Release Notes for Platform LSF

Platform LSF
Version 8.0.1
Release date: June 2011
Last modified: June 28 2011

Platform Computing

Contents

Release Notes for Platform LSF

Version 8.0.1

Release date: June 2011

Last modified: June 28, 2011

Comments to: *doc@platform.com*

Support: *support@platform.com*

Upgrade and Compatibility Notes

For additional information about Platform LSF Version 8.0.1, visit the Platform Computing web site:

http://www.platform.com/Products/platform-lsf/features-benefits

Master host selection

To achieve the highest degree of performance and scalability, we strongly recommend that you use a powerful master host.

There is no minimum CPU requirement. For the platforms LSF is supported on, any host with sufficient physical memory can run LSF as master host. Swap space is normally configured as twice the physical memory. LSF daemons use about 20 MB of memory when no jobs are running. Active jobs consume most of the memory LSF requires.

Cluster size	Active jobs	Minimum Recommended Memory	Recommended server CPU (Intel, AMD, or equivalent)
Small (<100 hosts)	1,000	1 GB	any server CPU
	10,000	2 GB	recent server CPU
Medium (100-1000 hosts)	10,000	4 GB	multi-core CPU (2 cores)
	50,000	8 GB	multi-core CPU (4 cores)
Large (>1000 hosts)	50,000	16 GB	multi-core CPU (4 cores)
	500,000	32 GB	multi-core CPU (8 cores)

Server host compatibility

Platform LSF Version Version 8.0.1 is fully compatible with Platform LSF Version Version 8.0.

LSF 7.*x* and 6.*x* servers are compatible with Platform LSF Version 8 master hosts. All LSF 7.*x* and 6.*x* features are supported by Platform LSF Version 8 master hosts.

Important:

To use new features introduced in Platform LSF Version 8.0.1, you *must* upgrade all hosts in your cluster running LSF 7.*x* or earlier to Platform LSF Version 8.0.1.

Upgrade from an earlier version of Platform LSF on UNIX and Linux

Follow the steps in *Upgrading Platform LSF on UNIX and Linux* (`lsf_upgrade_unix.pdf`) to run `lsfinstall` to *upgrade* LSF:

- Upgrade a pre-LSF Version 7 UNIX or Linux cluster to Platform LSF Version 8
- Upgrade an LSF Version 7 Update 2 through Update 6 UNIX or Linux cluster to Platform LSF Version 8

Important:

DO NOT use the UNIX and Linux upgrade steps to migrate an existing LSF Version 7 or LSF 7 Update 1 cluster to LSF Version 8. Follow the manual steps in the document *Migrating to Platform LSF Version 8 on UNