
uname
crontab
End LocalTasks
Begin RemoteTasks
+ "newjob/mem>25"
+ "verilog/select[type==any && swp>100]"
make/cpu
nroff/-
End RemoteTasks
```

Tasks are listed one per line. Each line in a section consists of a task name, and, for the RemoteTasks section, an optional resource requirement string separated by a slash (/).

A plus sign (+) or a minus sign (-) can optionally precede each entry. If no + or - is specified, + is assumed.

A + before a task name means adding a new entry (if non-existent) or replacing an entry (if already existent) in the task list. A - before a task name means removing an entry from the application's task lists if it was already created by reading higher level task files.

LocalTasks section

The section starts with Begin LocalTasks and ends with End LocalTasks.

This section lists tasks that are not eligible for remote execution, either because they are trivial tasks or because they need resources on the local host.

RemoteTasks section

The section starts with Begin RemoteTasks and ends with End RemoteTasks.

This section lists tasks that are eligible for remote execution. You can associate resource requirements with each task name.

See *Administering Platform LSF* for information about resource requirement strings. If the resource requirement string is not specified for a remote task, the default is "select[type==local] order[r15s:pg]".

setup.config

About setup.config

The setup.config file contains options for Platform LSF License Scheduler installation and configuration for systems without Platform LSF. You only need to edit this file if you are installing License Scheduler as a standalone product without LSF.

Template location

A template setup.config is included in the License Scheduler installation script tar file and is located in the directory created when you uncompress and extract the installation script tar file. Edit the file and uncomment the options you want in the template file. Replace the example values with your own settings to specify the options for your new License Scheduler installation.

Important:

The sample values in the setup.config template file are examples only. They are not default installation values.

After the License Scheduler installation, the setup.config containing the options you specified is located in LS_TOP/7.0/install/.

Format

Each entry in setup.config has the form:

```
NAME="STRING1 STRING2 ..."
```

The equal sign = must follow each NAME even if no value follows and there should be no spaces around the equal sign.

A value that contains multiple strings separated by spaces must be enclosed in quotation marks.

Blank lines and lines starting with a pound sign (#) are ignored.

Parameters

- LS_ADMIN
- LS_HOSTS
- LS_LICENSE_FILE
- LS_LMSTAT_PATH
- LS_TOP

LS_ADMIN

Syntax

LS_ADMIN="*user_name* [*user_name ...*]"

Description

Lists the License Scheduler administrators. The first user account name in the list is the primary License Scheduler administrator.

The primary License Scheduler administrator account is typically named lsadmin.

> **Caution:**
>
> You should *not* configure the root account as the primary License
> Scheduler administrator.

Valid Values

User accounts for License Scheduler administrators must exist on all hosts using License
Scheduler prior to installation.

Example

```
LS_ADMINS="lsadmin user1 user2"
```

Default

The user running the License Scheduler installation script.

LS_HOSTS

Syntax

```
LS_HOSTS="host_name [host_name ... ]"
```

Description

Defines a list of hosts that are candidates to become License Scheduler master hosts. Provide
at least one host from which the License Scheduler daemon will run.

Valid Values

Any valid License Scheduler host name.

Example

```
LS_HOSTS="host_name1 host_name2"
```

Default

The local host in which the License Scheduler installation script is running.

LS_LICENSE_FILE

Syntax

```
LS_LICENSE_FILE="/path/license_file"
```

Description

Defines the full path to, and name of the License Scheduler license file.

Valid Values

Any valid file name and directory path.

Example

```
LS_LICENSE_FILE="/usr/share/ls/conf/license.dat"
```

Default

$LS_TOP/conf/license.dat

LS_LMSTAT_PATH

Syntax

LS_LMSTAT_PATH="/*path*"

Description

Defines the full path to the lmstat program. License Scheduler uses lmstat to gather the FLEXlm license information for scheduling. This path does not include the name of the lmstat program itself.

Example

LS_LMSTAT_PATH="/usr/bin"

Default

The installation script attempts to find a working copy of lmstat on the current system. If it is unsuccessful, the path is set as blank ("").

LS_TOP

Syntax

LS_TOP="/*path*"

Description

Defines the full path to the top level License Shceduler installation directory.

Valid Values

Must be an absolute path to a shared directory that is accessible to all hosts using License Scheduler. Cannot be the root directory (/).

Recommended Value

The file system containing LS_TOP must have enough disk space for all host types (approximately 300 MB per host type).

Example

LS_TOP="/usr/share/ls"

Default

None — required variable

slave.config

About slave.config

Dynamically added LSF hosts that will not be master candidates are *slave hosts*. Each dynamic slave host has its own LSF binaries and local `lsf.conf` and shell environment scripts (`cshrc.lsf` and `profile.lsf`). You must install LSF on each slave host.

The `slave.config` file contains options for installing and configuring a slave host that can be dynamically added or removed.

Use `lsfinstall -s -f slave.config` to install LSF using the options specified in `slave.config`.

Template location

A template `slave.config` is located in the installation script directory created when you extract the LSF installation script tar file. Edit the file and uncomment the options you want in the template file. Replace the example values with your own settings to specify the options for your new LSF installation.

Important:

The sample values in the slave.config template file are examples only. They are not default installation values.

Format

Each entry in `slave.config` has the form:
```
NAME="STRING1 STRING2 ..."
```

The equal sign = must follow each `NAME` even if no value follows and there should be no spaces around the equal sign.

A value that contains multiple strings separated by spaces must be enclosed in quotation marks.

Blank lines and lines starting with a pound sign (#) are ignored.

Parameters

- EGO_DAEMON_CONTROL
- ENABLE_EGO
- EP_BACKUP
- LSF_ADMINS
- LSF_LIM_PORT
- LSF_SERVER_HOSTS
- LSF_TARDIR
- LSF_LOCAL_RESOURCES
- LSF_TOP

EGO_DAEMON_CONTROL

Syntax

EGO_DAEMON_CONTROL="Y" | **"N"**

Description

Enables Platform EGO to control LSF res and sbatchd. Set the value to "Y" if you want EGO Service Controller to start res and sbatchd, and restart if they fail.

All hosts in the cluster must use the same value for this parameter (this means the value of EGO_DAEMON_CONTROL in this file must be the same as the specification for EGO_DAEMON_CONTROL in install.config).

To avoid conflicts, leave this parameter undefined if you use a script to start up LSF daemons.

Note:

If you specify EGO_ENABLE="N", this parameter is ignored.

Example

EGO_DAEMON_CONTROL="N"

Default

N (res and sbatchd are started manually)

ENABLE_EGO

Syntax

ENABLE_EGO="Y" | **"N"**

Description

Enables Platform EGO functionality in the LSF cluster.

ENABLE_EGO="Y" causes lsfinstall uncomment LSF_EGO_ENVDIR and sets LSF_ENABLE_EGO="Y" in lsf.conf.

ENABLE_EGO="N" causes lsfinstall to comment out LSF_EGO_ENVDIR and sets LSF_ENABLE_EGO="N" in lsf.conf.

Set the value to "N" if you do not want to take advantage of the following LSF features that depend on EGO:

- LSF daemon control by EGO Service Controller
- EGO-enabled SLA scheduling
- Platform Management Console (PMC)
- LSF reporting

Default

Y (EGO is enabled in the LSF cluster)

EP_BACKUP

Syntax

EP_BACKUP="Y" | "N"

Description

Enables backup and rollback for enhancement packs. Set the value to "N" to disable backups when installing enhancement packs (you will not be able to roll back to the previous patch level after installing an EP, but you will still be able to roll back any fixes installed on the new EP).

You may disable backups to speed up install time, to save disk space, or because you have your own methods to back up the cluster.

Default

Y (backup and rollback are fully enabled)

LSF_ADMINS

Syntax

LSF_ADMINS="*user_name* [*user_name ...*]**"**

Description

Required. List of LSF administrators.

The first user account name in the list is the primary LSF administrator. It cannot be the root user account.

Typically this account is named lsfadmin. It owns the LSF configuration files and log files for job events. It also has permission to reconfigure LSF and to control batch jobs submitted by other users. It typically does not have authority to start LSF daemons. Usually, only root has permission to start LSF daemons.

All the LSF administrator accounts must exist on all hosts in the cluster before you install LSF. Secondary LSF administrators are optional.

Valid Values

Existing user accounts

Example

```
LSF_ADMINS="lsfadmin user1 user2"
```

Default

None—required variable

LSF_LIM_PORT

Syntax

LSF_LIM_PORT="*port_number***"**

Description

TCP service port for slave host.

Use the same port number as LSF_LIM_PORT in lsf.conf on the master host.

Default

7869

LSF_SERVER_HOSTS

Syntax

LSF_SERVER_HOSTS="*host_name* [*host_name ...*]**"**

Description

Required for non-shared slave host installation. This parameter defines a list of hosts that can provide host and load information to client hosts. If you do not define this parameter, clients will contact the master LIM for host and load information. List of LSF server hosts in the cluster to be contacted.

Recommended for large clusters to decrease the load on the master LIM. Do not specify the master host in the list. Client commands will query the LIMs on the LSF_SERVER_HOSTS, which off-loads traffic from the master LIM.

Define this parameter to ensure that commands execute successfully when no LIM is running on the local host, or when the local LIM has just started.

You should include the list of hosts defined in LSF_MASTER_LIST in lsf.conf; specify the primary master host last. For example:

```
LSF_MASTER_LIST="lsfmaster hostE"
```

```
LSF_SERVER_HOSTS="hostB hostC hostD hostE lsfmaster"
```

Specify a list of host names two ways:

- Host names separated by spaces
- Name of a file containing a list of host names, one host per line.

Valid Values

Any valid LSF host name

Examples

List of host names:

```
LSF_SERVER_HOSTS="hosta hostb hostc hostd"
```

Host list file:

```
LSF_SERVER_HOSTS=:lsf_server_hosts
```

The file lsf_server_hosts contains a list of hosts:

```
hosta hostb hostc hostd
```

Default

None

LSF_TARDIR

Syntax

LSF_TARDIR="*/path***"**

Description

Full path to the directory containing the LSF distribution tar files.

Example

```
LSF_TARDIR="/usr/local/lsf_distrib"
```

Default

The parent directory of the current working directory. For example, if `lsfinstall` is running under `usr/share/lsf_distrib/lsf_lsfinstall` the LSF_TARDIR default value is `usr/share/lsf_distrib`.

LSF_LOCAL_RESOURCES

Syntax

LSF_LOCAL_RESOURCES="*resource* ...**"**

Description

Defines instances of local resources residing on the slave host.

- For numeric resources, define name-value pairs:
  ```
  "[resourcemap value*resource_name]"
  ```
- For Boolean resources, define the resource name in the form:
  ```
  "[resource resource_name]"
  ```

When the slave host calls the master host to add itself, it also reports its local resources. The local resources to be added must be defined in `lsf.shared`.

If the same resource is already defined in `lsf.shared` as default or all, it cannot be added as a local resource. The shared resource overrides the local one.

Tip:

LSF_LOCAL_RESOURCES is usually set in the `slave.config` file during installation. If LSF_LOCAL_RESOURCES are already defined in a local `lsf.conf` on the slave host, `lsfinstall` does not add resources you define in LSF_LOCAL_RESOURCES in `slave.config`. You should not have duplicate LSF_LOCAL_RESOURCES entries in lsf.conf. If local resources are defined more than once, only the last definition is valid.

Important:

Resources must already be mapped to hosts in the ResourceMap section of lsf.cluster.*cluster_name*. If the ResourceMap section does not exist, local resources are not added.

Example

```
LSF_LOCAL_RESOURCES="[resourcemap 1*verilog] [resource linux]"
```

Default

None

LSF_TOP

Syntax

LSF_TOP="/*path***"**

Description

Required. Full path to the top-level LSF installation directory.

Important:

You must use the same path for every slave host you install.

Valid value

The path to LSF_TOP cannot be the root directory (/).

Example

```
LSF_TOP="/usr/local/lsf"
```

Default

None—required variable

slave.config

III

Environment Variables
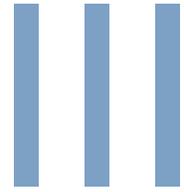
Environment Variables

Environment variables

Environment variables set for job execution

LSF transfers most environment variables between submission and execution hosts.

Environment variables related to file names and job spooling directories support paths that contain up to 4094 characters for UNIX and Linux, or up to 255 characters for Windows.

Environment variables related to command names and job names can contain up to 4094 characters for UNIX and Linux, or up to 255 characters for Windows.

In addition to environment variables inherited from the user environment, LSF also sets several other environment variables for batch jobs:

- LSB_ERRORFILE: Name of the error file specified with a `bsub -e`.
- LSB_JOBID: job ID assigned by LSF.
- LSB_JOBINDEX: Index of the job that belongs to a job array.
- LSB_CHKPNT_DIR: This variable is set each time a checkpointed job is submitted. The value of the variable is *chkpnt_dir/job_Id*, a subdirectory of the checkpoint directory that is specified when the job is submitted. The subdirectory is identified by the job ID of the submitted job.
- LSB_HOSTS: The list of hosts that are used to run the batch job. For sequential jobs, this is only one host name. For parallel jobs, this includes multiple host names.
- LSB_RESIZABLE: Indicates that a job is resizable or auto-resizable.
- LSB_QUEUE: The name of the queue the job is dispatched from.
- LSB_JOBNAME: Name of the job.
- LSB_RESTART: Set to 'Y' if the job is a restarted job or if the job has been migrated. Otherwise this variable is not defined.
- LSB_EXIT_PRE_ABORT: Set to an integer value representing an exit status. A pre-execution command should exit with this value if it wants the job to be aborted instead of requeued or executed.
- LSB_EXIT_REQUEUE: Set to the REQUEUE_EXIT_VALUES parameter of the queue. This variable is not defined if REQUEUE_EXIT_VALUES is not configured for the queue.
- LSB_INTERACTIVE: Set to 'Y' if the job is submitted with the -I option. Otherwise, it is not defined.
- 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.
- LSB_BIND_JOB: Set to the value of binding option. But when the binding option is USER, LSB_BIND_JOB is set to the real binding decision of end user.

  > **Note:**
  >
  > If the binding option is Y, LSB_BIND_JOB is set to BALANCE.
  > If the binding option is N, LSB_BIND_JOB is set to NONE.

- LSB_BIND_CPU_LIST: Set to the actual CPU list used when the job is sequential job and single host parallel job.

  If the job is a multi-host parallel job, LSB_BIND_CPU_LIST is set to the value in submission environment variable $LSB_USER_BIND_CPU_LIST. If there is no such

submission environment variable in user's environment, LSB_BIND_CPU_LIST is set to an empty string.

Environment variables for notification command

The following environment variables apply only to the notification command environment:

- LSB_RESIZE_NOTIFY_OK: A notification command should exit with this variable if the allocation resize notification command succeeds.

  LSF updates the job allocation to reflect the new allocation.
- LSB_RESIZE_NOTIFY_FAIL: A notification command should exit with this variable if the allocation resize notification command fails.

  For an allocation grow event, LSF schedules the pending allocation request.

  For an allocation shrink event, LSF fails the release request.
- LSB_RESIZE_EVENT = grow | shrink: Indicates why the notification command was called. Grow means add more resources to an existing allocation. Shrink means remove some resources from existing allocation.
- LSB_RESIZE_HOSTS = hostA numA hostB numB … hostZ numZ: Lists the additional slots for a grow event, or the released slots for a shrink event.

Environment variable reference

BSUB_BLOCK	BSUB_CHK_RESREQ	
BSUB_QUIET	BSUB_QUIET2	
BSUB_STDERR	CLEARCASE_DRIVE	CLEARCASE_MOUNTDIR
CLEARCASE_ROOT	ELIM_ABORT_VALUE	LM_LICENSE_FILE
LS_EXEC_T	LS_JOBPID	LS_LICENSE_SERVER_*feature*
LS_SUBCWD	LSB_CHKPNT_DIR	LSB_DEBUG
LSB_DEBUG_CMD	LSB_DEBUG_MBD	LSB_DEBUG_NQS
LSB_DEBUG_SBD	LSB_DEBUG_SCH	LSB_DEFAULT_JOBGROUP
LSB_DEFAULTPROJECT	LSB_DEFAULTQUEUE	LSB_DJOB_COMMFAIL_ACTION
LSB_DJOB_ENV_SCRIPT	LSB_ECHKPNT_METHOD	LSB_ECHKPNT_METHOD_DIR
LSB_ECHKPNT_KEEP_OUTPUT	LSB_ERESTART_USRCMD	LSB_EXEC_RUSAGE
LSB_EXECHOSTS	LSB_EXIT_IF_CWD_NOTEXIST	LSB_EXIT_PRE_ABORT
LSB_EXIT_REQUEUE	LSB_FRAMES	LSB_HOSTS
LSB_INTERACTIVE	LSB_JOB_INCLUDE_POSTPROC	LSB_JOBEXIT_INFO
LSB_JOBEXIT_STAT	LSB_JOBFILENAME	LSB_JOBGROUP
LSB_JOBID	LSB_JOBINDEX	LSB_JOBINDEX_STEP
LSB_JOBNAME	LSB_JOBPEND	LSB_JOBPGIDS

LSB_JOBPIDS	LSB_MAILSIZE	LSB_MCPU_HOSTS
LSB_NQS_PORT	LSB_NTRIES	LSB_OLD_JOBID
LSB_OUTPUT_TARGETFAILED	LSB_QUEUE	LSB_REMOTEINDEX
LSB_REMOTEJID	LSB_RESIZABLE	LSB_RESIZE_NOTIFY_OK
LSB_RESIZE_NOTIFY_FAIL	LSB_RESTART_PGID	LSB_RESTART
LSB_RESTART_PID	LSB_RTASK_GONE_ACTION	LSB_SUB_APP_NAME
LSB_SUB_CLUSTER	LSB_SUB_COMMAND_LINE	LSB_SUB_EXTSCHED_PARAM
LSB_SUB_JOB_ACTION_WARNING_TIME	LSB_SUB_JOB_WARNING_ACTION	LSB_SUB_PARM_FILE
LSB_SUCCESS_EXIT_VALUES	LSB_SUSP_REASONS	LSB_SUSP_SUBREASONS
LSB_UNIXGROUP	LSB_USER_BIND_CPU_LIST	LSB_USER_BIND_JOB
	LSF_CMD_LOGDIR	LSF_DEBUG_CMD
LSF_DEBUG_LIM	LSF_DEBUG_RES	LSF_EAUTH_AUX_DATA
LSF_EAUTH_AUX_PASS	LSF_EAUTH_CLIENT	LSF_EAUTH_SERVER
LSF_EAUTH_UID	LSF_EXECUTE_DOMAIN	LSF_INTERACTIVE_STDERR
LSF_INVOKE_CMD	LSF_JOB_STARTER	LSF_LD_PRELOAD
LSF_LD_LIBRARY_PATH	LSF_LIM_API_NTRIES	LSF_LIM_DEBUG
LSF_LOGDIR	LSF_MASTER	LSF_NIOS_DEBUG
LSF_NIOS_DIE_CMD	LSF_NIOS_IGNORE_SIGWINDOW	LSF_NIOS_PEND_TIMEOUT
LSF_NIOS_PORT_RANGE	LSF_RESOURCES	LSF_TS_LOGON_TIME
LSF_USE_HOSTEQUIV	LSF_USER_DOMAIN	

BSUB_BLOCK

Description

If set, tells NIOS that it is running in batch mode.

Default

Not defined

Notes

If you submit a job with the -K option of bsub, which is synchronous execution, then BSUB_BLOCK is set. Synchronous execution means you have to wait for the job to finish before you can continue.

Where defined

Set internally

See also

The -K option of bsub

BSUB_CHK_RESREQ

Syntax

BSUB_CHK_RESREQ=*any_value*

Description

When BSUB_CHK_RESREQ is set, bsub checks the syntax of the resource requirement selection string without actually submitting the job for scheduling and dispatch. Use BSUB_CHK_RESREQ to check the compatibility of your existing resource requirement select strings against the stricter syntax enabled by LSF_STRICT_RESREQ=y in lsf.conf. LSF_STRICT_RESREQ does not need to be set to check the resource requirement selection string syntax.

bsub only checks the select section of the resource requirement. Other sections in the resource requirement string are not checked.

Default

Not defined

Where defined

From the command line

Example

BSUB_CHK_RESREQ=1

BSUB_QUIET

Syntax

BSUB_QUIET=*any_value*

Description

Controls the printing of information about job submissions. If set, bsub will not print any information about job submission. For example, it will not print <Job is submitted to default queue <normal>, nor <Waiting for dispatch>.

Default

Not defined

Where defined

From the command line

Example

BSUB_QUI ET=1

BSUB_QUIET2

Syntax

BSUB_QUIET2=*any_value*

Description

Suppresses the printing of information about job completion when a job is submitted with the bsub -K option.

If set, bsub will not print information about job completion to stdout. For example, when this variable is set, the message <<Job is finished>> will not be written to stdout.

If BSUB_QUIET and BSUB_QUIET2 are both set, no job messages will be printed to stdout.

Default

Not defined

Where defined

From the command line

Example

BSUB_QUI ET2=1

BSUB_STDERR

Syntax

BSUB_STDERR=y

Description

Redirects LSF messages for bsub to stderr.

By default, when this parameter is not set, LSF messages for bsub are printed to stdout.

When this parameter is set, LSF messages for bsub are redirected to stderr.

Default

Not defined

Where defined

From the command line on UNIX. For example, in csh:

```
setenv BSUB_STDERR Y
```

From the Control Panel on Windows, as an environment variable

CLEARCASE_DRIVE

Syntax

CLEARCASE_DRIVE=_drive_letter_:

Description

Optional, Windows only.

Defines the virtual drive letter for a Rational ClearCase view to the drive. This is useful if you wish to map a Rational ClearCase view to a virtual drive as an alias.

If this letter is unavailable, Windows attempts to map to another drive. Therefore, CLEARCASE_DRIVE only defines the default drive letter to which the Rational ClearCase view is mapped, not the final selected drive letter. However, the PATH value is automatically updated to the final drive letter if it is different from CLEARCASE_DRIVE.

Notes:

CLEARCASE_DRIVE is case insensitive.

Where defined

From the command line

Example

```
CLEARCASE_DRI VE=F:
```
```
CLEARCASE_DRI VE=f:
```

See also

CLEARCASE_MOUNTDIR, CLEARCASE_ROOT

CLEARCASE_MOUNTDIR

Syntax

CLEARCASE_MOUNTDIR=_path_

Description

Optional.

Defines the Rational ClearCase mounting directory.

Default

```
/vobs
```

Notes:

CLEARCASE_MOUNTDIR is used if any of the following conditions apply:

- A job is submitted from a UNIX environment but run in a Windows host.
- The Rational ClearCase mounting directory is not the default /vobs

Where defined

From the command line

Example

```
CLEARCASE_MOUNTDIR=/myvobs
```

See also

CLEARCASE_DRIVE, CLEARCASE_ROOT

CLEARCASE_ROOT

Syntax

CLEARCASE_ROOT=*path*

Description

The path to the Rational ClearCase view.

In Windows, this path must define an absolute path starting with the default ClearCase drive and ending with the view name without an ending backslash (\).

Notes

CLEARCASE_ROOT must be defined if you want to submit a batch job from a ClearCase view.

For interactive jobs, use bsub -I to submit the job.

Where defined

In the job starter, or from the command line

Example

In UNIX:
```
CLEARCASE_ROOT=/view/myview
```

In Windows:
```
CLEARCASE_ROOT=F:\myview
```

See also

CLEARCASE_DRIVE, CLEARCASE_MOUNTDIR, LSF_JOB_STARTER

ELIM_ABORT_VALUE

Syntax

ELIM_ABORT_VALUE

Description

Used when writing an elim executable to test whether the elim should run on a particular host. If the host does not have or share any of the resources listed in the environment variable LSF_RESOURCES, your elim should exit with $ELIM_ABORT_VALUE.

When the MELIM finds an elim that exited with ELIM_ABORT_VALUE, the MELIM marks the elim and does not restart it on that host.

Where defined

Set by the master elim (MELIM) on the host when the MELIM invokes the elim executable

LM_LICENSE_FILE

Syntax

LM_LICENSE_FILE=_file_name_

Description

The path to where the license file is found. The file name is the name of the license file.

Default

/usr/share/flexlm/licenses/license.dat

Notes

A FLEXlm variable read by the lmgrd daemon.

Where defined

From the command line

See also

LSF_LICENSE_FILE in lsf.conf

LS_EXEC_T

Syntax

LS_EXEC_T=START | **END** | **CHKPNT** | **JOB_CONTROLS**

Description

Indicates execution type for a job. LS_EXEC_T is set to:

- START or END for a job when the job begins executing or when it completes execution

- • CHKPNT when the job is checkpointed
- • JOB_CONTROLS when a control action is initiated

Where defined

Set by sbatchd during job execution

LS_JOBPID

Description

The process ID of the job.

Where defined

During job execution, sbatchd sets LS_JOBPID to be the same as the process ID assigned by the operating system.

LS_LICENSE_SERVER_*feature*

Syntax

LS_LICENSE_SERVER_*feature*="*domain:server:num_available* ..."

server is of the format *port@host*

Description

The license server information provided to the job. The purpose of this environment variable is to provide license server information to the job.

Where defined

During the license job execution, sbatchd sets LS_LICENSE_SERVER_*feature* to be the same as the license server information defined in the job's rusage string. This is only used by the job and logged in the mbatchd log file if DEBUG1 and LC_LICSCHED are defined in lsf.conf.

LS_SUBCWD

Description

The current working directory (cwd) of the submission host where the remote task command was executed.

The current working directory can be up to 4094 characters long for UNIX and Linux or up to 255 characters for Windows.

How set

1. LSF looks for the PWD environment variable. If it finds it, sets LS_SUBCWD to PWD.
2. If the PWD environment variable does not exist, LSF looks for the CWD environment variable. If it finds CWD, sets LS_SUBCWD to CWD.

3. If the CWD environment variable does not exist, LSF calls the `getwd()` system function to retrieve the current working directory path name. LSF sets LS_SUBCWD to the value that is returned.

Where defined

Set by sbatchd

LSB_CHKPNT_DIR

Syntax

LSB_CHKPNT_DIR=*checkpoint_dir*/*job_ID*

Description

The directory containing files related to the submitted checkpointable job.

Valid values

The value of `checkpoint_dir` is the directory you specified through the `-k` option of `bsub` when submitting the checkpointable job.

The value of `job_ID` is the job ID of the checkpointable job.

Where defined

Set by LSF, based on the directory you specified when submitting a checkpointable job with the `-k` option of `bsub`.

LSB_DEBUG

This parameter can be set from the command line or from `lsf.conf`. See LSB_DEBUG in `lsf.conf`.

LSB_DEBUG_CMD

This parameter can be set from the command line or from `lsf.conf`. See LSB_DEBUG_CMD in `lsf.conf`.

LSB_DEBUG_MBD

This parameter can be set from the command line with `badmin mbddebug` or from `lsf.conf`.

See LSB_DEBUG_MBD in `lsf.conf`.

LSB_DEBUG_NQS

This parameter can be set from the command line or from `lsf.conf`. See LSB_DEBUG_NQS in `lsf.conf`.

LSB_DEBUG_SBD

This parameter can be set from the command line with `badmin sbddebug` or from `lsf.conf`.

See LSB_DEBUG_SBD in lsf.conf.

LSB_DEBUG_SCH

This parameter can be set from the command line or from lsf.conf. See LSB_DEBUG_SCH in lsf.conf.

LSB_DEFAULT_JOBGROUP

Syntax

LSB_DEFAULT_JOBGROUP=*job_group_name*

Description

The name of the default job group.

When you submit a job to LSF without explicitly specifying a job group, LSF associates the job with the specified job group. LSB_DEFAULT_JOBGROUP overrrides the setting of DEFAULT_JOBGROUP in lsb.params. The bsub -g *job_group_name* option overrides both LSB_DEFAULT_JOBGROUP and DEFAULT_JOBGROUP.

If you submit a job without the -g option of bsub, but you defined LSB_DEFAULT_JOBGROUP, then the job belongs to the job group specified in LSB_DEFAULT_JOBGROUP.

Job group names must follow this format:

- Job group names must start with a slash character (/). For example, LSB_DEFAULT_JOBGROUP=/A/B/C is correct, but LSB_DEFAULT_JOBGROUP=A/B/C is not correct.
- Job group names cannot end with a slash character (/). For example, LSB_DEFAULT_JOBGROUP=/A/ is not correct.
- Job group names cannot contain more than one slash character (/) in a row. For example, job group names like LSB_DEFAULT_JOBGROUP=/A//B or LSB_DEFAULT_JOBGROUP=A////B are not correct.
- Job group names cannot contain spaces. For example, LSB_DEFAULT_JOBGROUP=/A/B C/D is not correct.
- Project names and user names used for macro substitution with %p and %u cannot start or end with slash character (/).
- Project names and user names used for macro substitution with %p and %u cannot contain spaces or more than one slash character (/) in a row.
- Project names or user names containing slash character (/) will create separate job groups. For example, if the project name is canada/projects, LSB_DEFAULT_JOBGROUP=/%p results in a job group hierarchy /canada/projects.

Where defined

From the command line

Example

```
LSB_DEFAULT_JOBGROUP=/canada/projects
```

Default

Not defined

See also

DEFAULT_JOBGROUP in lsb.params, the -g option of bsub

LSB_DEFAULTPROJECT

Syntax

LSB_DEFAULTPROJECT=*project_name*

Description

The name of the project to which resources consumed by a job will be charged.

Default

Not defined

Notes

If the LSF administrator defines a default project in the lsb.params configuration file, the system uses this as the default project. You can change the default project by setting LSB_DEFAULTPROJECT or by specifying a project name with the -P option of bsub.

If you submit a job without the -P option of bsub, but you defined LSB_DEFAULTPROJECT, then the job belongs to the project specified in LSB_DEFAULTPROJECT.

If you submit a job with the -P option of bsub, the job belongs to the project specified through the -P option.

Where defined

From the command line, or through the -P option of bsub

Example

LSB_DEFAULTPROJECT=engineering

See also

DEFAULT_PROJECT in lsb.params, the -P option of bsub

LSB_DEFAULTQUEUE

Syntax

LSB_DEFAULTQUEUE=*queue_name*

Description

Defines the default LSF queue.

Default

mbatchd decides which is the default queue. You can override the default by defining LSB_DEFAULTQUEUE.

Notes

If the LSF administrator defines a default queue in the lsb.params configuration file, then the system uses this as the default queue. Provided you have permission, you can change the default queue by setting LSB_DEFAULTQUEUE to a valid queue (see bqueues for a list of valid queues).

Where defined

From the command line

See also

DEFAULT_QUEUE in lsb.params

LSB_ECHKPNT_METHOD

This parameter can be set as an environment variable and/or in lsf.conf. See LSB_ECHKPNT_METHOD in lsf.conf.

LSB_ECHKPNT_METHOD_DIR

This parameter can be set as an environment variable and/or in lsf.conf. See LSB_ECHKPNT_METHOD_DIR in lsf.conf.

LSB_ECHKPNT_KEEP_OUTPUT

This parameter can be set as an environment variable and/or in lsf.conf. See LSB_ECHKPNT_KEEP_OUTPUT in lsf.conf.

LSB_ERESTART_USRCMD

Syntax

LSB_ERESTART_USRCMD=_command_

Description

Original command used to start the job.

This environment variable is set by erestart to pass the job's original start command to a custom erestart method erestart._method_name_. The value of this variable is extracted from the job file of the checkpointed job.

If a job starter is defined for the queue to which the job was submitted, the job starter is also included in LSB_ERESTART_USRCMD. For example, if the job starter is /bin/sh -c "% USRCMD" in lsb.queues, and the job name is myapp -d, LSB_ERESTART_USRCMD will be set to:

```
/bin/sh -c "myapp -d"
```

Where defined

Set by `erestart` as an environment variable before a job is restarted

See also

LSB_ECHKPNT_METHOD, `erestart`, `echkpnt`

LSB_EXEC_RUSAGE

Syntax

LSB_EXEC_RUSAGE="*resource_name1 resource_value1 resource_name2 resource_value2...***"**

Description

Indicates which `rusage` string is satisfied to permit the job to run. This environment variable is necessary because the `OR` (`||`) operator specifies alternative `rusage` strings for running jobs.

Valid values

`resource_value1`, `resource_value2`,... refer to the resource values on `resource_name1`, `resource_name2`,... respectively.

Default

Not defined

Where defined

Set by LSF after reserving a resource for the job.

LSB_EXECHOSTS

Description

A list of hosts on which a batch job will run.

Where defined

Set by `sbatchd`

Product

MultiCluster

LSB_EXIT_IF_CWD_NOTEXIST

Syntax

LSB_EXIT_IF_CWD_NOTEXIST=Y | y | N | n

Description

Indicates that the job will exit if the current working directory specified by `bsub -cwd` or `bmod -cwd` is not accessible on the execution host.

Default

Not defined

Where defined

From the command line

LSB_EXIT_PRE_ABORT

Description

The queue-level or job-level `pre_exec_command` can exit with this value if the job is to be aborted instead of being requeued or executed

Where defined

Set by sbatchd

See also

See PRE_EXEC in `lsb.queues`, or the -E option of `bsub`

LSB_EXIT_REQUEUE

Syntax

LSB_EXIT_REQUEUE="*exit_value1 exit_value2***..."**

Description

Contains a list of exit values found in the queue's REQUEUE_EXIT_VALUES parameter defined in `lsb.queues`.

Valid values

Any positive integers

Default

Not defined

Notes

If LSB_EXIT_REQUEUE is defined, a job will be requeued if it exits with one of the specified values.

LSB_EXIT_REQUEUE is not defined if the parameter REQUEUE_EXIT_VALUES is not defined.

Where defined

Set by the system based on the value of the parameter REQUEUE_EXIT_VALUES in lsb.queues

Example

```
LSB_EXIT_REQUEUE="7 31"
```

See also

REQUEUE_EXIT_VALUES in lsb.queues

LSB_FRAMES

Syntax

LSB_FRAMES=_start_number,end_number,step_

Description

Determines the number of frames to be processed by a frame job.

Valid values

The values of _start_number_, _end_number_, and _step_ are positive integers. Use commas to separate the values.

Default

Not defined

Notes

When the job is running, LSB_FRAMES will be set to the relative frames with the format LSB_FRAMES=_start_number,end_number,step_.

From the _start_number_, _end_number_, and _step_, the frame job can know how many frames it will process.

Where defined

Set by sbatchd

Example

```
LSB_FRAMES=10,20,1
```

LSB_HOSTS

Syntax

LSB_HOSTS="_host_name_..."

Description

A list of hosts selected by LSF to run the job.

Notes

If a job is run on a single processor, the system sets LSB_HOSTS to the name of the host used.

For parallel jobs, the system sets LSB_HOSTS to the names of all the hosts used.

Where defined

Set by sbatchd when the job is executed. LSB_HOSTS is set only when the list of host names is less than 4096 bytes.

See also

LSB_MCPU_HOSTS

LSB_INTERACTIVE

Syntax

LSB_INTERACTIVE=Y

Description

Indicates an interactive job. When you submit an interactive job using bsub -I, the system sets LSB_INTERACTIVE to Y.

Valid values

LSB_INTERACTIVE=Y (if the job is interactive)

Default

Not defined (if the job is not interactive)

Where defined

Set by sbatchd

LSB_JOB_INCLUDE_POSTPROC

Syntax

LSB_JOB_INCLUDE_POSTPROC=Y | y | N | n

Description

Enables the post-execution processing of the job to be included as part of the job.

LSB_JOB_INCLUDE_POSTPROC in the user environment overrides the value of JOB_INCLUDE_POSTPROC in lsb.params and lsb.applications.

Default

Not defined

Where defined

From the command line

LSB_JOBEXIT_INFO

Syntax

LSB_JOBEXIT_INFO="SIGNAL *signal_value signal_name*"

Description

Contains information about signal that caused a job to exit.

Applies to post-execution commands. Post-execution commands are set with POST_EXEC in lsb.queues.

When the post-execution command is run, the environment variable LSB_JOBEXIT_INFO is set if the job is signalled internally. If the job ends successfully, or the job is killed or signalled externally, LSB_JOBEXIT_INFO is not set.

Examples

```
LSB_JOBEXIT_INFO="SIGNAL -1 SIG_CHKPNT"  LSB_JOBEXIT_INFO="SIGNAL -14
SIG_TERM_USER"  LSB_JOBEXIT_INFO="SIGNAL -23 SIG_KILL_REQUEUE"
```

Default

Not defined

Where defined

Set by sbatchd

LSB_JOBEXIT_STAT

Syntax

LSB_JOBEXIT_STAT=*exit_status*

Description

Indicates a job's exit status.

Applies to post-execution commands. Post-execution commands are set with POST_EXEC in lsb.queues.

When the post-execution command is run, the environment variable LSB_JOBEXIT_STAT is set to the exit status of the job. Refer to the man page for the wait(2) command for the format of this exit status.

The post-execution command is also run if a job is requeued because the job's execution environment fails to be set up, or if the job exits with one of the queue's REQUEUE_EXIT_VALUES. The LSB_JOBPEND environment variable is set if the job is requeued. If the job's execution environment could not be set up, LSB_JOBEXIT_STAT is set to 0.

Valid values

Any positive integer

Where defined

Set by sbatchd

LSB_JOBFILENAME

Syntax

LSB_JOBFILENAME=*file_name*

Description

The path to the batch executable job file that invokes the batch job. The batch executable job file is a /bin/sh script on UNIX systems or a .BAT command script on Windows systems.

LSB_JOBGROUP

Syntax

LSB_JOBGROUP=*job_group_name*

Description

The name of the job group associated with the job. When a job is dispatched, if it belongs to a job group, the runtime variable LSB_JOBGROUP is defined as its group. For example, if a dispatched job belongs to job group /X, LSB_JOBGROUP=/X.

Where defined

Set during job execution based on bsub options or the default job group defined in DEFAULT_JOBGROUP in lsb.params and the LSB_DEFAULT_JOBGROUP environment variable.

Default

Not defined

LSB_JOBID

Syntax

LSB_JOBID=*job_ID*

Description

The job ID assigned by sbatchd. This is the ID of the job assigned by LSF, as shown by bjobs.

Valid values

Any positive integer

Where defined

Set by sbatchd, defined by mbatchd

See also

LSB_REMOTEJID

LSB_JOBINDEX

Syntax

LSB_JOBINDEX=*index*

Description

Contains the job array index.

Valid values

Any integer greater than zero but less than the maximum job array size.

Notes

LSB_JOBINDEX is set when each job array element is dispatched. Its value corresponds to the job array index. LSB_JOBINDEX is set for all jobs. For non-array jobs, LSB_JOBINDEX is set to zero (0).

Where defined

Set during job execution based on bsub options.

Example

You can use LSB_JOBINDEX in a shell script to select the job command to be performed based on the job array index.

For example:
```
if [$LSB_JOBINDEX -eq 1]; then cmd1 fi if [$LSB_JOBINDEX -eq 2]; then cmd2 fi
```

See also

LSB_JOBINDEX_STEP, LSB_REMOTEINDEX

LSB_JOBINDEX_STEP

Syntax

LSB_JOBINDEX_STEP=*step*

Description

Step at which single elements of the job array are defined.

Valid values

Any integer greater than zero but less than the maximum job array size

Default

1

Notes

LSB_JOBINDEX_STEP is set when a job array is dispatched. Its value corresponds to the step of the job array index. This variable is set only for job arrays.

Where defined

Set during job execution based on bsub options.

Example

The following is an example of an array where a step of 2 is used:

```
array[1-10:2] elements: 1 3 5 7 9
```

If this job array is dispatched, then LSB_JOBINDEX_STEP=2

See also

LSB_JOBINDEX

LSB_JOBNAME

Syntax

LSB_JOBNAME=_job_name_

Description

The name of the job defined by the user at submission time.

Default

The job's command line

Notes

The name of a job can be specified explicitly when you submit a job. The name does not have to be unique. If you do not specify a job name, the job name defaults to the actual batch command as specified on the bsub command line.

The job name can be up to 4094 characters long for UNIX and Linux or up to 255 characters for Windows.

Where defined

Set by sbatchd

Example

When you submit a job using the -J option of bsub, for example:

```
% bsub -J "myjob" job
```

sbatchd sets LSB_JOBNAME to the job name that you specified:

LSB_JOBNAME=myj ob

LSB_JOBPEND

Description

Set if the job is requeued.

Where defined

Set by sbatchd for POST_EXEC only

See also

LSB_JOBEXIT_STAT, REQUEUE_EXIT_VALUES, POST_EXEC

LSB_JOBPGIDS

Description

A list of the current process group IDs of the job.

Where defined

The process group IDs are assigned by the operating system, and LSB_JOBPGIDS is set by sbatchd.

See also

LSB_JOBPIDS

LSB_JOBPIDS

Description

A list of the current process IDs of the job.

Where defined

The process IDs are assigned by the operating system, and LSB_JOBPIDS is set by sbatchd.

See also

LSB_JOBPGIDS

LSB_MAILSIZE

Syntax

LSB_MAILSIZE=*value*

Description

Gives an estimate of the size of the batch job output when the output is sent by email. It is not necessary to configure LSB_MAILSIZE_LIMIT.

LSF sets LSB_MAILSIZE to the size in KB of the job output, allowing the custom mail program to intercept output that is larger than desired.

LSB_MAILSIZE is not recognized by the LSF default mail program. To prevent large job output files from interfering with your mail system, use LSB_MAILSIZE_LIMIT to explicitly set the maximum size in KB of the email containing the job information.

Valid values

A positive integer

If the output is being sent by email, LSB_MAILSIZE is set to the estimated mail size in kilobytes.

-1

If the output fails or cannot be read, LSB_MAILSIZE is set to -1 and the output is sent by email using LSB_MAILPROG if specified in lsf.conf.

Not defined

If you use the -o or -e options of bsub, the output is redirected to an output file. Because the output is not sent by email in this case, LSB_MAILSIZE is not used and LSB_MAILPROG is not called.

If the -N option is used with the -o option of bsub, LSB_MAILSIZE is not set.

Where defined

Set by sbatchd when the custom mail program specified by LSB_MAILPROG in lsf.conf is called.

LSB_MCPU_HOSTS

Syntax

LSB_MCPU_HOSTS="*host_nameA num_processors1 host_nameB num_processors2***..."**

Description

Contains a list of the hosts and the number of CPUs used to run a job.

Valid values

num_processors1, num_processors2,... refer to the number of CPUs used on host_nameA, host_nameB,..., respectively

Default

Not defined

Notes

The environment variables LSB_HOSTS and LSB_MCPU_HOSTS both contain the same information, but the information is presented in different formats. LSB_MCPU_HOSTS uses a shorter format than LSB_HOSTS. As a general rule, sbatchd sets both these variables. However, for some parallel jobs, LSB_HOSTS is not set.

For parallel jobs, several CPUs are used, and the length of LSB_HOSTS can become very long. sbatchd needs to spend a lot of time parsing the string. If the size of LSB_HOSTS exceeds 4096 bytes, LSB_HOSTS is ignored, and sbatchd sets only LSB_MCPU_HOSTS.

To verify the hosts and CPUs used for your dispatched job, check the value of LSB_HOSTS for single CPU jobs, and check the value of LSB_MCPU_HOSTS for parallel jobs.

Where defined

Set by sbatchd before starting a job on the execution host

Example

When the you submit a job with the -m and -n options of bsub, for example,

```
% bsub -m "hostA hostB" -n 6 job
```

sbatchd sets the environment variables LSB_HOSTS and LSB_MCPU_HOSTS as follows:

```
LSB_HOSTS= "hostA hostA hostA hostB hostB hostB"
LSB_MCPU_HOSTS="hostA 3 hostB 3"
```

Both variables are set in order to maintain compatibility with earlier versions.

See also

LSB_HOSTS

LSB_NQS_PORT

This parameter can be defined in lsf.conf or in the services database such as /etc/services.

See LSB_NUM_NIOS_CALLBACK_THREADS in lsf.conf for more details.

LSB_NTRIES

Syntax

LSB_NTRIES=_integer_

Description

The number of times that LSF libraries attempt to contact mbatchd or perform a concurrent jobs query.

For example, if this parameter is not defined, when you type bjobs, LSF keeps displaying "batch system not responding" if mbatchd cannot be contacted or if the number of pending jobs exceeds MAX_PEND_JOBS specified in lsb.params or lsb.users.

If this parameter is set to a value, LSF only attempts to contact mbatchd the defined number of times and then quits. LSF will wait for a period of time equal to SUB_TRY_INTERVAL specified in lsb.params before attempting to contact mbatchd again.

Valid values

Any positive integer

Default

INFINIT_INT (The default is to continue the attempts to contact mbatchd)

LSB_OLD_JOBID

Syntax

LSB_OLD_JOBID=*job_ID*

Description

The job ID of a job at the time it was checkpointed.

When a job is restarted, it is assigned a new job ID and LSB_JOBID is replaced with the new job ID. LSB_OLD_JOBID identifies the original ID of a job before it is restarted.

Valid values

Any positive integer

Where defined

Set by sbatchd, defined by mbatchd

See also

LSB_JOBID

LSB_OUTPUT_TARGETFAILED

Syntax

LSB_OUTPUT_TARGETFAILED=Y

Description

Indicates that LSF cannot access the output file specified for a job submitted the bsub - o option.

Valid values

Set to Y if the output file cannot be accessed; otherwise, it is not defined.

Where defined

Set by sbatchd during job execution

LSB_DJOB_COMMFAIL_ACTION

Syntax

LSB_DJOB_COMMFAIL_ACTION="KILL_TASKS"

Description

Defines the action LSF should take if it detects a communication failure with one or more remote parallel or distributed tasks. If defined, LSF will try to kill all the current tasks of a parallel or distributed job associated with the communication failure. If not defined, the job RES notifies the task RES to terminate all tasks, and shut down the entire job.

Default

Terminate all tasks, and shut down the entire job

Valid values

KILL_TASKS

Where defined

Set by the system based on the value of the parameter DJOB_COMMFAIL_ACTION in lsb.applications when running bsub -app for the specified application

See also

DJOB_COMMFAIL_ACTION in lsb.applications

LSB_DJOB_ENV_SCRIPT

Syntax

LSB_DJOB_ENV_SCRIPT=*script_name*

Description

Defines the name of a user-defined script for setting and cleaning up the parallel or distributed job environment. This script will be executed by LSF with the argument setup before launching a parallel or distributed job, and with argument cleanup after the parallel job is finished.

The script will run as the user, and will be part of the job.

If a full path is specified, LSF will use the path name for the execution. Otherwise, LSF will look for the executable from $LSF_BINDIR.

Where defined

Set by the system to the value of the parameter DJOB_ENV_SCRIPT in lsb.applications when running bsub -app for the specified application

See also

DJOB_ENV_SCRIPT in lsb.applications

LSB_QUEUE

Syntax

LSB_QUEUE=*queue_name*

Description

The name of the queue from which the job is dispatched.

Where defined

Set by sbatchd

LSB_REMOTEINDEX

Syntax

LSB_REMOTEINDEX=*index*

Description

The job array index of a remote MultiCluster job. LSB_REMOTEINDEX is set only if the job is an element of a job array.

Valid values

Any integer greater than zero, but less than the maximum job array size

Where defined

Set by sbatchd

See also

LSB_JOBINDEX, MAX_JOB_ARRAY_SIZE in lsb.params

LSB_REMOTEJID

Syntax

LSB_REMOTEJID=*job_ID*

Description

The job ID of a remote MultiCluster job.

Where defined

Set by sbatchd, defined by mbatchd

See also

LSB_JOBID

LSB_RESTART

Syntax

LSB_RESTART=Y

Description

Indicates that a job has been restarted or migrated.

Valid values

Set to Y if the job has been restarted or migrated; otherwise, it is not defined.

Notes

If a batch job is submitted with the -r option of bsub, and is restarted because of host failure, then LSB_RESTART is set to Y. If a checkpointable job is submitted with the -k option of bsub, then LSB_RESTART is set to Y when the job is restarted. If bmi g is used to migrate a job, then LSB_RESTART is set to Y when the migrated job is restarted.

If the job is not a restarted job, then LSB_RESTART is not set.

Where defined

Set by sbatchd during job execution

See also

LSB_RESTART_PGID, LSB_RESTART_PID

LSB_RESTART_PGID

Syntax

LSB_RESTART_PGID=_pgid_

Description

The process group ID of the checkpointed job when the job is restarted.

Notes

When a checkpointed job is restarted, the operating system assigns a new group process ID to the job. LSF sets LSB_RESTART_PGID to the new group process ID.

Where defined

Set during restart of a checkpointed job.

See also

LSB_RESTART_PID, LSB_RESTART

LSB_RESTART_PID

Syntax

LSB_RESTART_PID=_pid_

Description

The process ID of the checkpointed job when the job is restarted.

Notes

When a checkpointed job is restarted, the operating system assigns a new process ID to the job. LSF sets LSB_RESTART_PID to the new process ID.

Where defined

Defined during restart of a checkpointed job

See also

LSB_RESTART_PGID, LSB_RESTART

LSB_RTASK_GONE_ACTION

Syntax

LSB_RTASK_GONE_ACTION=*task_action* ...

Description

Defines the actions LSF should take if it detects that a remote task of a parallel job is gone. Where *task_action* is:

IGNORE_TASKCRASH

A remote task crashes. The job RES does nothing.

KILLJOB_TASKDONE

A remote task exits with zero value. The job RES notifies the task RES to terminate all tasks in the job.

KILLJOB_TASKEXIT

A remote task exits with non-zero value. The job RES notifies the task RES to terminate all tasks in the job.

Where defined

Set by the system based on the value of the parameter RTASK_GONE_ACTION in lsb.applications when running bsub -app for the specified application

See also

RTASK_GONE_ACTION in lsb.applications

LSB_SUB_APP_NAME

Description

Application profile name specified by bsub -app.

Where defined

Set by esub before a job is dispatched.

LSB_SUB_CLUSTER

Description

Name of submission cluster (MultiCluster only)

Where defined

Set on the submission environment and passed to the execution cluster environment. The parameter will ONLY be valid in Multi Cluster environment. For jobs on a local cluster, the parameter is not set when using any daemon wrappers such as job starter, post-, pre- or eexec scripts.

LSB_SUB_COMMAND_LINE

Description

The job command line.

The job command line can be up to 4094 characters long for UNIX and Linux or up to 255 characters for Windows.

Where defined

Set by esub before a job is submitted.

LSB_SUB_EXTSCHED_PARAM

Description

Value of external scheduling options specified by bsub -extsched, or queue-level MANDATORY_EXTSCHED or DEFAULT_EXTSCHED.

Where defined

Set by esub before a job is submitted.

LSB_SUB_JOB_ACTION_WARNING_TIME

Description

Value of job warning time period specified by bsub -wt.

Where defined

Set by esub before a job is submitted.

LSB_SUB_JOB_WARNING_ACTION

Description

Value of job warning action specified by bsub -wa.

Where defined

Set by esub before a job is submitted.

LSB_SUB_PARM_FILE

Syntax

LSB_SUB_PARM_FILE=*file_name*

Description

Points to a temporary file that LSF uses to store the bsub options entered in the command line. An esub reads this file at job submission and either accepts the values, changes the values, or rejects the job. Job submission options are stored as name-value pairs on separate lines in the format option_name=value. A typical use of this file is to control job submission options.

Where defined

Set by LSF on the submission host before running esub. Not defined when lsrun or lsgrun are used for interactive remote execution.

LSB_SUCCESS_EXIT_VALUES

Syntax

LSB_SUCCESS_EXIT_VALUES=[*exit_code ...*]

Description

Specifies the exit values that indicate successful execution for applications that successfully exit with non-zero values. Use spaces to separate multiple exit codes. exit_code should be the value between 0 and 255.

User-defined LSB_SUCCESS_EXIT_VALUES overrides application profile level specification of SUCCESS_EXIT_VALUES in lsb.applications.

LSB_SUSP_REASONS

Syntax

LSB_SUSP_REASONS=*integer*

Description

An integer representing suspend reasons. Suspend reasons are defined in lsbatch.h.

This parameter is set when a job goes to system-suspended (SSUSP) or user-suspended status (USUSP). It indicates the exact reason why the job was suspended.

To determine the exact reason, you can test the value of LSB_SUSP_REASONS against the symbols defined in lsbatch.h.

Where defined

Set during job execution

See also

LSB_SUSP_SUBREASONS

LSB_SUSP_SUBREASONS

Syntax

LSB_SUSP_SUBREASONS=*integer*

Description

An integer representing the load index that caused a job to be suspended.

When the suspending reason SUSP_LOAD_REASON (suspended by load) is set in LSB_SUSP_REASONS, LSB_SUSP_SUBREASONS set to one of the load index values defined in lsf.h.

Use LSB_SUSP_REASONS and LSB_SUSP_SUBREASONS together in you custom job control to determine the exact load threshold that caused a job to be suspended.

Load index values are defined in lsf.h.

Load Index	Value
R15S	0
R1M	1
R15M	2
UT	3
PG	4
IO	5
LS	6
IT	7
TMP	8
SWP	9
MEM	10

Default

Not defined

Where defined

Set during job execution

See also

LSB_SUSP_REASONS

LSB_UNIXGROUP

Description

Specifies the UNIX user group of the submitting user.

Notes

This variable is useful if you want pre- or post-execution processing to use the user group of the user who submitted the job, and not sys(1).

Where defined

Set during job execution

LSB_USER_BIND_CPU_LIST

The binding requested at job submission takes effect when LSF_BIND_JOB=USER_CPU_LIST in lsf.conf or BIND_JOB=USER_CPU_LIST in an application profile in lsb.applications. LSF makes sure that the value is in the correct format, but does not check that the value is valid for the execution hosts.

The correct format is a list which may contain multiple items, separated by comma, and ranges. For example: 0,5,7,9-11.

LSB_USER_BIND_JOB

The binding requested at job submission takes effect when LSF_BIND_JOB=USER in lsf.conf or BIND_JOB=USER in an application profile in lsb.applications. This value must be one of Y, N, NONE, BALANCE, PACK, or ANY. Any other value is treated as ANY.

LSF_CMD_LOGDIR

This parameter can be set from the command line or from lsf.conf.

See LSF_CMD_LOGDIR in lsf.conf.

LSF_DEBUG_CMD

This parameter can be set from the command line or from lsf.conf.

See LSB_DEBUG_MBD in lsf.conf.

LSF_DEBUG_LIM

This parameter can be set from the command line or from lsf.conf.

See LSF_DEBUG_LIM in lsf.conf.

LSF_DEBUG_RES

This parameter can be set from the command line or from lsf.conf.

See LSF_DEBUG_RES in lsf.conf.

LSF_EAUTH_AUX_DATA

Syntax

LSF_EAUTH_AUX_DATA=*path/file_name*

Description

Used in conjunction with LSF daemon authentication, specifies the full path to the temporary file on the local file system that stores auxiliary authentication information (such as credentials required by a remote host for use during job execution). Provides a way for eauth -c, mbatchd, and sbatchd to communicate the location of auxiliary authentication data. Set internally by the LSF libraries in the context of eauth.

For Kerberos authentication, used for forwarding credentials to the execution host.

LSF_EAUTH_AUX_PASS

Syntax

LSF_EAUTH_AUX_PASS=yes

Description

Enables forwarding of credentials from a submission host to an execution host when daemon authentication is enabled. LSF_EAUTH_AUX_PASS=yes indicates that a credential can be added to the execution context of a job. Set to yes by bsub during job submission or by bmod during job modification so that eauth -c can forward credentials.

LSF_EAUTH_CLIENT

Syntax

LSF_EAUTH_CLIENT=mbatchd | sbatchd | pam | res | user

Description

Used with LSF daemon authentication, specifies the LSF daemon, command, or user that invokes eauth -c. Used when writing a customized eauth executable to set the context for the call to eauth. Set internally by the LSF libraries or by the LSF daemon, command, or user calling eauth -c.

LSF_EAUTH_SERVER

Syntax

LSF_EAUTH_SERVER=mbatchd | sbatchd | pam | res

Description

Used with LSF daemon authentication, specifies the daemon that invokes eauth -s. Used when writing a customized eauth executable to set the context for the call to eauth. Set internally by the LSF libraries or by the LSF daemon calling eauth -s.

LSF_EAUTH_UID

Syntax

LSF_EAUTH_UID=*user_ID*

Description

Specifies the user account under which eauth -s runs. Set by the LSF daemon that executes eauth. Set by the LSF daemon that executes eauth.

LSF_EXECUTE_DOMAIN

Syntax

LSF_EXECUTE_DOMAIN=domain_name**setenv LSF_EXECUTE_DOMAIN** *domain_name*

Description

If UNIX/Windows user account mapping is enabled, specifies the preferred Windows execution domain for a job submitted by a UNIX user. The execution domain must be one of the domains listed in LSF_USER_DOMAIN.

LSF_EXECUTE_DOMAIN is defined in the user environment (.cshrc or .profile) or from the command line. Specify only one domain.

Use this parameter in conjunction with the bsub, lsrun, and lsgrun commands to bypass the order of the domains listed in LSF_USER_DOMAIN and run the job using the specified domain. If you do not have a Windows user account in the execution domain, LSF tries to run the job using one of the other domains defined by LSF_USER_DOMAIN. Once you submit a job with an execution domain defined, you cannot change the execution domain for that particular job.

LSF_INTERACTIVE_STDERR

This parameter can be defined in lsf.conf.

See LSF_INTERACTIVE_STDERR in lsf.conf for more details.

LSF_INVOKE_CMD

Syntax

LSF_INVOKE_CMD=invoking_command_name

Description

Indicates the name of the last LSF command that invoked an external executable (for example, esub or eexec).

External executables get called by different LSF commands, such as bsub, bmod, or lsrun.

Default

Not defined

Where defined

Set internally within by LSF

LSF_JOB_STARTER

Syntax

LSF_JOB_STARTER=_binary_

Description

Specifies an executable program that has the actual job as an argument.

Default

Not defined

Interactive Jobs

If you want to run an interactive job that requires some preliminary setup, LSF provides a job starter function at the command level. A command-level job starter allows you to specify an executable file that will run prior to the actual job, doing any necessary setup and running the job when the setup is complete.

If LSF_JOB_STARTER is properly defined, RES will invoke the job starter (rather than the job itself), supplying your commands as arguments.

Batch Jobs

A job starter can also be defined at the queue level using the JOB_STARTER parameter, although this can only be done by the LSF administrator.

Where defined

From the command line

Example: UNIX

The job starter is invoked from within a Bourne shell, making the command-line equivalent:

```
/bin/sh -c "$LSF_JOB_STARTER command [argument...]"
```

where `command [argument...]` are the command line arguments you specified in `lsrun`, `lsgrun`, or `ch`.

If you define LSF_JOB_STARTER as follows:

```
setenv LSF_JOB_STARTER "/bin/csh -c"
```

and run a simple C-shell job:

```
lsrun "'a.out; echo hi'"
```

The following will be invoked to correctly start the job:

```
/bin/sh -c "/bin/csh -c 'a.out; echo hi'"
```

Example: Windows

RES runs the job starter, passing it your commands as arguments:

```
LSF_JOB_STARTER command [argument...]
```

If you define LSF_JOB_STARTER as follows:

```
set LSF_JOB_STARTER=C:\cmd.exe /C
```

and run a simple DOS shell job:

```
C:\> lsrun dir /p
```

then the following will be invoked to correctly start the job:

```
C:\cmd.exe /C dir /p
```

See also

JOB_STARTER in lsb.queues

LSF_LD_LIBRARY_PATH

Description

When LSF_LD_SECURITY=Y in lsf.conf, contains the value of the LD_LIBRARY_PATH environment variable, which is removed from the job environment during job initialization to ensure enhanced security against users obtaining root privileges.
LSF_LD_LIBRARY_PATH allows the LD_LIBRARY_PATH environment variable to be put back before the job runs.

Where defined

For jobs submitted using bsub -Is or bsub -Ip only.

See also

LSF_LD_PRELOAD, LSF_LD_SECURITY in lsf.conf

LSF_LD_PRELOAD

Description

When LSF_LD_SECURITY=Y in lsf.conf, contains the value of the LD_PRELOAD evnironment variable, which is removed from the job environment during job initialization to ensure enhanced security against users obtaining root privileges. LSF_LD_PRELOAD allows the LD_PRELOAD environment variable to be put back before the job runs.

Where defined

For jobs submitted using bsub -Is or bsub -Ip only.

See also

LSF_LD_LIBRARY_PATH, LSF_LD_SECURITY in lsf.conf

LSF_LIM_API_NTRIES

Syntax

LSF_LIM_API_NTRIES=*integer*

Description

Defines the number of times LSF commands will retry to communicate with the LIM API when LIM is not available. LSF_LIM_API_NTRIES is ignored by LSF and EGO daemons and EGO commands. The LSF_LIM_API_NTRIES environment variable. overrides the value of LSF_LIM_API_NTRIES in lsf.conf.

Valid values

1 to 65535

Where defined

From the command line or from lsf.conf

Default

Not defined. If not defined in lsf.conf. LIM API exits without retrying.

LSF_LIM_DEBUG

This parameter can be set from the command line or from lsf.conf.

See LSF_LIM_DEBUG in lsf.conf.

LSF_LOGDIR

This parameter can be set from the command line or from lsf.conf.

See LSF_LOGDIR in lsf.conf.

LSF_MASTER

Description

Set by the LIM to identify the master host. The value is Y on the master host and N on all other hosts. An elim executable can use this parameter to check the host on which the elim is currently running.

Used when the external load indices feature is enabled.

When defined

Set by the LIM when it starts the master external load information manager (MELIM).

See also

LSF_RESOURCES

LSF_NIOS_DEBUG

This parameter can be set from the command line or from l sf. conf.

See LSF_NIOS_DEBUG in l sf. conf.

LSF_NIOS_DIE_CMD

Syntax

LSF_NIOS_DIE_CMD=*command*

Description

If set, the command defined by LSF_NIOS_DIE_CMD is executed before NIOS exits.

Default

Not defined

Where defined

From the command line

LSF_NIOS_IGNORE_SIGWINDOW

Syntax

LSF_NIOS_IGNORE_SIGWINDOW=*any_value*

Description

If defined, the NIOS will ignore the SIGWINDOW signal.

Default

Not defined

Notes

When the signal SIGWINDOW is defined, some tasks appear to die when they receive the SIGWINDOW while doing I/O. By defining LSF_NIOS_IGNORE_SIGWINDOW, these tasks are given the chance to ignore the signal.

Where defined

From the command line

LSF_NIOS_PEND_TIMEOUT

Syntax

LSF_NIOS_PEND_TIMEOUT=*minutes*

Description

Applies only to interactive batch jobs.

Maximum amount of time that an interactive batch job can remain pending.

If this parameter is defined, and an interactive batch job is pending for longer than the specified time, the interactive batch job is terminated.

Valid values

Any integer greater than zero

Default

Not defined

LSF_NIOS_PORT_RANGE

Syntax

LSF_NIOS_PORT_RANGE=_min_port_number-max_port_number_

Description

Defines a range of listening ports for NIOS to use.

Example

```
LSF_NIOS_PORT_RANGE=5000-6000
```

Default

Not defined. LSF randomly assigns a NIOS port number.

LSF_RESOURCES

Syntax

LSF_RESOURCES=_dynamic_external_resource_name..._

Description

Space-separated list of dynamic external resources. When the LIM starts a master external load information manager (MELIM) on a host, the LIM checks the resource mapping defined in the ResourceMap section of `lsf.cluster.`_cluster_name_. Based on the mapping (default, all, or a host list), the LIM sets LSF_RESOURCES to the list of resources expected on the host and passes the information to the MELIM.

Used when the external load indices feature is enabled.

When defined

Set by the MELIM on the host when the MELIM invokes the `elim` executable.

See also

LSF_MASTER

LSF_TS_LOGON_TIME

Syntax

LSF_TS_LOGON_TIME=*milliseconds*

Description

Specifies the time to create a Windows Terminal Service session. Configure LSF_TS_LOGON_TIME according to the load on your network enviroment.

The default, 30000 milliseconds, is suitable for most environments. If you set LSF_TS_LOGON_TIME too small, the LSF tries multiple times before it succeeds in making a TS session with the TS server, which can cause the job wait a long time before it runs. For a congested network. set LSF_TS_LOGON_TIME=1000000.

Where defined

From the command line

Default

30000 milliseconds

LSF_USE_HOSTEQUIV

Syntax

LSF_USE_HOSTEQUIV=y | Y

Description

Used for authentication purposes. If LSF_USE_HOSTEQUIV is defined, RES and mbatchd call the `ruserok`(3) function to decide if a user is allowed to run remote jobs. LSF trusts all hosts configured in the LSF cluster that are defined in `hosts.equiv`, or in `.rhosts` in the user's home directory.

The `ruserok`(3) function checks in the `/etc/hosts.equiv` file and the user's `$HOME/.rhosts` file to decide if the user has permission to execute remote jobs.

If LSF_USE_HOSTEQUIV is not defined, all normal users in the cluster can execute remote jobs on any host.

If LSF_ROOT_REX is set, root can also execute remote jobs with the same permission test as for normal users.

Default

Not defined

See also

LSF_ROOT_REX and LSF_AUTH in `lsf.conf`

LSF_USER_DOMAIN

Syntax

LSF_USER_DOMAIN=_domain_name_ | **.**

Description

Set during LSF installation or setup. If you modify this parameter in an existing cluster, you probably have to modify passwords and configuration files also.

Windows or mixed UNIX-Windows clusters only.

Enables default user mapping, and specifies the LSF user domain. The period (.) specifies local accounts, not domain accounts.

- A user name specified without a domain is interpreted (on a Windows host) as belonging to the LSF user domain
- A user name specified with the domain name of the LSF user domain is not valid
- In a mixed cluster, this parameter defines a 2-way, 1:1 user map between UNIX user accounts and Windows user accounts belonging to the specified domain, as long as the accounts have the same user name. This means jobs submitted by the Windows user account can run on a UNIX host, and jobs submitted by the UNIX account can run on any Windows host that is available to the Windows user account.

If this parameter is not defined, the default user mapping is not enabled. You can still configure user mapping at the user or system level. User account mapping is required to run cross-platform jobs in a UNIX-Windows mixed cluster.

Where defined

lsf.conf

Default

- If you upgrade from LSF 4.0.1 or earlier, the default is the existing LSF user domain.
- For a new, Windows-only cluster, this parameter is not defined (no LSF user domain, no default user mapping).
- For a new, mixed UNIX-Windows cluster, the default is the domain that the Windows installation account belongs to. This can be modified during LSF installation.

IV

Troubleshooting

Troubleshooting

Troubleshooting and error messages

Shared file access

A frequent problem is non-accessible files due to a non-uniform file space. If a task is run on a remote host where a file it requires cannot be accessed using the same name, an error results. Almost all interactive LSF commands fail if the user's current working directory cannot be found on the remote host.

Shared files on UNIX

If you are running NFS, rearranging the NFS mount table may solve the problem. If your system is running the automount server, LSF tries to map the filenames, and in most cases it succeeds. If shared mounts are used, the mapping may break for those files. In such cases, specific measures need to be taken to get around it.

The automount maps must be managed through NIS. When LSF tries to map filenames, it assumes that automounted file systems are mounted under the /tmp_mnt directory.

Shared files on Microsoft Windows

To share files among Windows machines, set up a share on the server and access it from the client. You can access files on the share either by specifying a UNC path (\\server\share\path) or connecting the share to a local drive name and using a drive:\path syntax. Using UNC is recommended because drive mappings may be different across machines, while UNC allows you to unambiguously refer to a file on the network.

Shared files across UNIX and Windows

For file sharing across UNIX and Windows, you require a third party NFS product on Windows to export directories from Windows to UNIX.

Common LSF problems

This section lists some common problems with LSF jobs. Most problems are due to incorrect installation or configuration. Check the mbatchd and sbatchd error log files; often the log message points directly to the problem.

The section also includes some common problems with the LIM, the RES and interactive applications.

LIM dies quietly

Run the following command to check for errors in the LIM configuration files.

lsadmin ckconfig -v

This displays most configuration errors. If this does not report any errors, check in the LIM error log.

LIM unavailable

Sometimes the LIM is up, but executing the lsload command prints the following error message:

```
Communication time out.
```

If the LIM has just been started, this is normal, because the LIM needs time to get initialized by reading configuration files and contacting other LIMs. If the LIM does not become available within one or two minutes, check the LIM error log for the host you are working on.

To prevent communication timeouts when starting or restarting the local LIM, define the parameter LSF_SERVER_HOSTS in the `lsf.conf` file. The client will contact the LIM on one of the LSF_SERVER_HOSTS and execute the command, provided that at least one of the hosts defined in the list has a LIM that is up and running.

When the local LIM is running but there is no master LIM in the cluster, LSF applications display the following message:

```
Cannot locate master LIM now, try later.
```

Check the LIM error logs on the first few hosts listed in the `Host` section of the `lsf.cluster.`*cluster_name* file. If LSF_MASTER_LIST is defined in `lsf.conf`, check the LIM error logs on the hosts listed in this parameter instead.

Master LIM is down

Sometimes the master LIM is up, but executing the `lsload` or `lshosts` command prints the following error message:

```
Master LIM is down; try later
```

If the `/etc/hosts` file on the host where the master LIM is running is configured with the host name assigned to the loopback IP address (127.0.0.1), LSF client LIMs cannot contact the master LIM. When the master LIM starts up, it sets its official host name and IP address to the loopback address. Any client requests will get the master LIM address as 127.0.0.1, and try to connect to it, and in fact will try to access itself.

Check the IP configuration of your master LIM in `/etc/hosts`. The following IPv4 example incorrectly sets the master LIM IP address to the loopback address:

```
127.0.0.1          localhost        myhostname
```

The following example correctly sets the master LIM IP address:

```
127.0.0.1          localhost
192.168.123.123    myhostname
```

For a master LIM running on a host that uses an IPv6 address, the loopback address is

```
::1
```

An IPv6-enabled host should have the following entries in its `/etc/hosts` file:

```
::1          localhost ipv6-localhost ipv6-loopback
fe00::0        ipv6-localnet
ff00::0        ipv6-mcastprefix
ff02::1        ipv6-allnodes
ff02::2        ipv6-allrouters
ff02::3        ipv6-allhosts
```

RES does not start

Check the RES error log.

On 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`.

On Windows, if the RES is unable to read the `lsf.conf` file and does not know where to write error messages, it logs errors into `C:\temp`.

User permission denied

If remote execution fails with the following error message, the remote host could not securely determine the user ID of the user requesting remote execution.

```
User permission denied.
```

Check the RES error log on the remote host; this usually contains a more detailed error message.

If you are not using an identification daemon (LSF_AUTH is not defined in the lsf.conf file), then all applications that do remote executions must be owned by root with the setuid bit set. This can be done as follows.

chmod 4755 filename

If the binaries are on an NFS-mounted file system, make sure that the file system is not mounted with the nosuid flag.

If you are using an identification daemon (defined in the lsf.conf file by LSF_AUTH), inetd must be configured to run the daemon. The identification daemon must not be run directly.

If LSF_USE_HOSTEQUIV=Y is defined in the lsf.conf file, check if /etc/hosts.equiv or HOME/.rhosts on the destination host has the client host name in it. Inconsistent host names in a name server with /etc/hosts and /etc/hosts.equiv can also cause this problem.

On SGI hosts running a name server, you can try the following command to tell the host name lookup code to search the /etc/hosts file before calling the name server.

setenv HOSTRESORDER "local,nis,bind"

For Windows hosts, users must register and update their Windows passwords using the lspasswd command. Passwords must be 3 characters or longer and 31 characters or less.

For Windows password authentication in a non-shared file system environment, you must define the parameter LSF_MASTER_LIST in lsf.conf so that jobs will run with correct permissions. If you do not define this parameter, LSF assumes that the cluster uses a shared file system environment.

Non-uniform file name space

A command may fail with the following error message due to a non-uniform file name space.

```
chdir(...) failed: no such file or directory
```

You are trying to execute a command remotely, where either your current working directory does not exist on the remote host, or your current working directory is mapped to a different name on the remote host.

If your current working directory does not exist on a remote host, you should not execute commands remotely on that host.

On UNIX, if the directory exists, but is mapped to a different name on the remote host, you have to create symbolic links to make them consistent.

LSF can resolve most, but not all, problems using automount. The automount maps must be managed through NIS. Follow the instructions in your Release Notes for obtaining technical

support if you are running automount and LSF is not able to locate directories on remote hosts.

Batch daemons die quietly

First, check the sbatchd and mbatchd error logs. Try running the following command to check the configuration.

badmin ckconfig

This reports most errors. You should also check if there is any email from LSF in the LSF administrator's mailbox. If the mbatchd is running but the sbatchd dies on some hosts, it may be because mbatchd has not been configured to use those hosts.

sbatchd starts but mbatchd does not

Check whether LIM is running. You can test this by running the lsid command. If LIM is not running properly, follow the suggestions in this chapter to fix the LIM first. You should make sure that all hosts use the same lsf.conf file. Note that it is possible that mbatchd is temporarily unavailable because the master LIM is temporarily unknown, causing the following error message.

```
sbatchd: unknown service
```

Check whether services are registered properly. See *Administering Platform LSF* for information about registering LSF services.

Host not used by LSF

If you configure a list of server hosts in the Host section of the lsb.hosts file, mbatchd allows sbatchd to run only on the hosts listed. If you try to configure an unknown host as a HOSTS definition for a queue in the lsb.queues file, mbatchd logs the following message.

```
mbatchd on host: LSB_CONFDIR/cluster/configdir/file(line #): Host
hostname is not used by lsbatch; ignored
```

If you try to configure an unknown host in the HostGroup or HostPartition sections of the lsb.hosts file, you also see the message.

If you start sbatchd on a host that is not known by mbatchd, mbatchd rejects the sbatchd. The sbatchd logs the following message and exits.

```
This host is not used by lsbatch system.
```

Both of these errors are most often caused by not running the following commands, in order, after adding a host to the configuration.

```
lsadmin reconfig badmin reconfig
```

You must run both of these before starting the daemons on the new host.

Data limit error on AIX

On AIX, if the XPG_SUS_ENV=ON environment variable is set in the user's environment before the process is executed and a process attempts to set the limit lower than current usage, the operation fails with errno set to EINVAL. If the XPG_SUS_ENV environment variable is not set, the operation fails with errno set to EFAULT.

Error messages

The following error messages are logged by the LSF daemons, or displayed by the following commands.

```
lsadmin ckconfig badmin ckconfig
```

General errors

The messages listed in this section may be generated by any LSF daemon.

```
can't open file: error
```

The daemon could not open the named file for the reason given by *error*. This error is usually caused by incorrect file permissions or missing files. All directories in the path to the configuration files must have execute (x) permission for the LSF administrator, and the actual files must have read (r) permission. Missing files could be caused by incorrect path names in the lsf.conf file, running LSF daemons on a host where the configuration files have not been installed, or having a symbolic link pointing to a nonexistent file or directory.

```
file(line): malloc failed
```

Memory allocation failed. Either the host does not have enough available memory or swap space, or there is an internal error in the daemon. Check the program load and available swap space on the host; if the swap space is full, you must add more swap space or run fewer (or smaller) programs on that host.

```
auth_user: getservbyname(ident/tcp) failed: error; ident must be
registered in services
```

LSF_AUTH=ident is defined in the lsf.conf file, but the ident/tcp service is not defined in the services database. Add ident/tcp to the services database, or remove LSF_AUTH from the lsf.conf file and setuid root those LSF binaries that require authentication.

```
auth_user: operation(<host>/<port>) failed: error
```

LSF_AUTH=ident is defined in the lsf.conf file, but the LSF daemon failed to contact the identd daemon on host. Check that identd is defined in inetd.conf and the identd daemon is running on host.

```
auth_user: Authentication data format error (rbuf=<data>) from
<host>/<port>
```

```
auth_user: Authentication port mismatch (...) from <host>/<port>
```

LSF_AUTH=ident is defined in the lsf.conf file, but there is a protocol error between LSF and the ident daemon on *host*. Make sure the ident daemon on the host is configured correctly.

```
userok: Request from bad port (<port_number>), denied
```

LSF_AUTH is not defined, and the LSF daemon received a request that originates from a non-privileged port. The request is not serviced.

Set the LSF binaries (for example, lsrun) to be owned by root with the setuid bit set, or define LSF_AUTH=ident and set up an ident server on all hosts in the cluster. If the binaries are on an NFS-mounted file system, make sure that the file system is not mounted with the nosuid flag.

```
userok: Forged username suspected from <host>/<port>: <claimed_user>/
<actual_user>
```

The service request claimed to come from user *claimed_user* but ident authentication returned that the user was actually *actual_user*. The request was not serviced.

```
userok: ruserok(<host>,<uid>) failed
```

LSF_USE_HOSTEQUIV=Y is defined in the `lsf.conf` file, but *host* has not been set up as an equivalent host (see `/etc/host.equiv`), and user *uid* has not set up a `.rhosts` file.

```
init_AcceptSock: RES service(res) not registered, exiting
```

```
init_AcceptSock: res/tcp: unknown service, exiting initSock: LIM
service not registered.
```

```
initSock: Service lim/udp is unknown. Read LSF Guide for help
get_ports: <serv> service not registered
```

The LSF services are not registered. See *Administering Platform LSF* for information about configuring LSF services.

```
init_AcceptSock: Can't bind daemon socket to port <port>: error,
exiting
```

```
init_ServSock: Could not bind socket to port <port>: error
```

These error messages can occur if you try to start a second LSF daemon (for example, RES is already running, and you execute RES again). If this is the case, and you want to start the new daemon, kill the running daemon or use the `lsadmin` or `badmin` commands to shut down or restart the daemon.

Configuration errors

The messages listed in this section are caused by problems in the LSF configuration files. General errors are listed first, and then errors from specific files.

```
file(line): Section name expected after Begin; ignoring section file
(line): Invalid section name name; ignoring section
```

The keyword begin at the specified line is not followed by a section name, or is followed by an unrecognized section name.

```
file(line): section section: Premature EOF
```

The end of file was reached before reading the end section line for the named section.

```
file(line): keyword line format error for section section; Ignore
this section
```

The first line of the section should contain a list of keywords. This error is printed when the keyword line is incorrect or contains an unrecognized keyword.

```
file(line): values do not match keys for section section; Ignoring
line
```

The number of fields on a line in a configuration section does not match the number of keywords. This may be caused by not putting `()` in a column to represent the default value.

```
file: HostModel section missing or invalid
```

```
file: Resource section missing or invalid
```

```
file: HostType section missing or invalid
```

The HostModel, Resource, or HostType section in the `lsf.shared` file is either missing or contains an unrecoverable error.

`file(line): Name name reserved or previously defined. Ignoring index`

The name assigned to an external load index must not be the same as any built-in or previously defined resource or load index.

`file(line): Duplicate clustername name in section cluster. Ignoring current line`

A cluster name is defined twice in the same `lsf.shared` file. The second definition is ignored.

`file(line): Bad cpuFactor for host model model. Ignoring line`

The CPU factor declared for the named host model in the `lsf.shared` file is not a valid number.

`file(line): Too many host models, ignoring model name`

You can declare a maximum of 127 host models in the `lsf.shared` file.

`file(line): Resource name name too long in section resource. Should be less than 40 characters. Ignoring line`

The maximum length of a resource name is 39 characters. Choose a shorter name for the resource.

`file(line): Resource name name reserved or previously defined. Ignoring line.`

You have attempted to define a resource name that is reserved by LSF or already defined in the `lsf.shared` file. Choose another name for the resource.

`file(line): illegal character in resource name: name, section resource. Line ignored.`

Resource names must begin with a letter in the set [a-zA-Z], followed by letters, digits or underscores [a-zA-Z0-9_].

LIM messages

The following messages are logged by the LIM:

`main: LIM cannot run without licenses, exiting`

The LSF software license key is not found or has expired. Check that FLEXlm is set up correctly, or contact Platform support at support@platform.com.

`main: Received request from unlicensed host <host>/<port>`

LIM refuses to service requests from hosts that do not have licenses. Either your LSF license has expired, or you have configured LSF on more hosts than your license key allows.

`initLicense: Trying to get license for LIM from source <LSF_CONFDIR/license.dat>`

`getLicense: Can't get software license for LIM from license file <LSF_CONFDIR/license.dat>: feature not yet available.`

Your LSF license is not yet valid. Check whether the system clock is correct.

`findHostbyAddr/<proc>: Host <host>/<port> is unknown by <myhostname>`

```
function: Gethostbyaddr_(<host>/<port>) failed: error
```

```
main: Request from unknown host <host>/<port>: error
```

```
function: Received request from non-LSF host <host>/<port>
```

The daemon does not recognize *host* as a Platform LSF host. The request is not serviced. These messages can occur if *host* was added to the configuration files, but not all the daemons have been reconfigured to read the new information. If the problem still occurs after reconfiguring all the daemons, check whether the host is a multi-addressed host. See *Administering Platform LSF* for information about working with multi-addressed hosts.

```
rcvLoadVector: Sender (<host>/<port>) may have different config?
MasterRegister: Sender (host) may have different config?
```

LIM detected inconsistent configuration information with the sending LIM. Run the following command so that all the LIMs have the same configuration information.

lsadmin reconfig

Note any hosts that failed to be contacted.

```
rcvLoadVector: Got load from client-only host <host>/<port>. Kill LIM
on <host>/<port>
```

A LIM is running on a Platform LSF client host. Run the following command, or go to the client host and kill the LIM daemon.

lsadmin limshutdown host

```
saveIndx: Unknown index name <name> from ELIM
```

LIM received an external load index name that is not defined in the lsf.shared file. If name is defined in lsf.shared, reconfigure the LIM. Otherwise, add name to the lsf.shared file and reconfigure all the LIMs.

```
saveIndx: ELIM over-riding value of index <name>
```

This is a warning message. The ELIM sent a value for one of the built-in index names. LIM uses the value from ELIM in place of the value obtained from the kernel.

```
getusr: Protocol error numIndx not read (cc=num): error
```

```
getusr: Protocol error on index number (cc=num): error
```

Protocol error between ELIM and LIM. See *Administering Platform LSF* for a description of the ELIM and LIM protocols.

RES messages

These messages are logged by the RES.

```
doacceptconn: getpwnam(<username>@<host>/<port>) failed: error
doacceptconn: User <username> has uid <uid1> on client host <host>/
<port>, uid <uid2> on RES host; assume bad user authRequest: username/
uid <userName>/<uid>@<host>/<port> does not exist
```

```
authRequest: Submitter's name <clname>@<clhost> is different from
name <lname> on this host
```

RES assumes that a user has the same userID and username on all the LSF hosts. These messages occur if this assumption is violated. If the user is allowed to use LSF for interactive remote execution, make sure the user's account has the same user ID and user name on all LSF hosts.

`doacceptconn: root remote execution permission denied authRequest: root job submission rejected`

Root tried to execute or submit a job but LSF_ROOT_REX is not defined in the `lsf.conf` file.

`resControl: operation permission denied, uid = <uid>`

The user with user ID *uid* is not allowed to make RES control requests. Only the LSF administrator, or root if LSF_ROOT_REX is defined in `lsf.conf`, can make RES control requests.

`resControl: access(respath, X_OK): error`

The RES received a reboot request, but failed to find the file `respath` to re-execute itself. Make sure `respath` contains the RES binary, and it has execution permission.

LSF messages

The following messages are logged by the mbatchd and sbatchd daemons:

`renewJob: Job <jobId>: rename(<from>,<to>) failed: error`

mbatchd failed in trying to re-submit a rerunnable job. Check that the file *from* exists and that the LSF administrator can rename the file. If *from* is in an AFS directory, check that the LSF administrator's token processing is properly setup

See *Administering Platform LSF* for information about installing on AFS.

`logJobInfo_: fopen(<logdir/info/jobfile>) failed: error`

`logJobInfo_: write <logdir/info/jobfile> <data> failed: error`

`logJobInfo_: seek <logdir/info/jobfile> failed: error`

`logJobInfo_: write <logdir/info/jobfile> xdrpos <pos> failed: error`

`logJobInfo_: write <logdir/info/jobfile> xdr buf len <len> failed: error`

`logJobInfo_: close(<logdir/info/jobfile>) failed: error`

`rmLogJobInfo: Job <jobId>: can't unlink(<logdir/info/jobfile>): error`

`rmLogJobInfo_: Job <jobId>: can't stat(<logdir/info/jobfile>): error`

`readLogJobInfo: Job <jobId> can't open(<logdir/info/jobfile>): error`

`start_job: Job <jobId>: readLogJobInfo failed: error`

`readLogJobInfo: Job <jobId>: can't read(<logdir/info/jobfile>) size size: error`

`initLog: mkdir(<logdir/info>) failed: error`

`<fname>: fopen(<logdir/file> failed: error`

`getElogLock: Can't open existing lock file <logdir/file>: error`

`getElogLock: Error in opening lock file <logdir/file>: error`

```
releaseElogLock: unlink(<logdir/lockfile>) failed: error
```

```
touchElogLock: Failed to open lock file <logdir/file>: error
```

```
touchElogLock: close <logdir/file> failed: error
```

mbatchd failed to create, remove, read, or write the log directory or a file in the log directory, for the reason given in *error*. Check that LSF administrator has read, write, and execute permissions on the logdir directory.

If logdir is on AFS, check that the instructions in *Administering Platform LSF* have been followed. Use the fs ls command to verify that the LSF administrator owns logdir and that the directory has the correct ACL.

```
replay_newjob: File <logfile> at line <line>: Queue <queue> not
found, saving to queue <lost_and_found>replay_switchjob: File
<logfile> at line <line>: Destination queue <queue> not found,
switching to queue <lost_and_found>
```

When mbatchd was reconfigured, jobs were found in *queue* but that queue is no longer in the configuration.

```
replay_startjob: JobId <jobId>: exec host <host> not found, saving to
host <lost_and_found>
```

When mbatchd was reconfigured, the event log contained jobs dispatched to host, but that host is no longer configured to be used by LSF.

```
do_restartReq: Failed to get hData of host <host_name>/<host_addr>
```

mbatchd received a request from sbatchd on host *host_name*, but that host is not known to mbatchd. Either the configuration file has been changed but mbatchd has not been reconfigured to pick up the new configuration, or *host_name* is a client host but the sbatchd daemon is running on that host. Run the following command to reconfigure the mbatchd or kill the sbatchd daemon on *host_name*.

badmin reconfig

LSF command messages

```
LSF daemon (LIM) not responding ... still trying
```

During LIM restart, LSF commands will fail and display this error message. User programs linked to the LIM API will also fail for the same reason. This message is displayed when LIM running on the master host list or server host list is restarted after configuration changes, such as adding new resources, binary upgrade, and so on.

Use LSF_LIM_API_NTRIES in lsf.conf or as an environment variable to define how many times LSF commands will retry to communicate with the LIM API while LIM is not available. LSF_LIM_API_NTRIES is ignored by LSF and EGO daemons and all EGO commands.

When LSB_API_VERBOSE=Y in lsf.conf, LSF batch commands will display the not responding retry error message to stderr when LIM is not available.

When LSB_API_VERBOSE=N in lsf.conf, LSF batch commands will not display the retry error meesage when LIM is not available.

Batch command client messages

LSF displays error messages when a batch command cannot communicate with mbatchd. The following table provides a list of possible error reasons and the associated error message output.

Point of failure	Possible reason	Error message output
Establishing a connection with mbatchd	mbatchd is too busy to accept new connections. The connect() system call times out.	LSF is processing your request. Please wait…
	mbatchd is down or there is no process listening at either the LSB_MBD_PORT or the LSB_QUERY_PORT	LSF is down. Please wait…
	mbatchd is down and the LSB_QUERY_PORT is busy	bhosts displays "LSF is down. Please wait. . ."
		bjobs displays "Cannot connect to LSF. Please wait…"
	Socket error on the client side	Cannot connect to LSF. Please wait…
	connect() system call fails	Cannot connect to LSF. Please wait…
	Internal library error	Cannot connect to LSF. Please wait…
Send/receive handshake message to/from mbatchd	mbatchd is busy. Client times out when waiting to receive a message from mbatchd.	LSF is processing your request. Please wait…
	Socket read()/write() fails	Cannot connect to LSF. Please wait…
	Internal library error	Cannot connect to LSF. Please wait…

Batch event format errors

When mbatchd and batch commands fail reading lsb.events, LSF logs and displays details about which event field was reached when parsing of the event file failed.

event time_stamp offset[*byte*:*field*]: *field_name* [*field_name ...*]

where:

event

The name of the event being parsed.

time_stamp

The timestamp of the event. Use this field to search for the problem record from history event files.

offset[*byte*:*field*]

Byte offset is the approximate position in the line (number of characters from beginning of the line) where parsing the fields failed.

Field offset is the number of field where parsing failed.

field_name

the name of the fields being read when parsing failed.

Examples

Errors like the following are logged to the mbatchd log file when mbatchd fails to read lsb.events:

```
Dec 28 14:25:30 2008 9861 3 7.02 init_log: Reading event file </home/user1/
LSF7/work/LSF7/logdir/lsb.events>: Bad event format at line 15: JOB_NEW
1198869866 offset[28:3]: First 10 fields: jobId userId options numProcessors
subTime beginTime termTime sigValue chkpntPeriod restartPid
Dec 28 14:25:30 2008 9861 3 7.02 init_log: Reading event file </home/user1/
LSF7/work/LSF7/logdir/lsb.events>: Bad event format at line 16: bad
eventVersion
Dec 28 14:25:30 2008 9861 3 7.02 switch_log(): reading event file </home/
user1/LSF7/work/LSF7/logdir/lsb.events>: Bad event format at line 15: JOB_NEW
1198869866
offset[28:2]: First 10 fields: jobId userId options numProcessors subTime
beginTime termTime sigValue chkpntPeriod restartPid
Dec 28 14:25:30 2008 9861 3 7.02 switch_log(): reading event file </home/
user1/LSF7/work/LSF7/logdir/lsb.events>: Bad event format at line 16: bad
eventVersion
```

Batch event format errors like the following are displayed by LSF batch commands:

bhist -l 309
```
Dec 28 16:04:53 2008 8146 3 7.02 File /home/user1/LSF7/work/LSF7/logdir/
lsb.events: Bad event format at line 20: JOB_EXECUTE 1198888660 offset[48:1]:
execCwd
```
badmin mbdhist
```
File /home/user1/LSF7/work/LSF7/logdir/lsb.events: Bad event format at line
19: JOB_EXECUTE 1198888660 offset[48:1]: execCwd
File /home/user1/LSF7/work/LSF7/logdir/lsb.events: Bad event format at line
20: bad eventVersion
```

EGO command messages

```
You cannot run the egosh command because the administrator has chosen
not to enable EGO in lsf.conf: LSF_ENABLE_EGO=N.
```

If EGO is disabled, the egosh command cannot find ego.conf or cannot contact vemkd (not started).

Understanding Platform LSF job exit information

Contents

- Why did my job exit?
- How LSF translates events into exit codes
- Application and system exit values
- LSF job termination reason logging
- Job termination by LSF exit information
- LSF RMS integration exit values

Why did my job exit?

LSF collects job information and reports the final status of a job. Traditionally jobs finishing normally report a status of 0, which usually means the job has finished normally. Any non-zero status means that the job has exited abnormally.

Most of the time, the abnormal job exit is related either to the job itself or to the system it ran on and not because of an LSF error. This document explains some of the information LSF provides about the abnormal job termination.

How LSF translates events into exit codes

The following table summarizes LSF exit behavior for some common error conditions.

Error codition	LSF exit code	Operating system	System exit code equivalent	Meaning
Command not found	127	all	1 or 127	Command shell returns 1 if command not found. If the command cannot be found inside a job script, LSF return exit code 127.
Directory not available for output	0	all	1	LSF sends the output back to user through email if directory not available for output (bsub -o).
LSF internal error	-127, 127	all	N/A	RES returns -127 or 127 for all internal problems.
Out of memory	N/A	all	N/A	Exit code depends on the error handling of the application itself.
LSF job states	0	all	N/A	Exit code 0 is returned for all job states

Host failure

If an LSF server host fails, jobs running on that host are lost. No other jobs are affected. At initial job submission, you must submit a job with specific options for them to be automatically rerun from the beginning or restarted from a checkpoint on another host if they are lost because of a host failure.

- If a job is submitted with bsub -r or to a queue with RERUNNABLE set, it reruns automatically on host failure.
- If a job is submitted with bsub -k or to a checkpointable queue or application profile, it can be restarted if the host fails and the checkpoint succeeds.

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.

Exited jobs

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 its owner or the LSF administrator while pending, or after being dispatched to a host.
- The job is not able to be dispatched before it reaches its termination deadline, and thus is aborted by LSF.
- 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.

You can configure hosts so that LSF detects an abnormally high rate of job exit from a host. See *Administering Platform LSF* for more information.

Application and system exit values

LSF monitors a job while running and returns the exit code returned from the job itself. LSF collects this exit code via wait3() system call on UNIX platforms. The exit code is a result of the system exit values. Use bhist or bjobs to see the exit code for your job.

Application exit values

The most common cause of abnormal LSF job termination is due to application system exit values. If your application had an explicit exit value less than 128, bjobs and bhist display the actual exit code of the application; for example,
```
Exited with exit code 3
```
. You would have to refer to the application code for the meaning of exit code 3.

It is possible for a job to explicitly exit with an exit code greater than 128, which can be confused with the corresponding UNIX signal. Make sure that applications you write do not use exit codes greater than128.

System signal exit values

When you send a signal that terminates the job, LSF reports either the signal or the *signal_value* +128. If the return status is greater than 128, and the job was terminated with a signal, then *return_status*-128=signal. For example, return status 133 means that the job was terminated with signal 5 (SIGTRAP on most systems, 133-128=5). A job with exit status 130 was terminated with signal 2 (SIGINT on most systems, 130-128 = 2).

Some operating systems define exit codes as 0-255. As a result, negative exit values or values > 255 may have a wrap-around effect on that range. The most common example of this is a program that exits -1 will be seen with "exit code 255" in LSF.

How or why the job may have been signaled, or exited with a certain exit code, can be application and/or system specific. The application or system logs might be able to give a better description of the problem.

> **Note:**
>
> Termination signals are operating system dependent, so signal 5 may not be SIGTRAP and 11 may not be SIGSEGV on all UNIX and Linux systems. You need to pay attention to the execution host type in order to correct translate the exit value if the job has been signaled.

bhist and bjobs output

In most cases, bjobs and bhist show the application exit value (128 + *signal*). In some cases, bjobs and bhist show the actual signal value.

If LSF sends catchable signals to the job, it displays the exit value. For example, if you run bkill *jobID* to kill the job, LSF passes SIGINT, which causes the job to exit with exit code 130 (SIGINT is 2 on most systems, 128+2 = 130).

If LSF sends uncatchable signals to the job, then the entire process group for the job exits with the corresponding signal. For example, if you run bkill -s SEGV *jobID* to kill the job, bjobs and bhist show

```
Exited by signal 7
```

Example

The following example shows a job that exited with exit code 139, which means that the job was terminated with signal 11 (SIGSEGV on most systems, 139-128=11). This means that the application had a core dump.

bjobs -l 2012
```
Job <2012>, User , Project , Status , Queue , Command
Fri Dec 27 22:47:28: Submitted from host , CWD <$HOME>;
Fri Dec 27 22:47:37: Started on , Execution Home , Execution CWD ; Fri Dec 27
22:48:02: Exited with exit code 139. The CPU time used is 0.2 seconds.
SCHEDULING PARAMETERS:
          r15s  r1m  r15m   ut       pg    io   ls   it    tmp    swp
mem
loadSched   -     -    -     -        -     -    -    -     -      -      -
loadStop    -     -    -     -        -     -    -    -     -      -      -
                cpuspeed    bandwidth
 loadSched       -           -
 loadStop        -           -
```

LSF job termination reason logging

When LSF takes action on a job, it may send multiple signals. In the case of job termination, LSF will send, SIGINT, SIGTERM and SIGKILL in succession until the job has terminated. As a result, the job may exit with any of those corresponding exit values at the system level. Other actions may send "warning" signals to applications (SIGUSR2) etc. For specific signal sequences, refer to the LSF documentation for that feature.

Run bhist to see the actions that LSF takes on a job:

bhist -l 1798

```
Job <1798>, User <user1>, Command <sleep 10000>
Tue Feb 25 16:35:31: Submitted from host <hostA>, to Queue <normal>, CWD <$H
                     OME/lsf_7.0/conf/lsbatch/lsf_7.0/configdir>;
Tue Feb 25 16:35:51: Dispatched to <hostA>;
Tue Feb 25 16:35:51: Starting (Pid 12955);
Tue Feb 25 16:35:53: Running with execution home </home/user1>, Execution CWD <
                     /home/user1/Testing/lsf_7.0/conf/lsbatch/lsf_7.0/configdir>,
                     Execution Pid <12955>;
Tue Feb 25 16:38:20: Signal <KILL> requested by user or administrator <user1>;
Tue Feb 25 16:38:22: Exited with exit code 130. The CPU time used is 0.1 seconds;
Summary of time in seconds spent in various states by  Tue Feb 25 16:38:22
PEND     PSUSP     RUN      USUSP     SSUSP     UNKWN     TOTAL
 20        0       151        0         0         0        171
```

Here we see that LSF itself sent the signal to terminate the job, and the job exits 130 (130-128 = 2 = SIGINT).

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

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

View logged job exit information (bacct -l)

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

bacct -l 7265
```
Accounting information about jobs that are:
  - submitted by all users.
  - accounted on all projects.
  - completed normally or exited
  - executed on all hosts.
  - submitted to all queues.
  - accounted on all service classes.
------------------------------------------------------------------------------
Job <7265>, User <lsfadmin>, Project <default>, Status <EXIT>, Queue <normal>,
                  Command <srun sleep 100000>
Thu Sep 16 15:22:09: Submitted from host <hostA>, CWD <$HOME>;
Thu Sep 16 15:22:20: Dispatched to 4 Hosts/Processors <4*hostA>;
Thu Sep 16 15:23:21: Completed <exit>; TERM_RUNLIMIT: job killed after reaching
          LSF run time limit.
Accounting information about this job:
     Share group charged </lsfadmin>
     CPU_T      WAIT     TURNAROUND    STATUS    HOG_FACTOR    MEM    SWAP
     0.04        11           72       exit        0.0006      OK      OK
------------------------------------------------------------------------------
SUMMARY:        ( time unit: second )
 Total number of done jobs:       0      Total number of exited jobs:     1
 Total CPU time consumed:       0.0      Average CPU time consumed:      0.0
 Maximum CPU time of a job:     0.0      Minimum CPU time of a job:      0.0
 Total wait time in queues:    11.0
 Average wait time in queue:   11.0
 Maximum wait time in queue:   11.0      Minimum wait time in queue:    11.0
 Average turnaround time:        72 (seconds/job)
 Maximum turnaround time:        72      Minimum turnaround time:         72
 Average hog factor of a job:  0.00 ( cpu time / turnaround time )
 Maximum hog factor of a job:  0.00      Minimum hog factor of a job:   0.00
```

Termination reasons displayed by bacct

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

Keyword displayed by bacct	Termination reason	Integer value logged to JOB_FINISH in lsb.acct
TERM_ADMIN	Job killed by root or LSF administrator	15
TERM_BUCKET_KILL	Job killed with bkill -b	23
TERM_CHKPNT	Job killed after checkpointing	13
TERM_CPULIMIT	Job killed after reaching LSF CPU usage limit	12
TERM_CWD_NOTEXIST	Current working directory is not accessible or does not exist on the execution host	25
TERM_DEADLINE	Job killed after deadline expires	6
TERM_EXTERNAL_SIGNAL	Job killed by a signal external to LSF	17
TERM_FORCE_ADMIN	Job killed by root or LSF administrator without time for cleanup	9
TERM_FORCE_OWNER	Job killed by owner without time for cleanup	8
TERM_LOAD	Job killed after load exceeds threshold	3
TERM_MEMLIMIT	Job killed after reaching LSF memory usage limit	16
TERM_OTHER	Member of a chunk job in WAIT state killed and requeued after being switched to another queue.	4
TERM_OWNER	Job killed by owner	14
TERM_PREEMPT	Job killed after preemption	1
TERM_PROCESSLIMIT	Job killed after reaching LSF process limit	7
TERM_REQUEUE_ADMIN	Job killed and requeued by root or LSF administrator	11
TERM_REQUEUE_OWNER	Job killed and requeued by owner	10
TERM_RMS	Job exited from an RMS system error	18
TERM_RUNLIMIT	Job killed after reaching LSF run time limit	5
TERM_SLURM	Job terminated abnormally in SLURM (node failure)	22
TERM_SWAP	Job killed after reaching LSF swap usage limit	20
TERM_THREADLIMIT	Job killed after reaching LSF thread limit	21
TERM_UNKNOWN	LSF cannot determine a termination reason—0 is logged but TERM_UNKNOWN is not displayed	0
TERM_WINDOW	Job killed after queue run window closed	2
TERM_ZOMBIE	Job exited while LSF is not available	19

Tip:

The integer values logged to the JOB_FINISH event in `lsb.acct` and termination reason keywords are mapped in `lsbatch.h`.

Restrictions

- If a queue-level JOB_CONTROL is configured, LSF cannot determine the result of the action. The termination reason only reflects what the termination reason could be in LSF.
- LSF cannot be guaranteed to catch any external signals sent directly to the job.
- In MultiCluster, a `brequeue` request sent from the submission cluster is translated to TERM_OWNER or TERM_ADMIN in the remote execution cluster. The termination reason in the email notification sent from the execution cluster as well as that in the `lsb.acct` is set to TERM_OWNER or TERM_ADMIN.

Example output of bacct and bhist

Example termination cause	Termination reason in bacct –l	Example bhist output
bkill -s KILL bkill *job_ID*	Completed <exit>; TERM_OWNER or TERM_ADMIN	Thu Mar 13 17:32:05: Signal <KILL> requested by user or administrator <user2>; Thu Mar 13 17:32:06: Exited by signal 2. The CPU time used is 0.1 seconds;
bkill –r	Completed <exit>; TERM_FORCE_ADMIN or TERM_FORCE_OWNER when sbatchd is not reachable. Otherwise, TERM_USER or TERM_ADMIN	Thu Mar 13 17:32:05: Signal <KILL> requested by user or administrator <user2>; Thu Mar 13 17:32:06: Exited by signal 2. The CPU time used is 0.1 seconds;
TERMINATE_WHEN	Completed <exit>; TERM_LOAD/ TERM_WINDOWS/ TERM_PREEMPT	Thu Mar 13 17:33:16: Signal <KILL> requested by user or administrator <user2>; Thu Mar 13 17:33:18: Exited by signal 2. The CPU time used is 0.1 seconds;
Memory limit reached	Completed <exit>; TERM_MEMLIMIT	Thu Mar 13 19:31:13: Exited by signal 2. The CPU time used is 0.1 seconds;
Run limit reached	Completed <exit>; TERM_RUNLIMIT	Thu Mar 13 20:18:32: Exited by signal 2. The CPU time used is 0.1 seconds.
CPU limit	Completed <exit>; TERM_CPULIMIT	Thu Mar 13 18:47:13: Exited by signal 24. The CPU time used is 62.0 seconds;
Swap limit	Completed <exit>; TERM_SWAPLIMIT	Thu Mar 13 18:47:13: Exited by signal 24. The CPU time used is 62.0 seconds;

Example termination cause	Termination reason in bacct –l	Example bhist output
Regular job exits when host crashes	Rusage 0, Completed <exit>; TERM_ZOMBIE	Thu Jun 12 15:49:02: Unknown; unable to reach the execution host; Thu Jun 12 16:10:32: Running; Thu Jun 12 16:10:38: Exited with exit code 143. The CPU time used is 0.0 seconds;
brequeue –r	For each requeue, Completed <exit>; TERM_REQUEUE_ADMIN or TERM_REQUEUE_OWNER	Thu Mar 13 17:46:39: Signal <REQUEUE_PEND> requested by user or administrator <user2>; Thu Mar 13 17:46:56: Exited by signal 2. The CPU time used is 0.1 seconds;
bchkpnt -k	On the first run: Completed <exit>; TERM_CHKPNT	Wed Apr 16 16:00:48: Checkpoint succeeded (actpid 931249); Wed Apr 16 16:01:03: Exited with exit code 137. The CPU time used is 0.0 seconds;
Kill –9 <RES> and job	Completed <exit>; TERM_EXTERNAL_SIGNAL	Thu Mar 13 17:30:43: Exited by signal 15. The CPU time used is 0.1 seconds;
Others	Completed <exit>;	Thu Mar 13 17:30:43: Exited with 3; The CPU time used is 0.1 seconds;

Job termination by LSF exit information

LSF also provides additional information in the POST_EXEC of the job. Use this information to detect conditions where LSF has terminated the job and take the appropriate action.

The job exit information in the POST_EXEC is defined in 2 parts:

- LSB_JOBEXIT_STAT—the raw wait3() output (converted using the wait macros /usr/include/sys/wait.h)
- LSB_JOBEXIT_INFO—defined only if the job exit was due to a defined LSF reason.

Queue-level POST_EXEC commands should be written by the cluster administrator to perform whatever task is necessary for specific exit situations.

> **Note:**
>
> System level enforced limits like CPU and Memory (listed above), cannot be shown in the LSB_JOBEXIT_INFO since it is the operating system performing the action and not LSF. Set appropriate parameters in the queue or at job submission to allow LSF to enforce the limits, which makes this information available to LSF.

Common LSB_JOBEXIT_STAT and LSB_JOBEXIT_INFO values

The following is a table of common scenarios covered and not covered by the LSB_JOBEXIT_INFO

Example termination cause	LSB_JOBEXIT_STAT	LSB_JOBEXIT_INFO	Example bhist output
Job killed with the SIGINT bkill -s INT 520	33280	SIGNAL 2 INT	Fri Feb 14 16:48:00: Exited with exit code 130. The CPU time used is 0.2 seconds;
Job killed with SIGTERM bkill -s TERM 521	36608	SIGNAL 15 TERM	Fri Feb 14 16:49:50: Exited with exit code 143. The CPU time used is 0.2 seconds;
Job killed with SIGKILL bkill -s KILL 522	33280	SIGNAL -14 SIG_TERM_USER	Fri Feb 14 16:51:03: Exited with exit code 130. The CPU time used is 0.2 seconds;
Automatic migration when MIG is defined at queue level	33280	SIGNAL -1 SIG_CHKPNT	Fri Feb 14 17:32:17: Job has been requeued; Fri Feb 14 17:32:17: Pending: Migrating job is waiting for rescheduling;
bsub –I "hostname;exit 130"	33280	Undefined	Fri Feb 14 14:41:51: Exited with exit code 130. The CPU time used is 0.2 seconds;
Killing the job with bkill command bkill 210	33280	SIGNAL -14 SIG_TERM_USER	Fri Feb 14 14:45:51: Exited with exit code 130. The CPU time used is 0.2 seconds;
Job being brequeued. brequeue -r Job <211> is being requeued	33280	SIGNAL -23 SIG_KILL_REQUEUE	Fri Feb 14 14:48:15: Signal <REQUEUE_PEND> requested by user or administrator <iayaz>; Fri Feb 14 14:48:18: Exited with exit code 130. The CPU time used is 0.2 second
Job being migrated bmig -m togni Job <213> is being migrated	33280	SIGNAL -1 SIG_CHKPNT	Fri Feb 14 15:04:42: Migration requested by user or administrator <iayaz>; Specified Hosts <togni>; Fri Feb 14 15:04:44: Job is being requeued; Fri Feb 14 15:05:01: Job has been requeued; Fri Feb 14 15:05:01: Pending: Migrating job is waiting for rescheduling;
Job killed due REQUEUE_EXIT_VALUE bsub "sleep 100;exit 34"	8704	Undefined	Fri Feb 14 15:10:21: Pending: Requeued job is waiting for rescheduling;(exit code 34)>;
Job killed by LSF when CPULIMIT enforced by LSF	158	SIGNAL -24 SIG_TERM_CPULIMIT	Wed Feb 19 14:18:13: Exited by signal 30. The CPU time used is 89.4 seconds.
Job killed because queue level CPULIMIT is reached.	40448	Undefined	Fri Feb 14 15:30:01: Exited with exit code 158. The CPU time used is 61.2 seconds;

Example termination cause	LSB_JOBEXIT_STAT	LSB_JOBEXIT_INFO	Example bhist output
Job killed because queue level RUNLIMIT is reached.	37120	Undefined	Fri Feb 14 15:37:44: Exited with exit code 145. The CPU time used is 0.2 seconds;
Job killed due to the check pointing. bchkpnt -k 838 Job <838> is being checkpointed	9	SIGNAL -1 SIG_CHKPNT	Fri Feb 14 17:59:12: Checkpoint succeeded (actpid 25298); Fri Feb 14 17:59:12: Exited by signal 9. The CPU time used is 0.1 seconds;
Job killed when reaches the MEMLIMIT bsub -M 5 "/home/iayaz/script/memwrite -m 10 -r 2"	2	SIGNAL -25 SIG_TERM_MEMLIMIT	Fri Feb 21 10:50:50: Exited by signal 2. The CPU time used is 0.1 seconds;
Job killed when termination time approaches bsub -t 21:11:10 sleep 500;date	37120	Undefined	Exited with exit code 145. The CPU time used is 0.2 seconds;
Job killed when TERMINATE_WHEN = LOAD	33280	SIGNAL -15 SIG_TERM_LOAD	Exited with exit code 130. The CPU time used is 7.2 seconds.
Job killed when TERMINATE_WHEN = PREEMPT	33280	SIGNAL -16 SIG_TERM_PREEMPT	Exited with exit code 130. The CPU time used is 0.3 seconds;

LSF RMS integration exit values

For the RMS integrations with LSF (HP AlphaServer SC and Linux QsNet), LSF jobs running through RMS will return rms_run() return code as the job exit code. RMS documents certain exit codes and corresponding job exit reasons.

See the rms_run() man page for more information.

Upon successful completion, rms_run() returns the global OR of the exit status values of the processes in the parallel program. If one of the processes is killed, rms_run() returns a status value of 128 plus the signal number. It can also return the following codes:

Return Code	RMS Meaning
0	A process exited with the code 127 (GLOBAL EXIT), which indicates success, causing all of the processes to exit.
123	A process exited with the code 123 (GLOBAL ERROR) causing all of the processes to exit.
124	The node the job executing on has been removed from the system.
125	One or more processes were still running when the exit timeout expired.
126	The resource is inadequate for the request.

Index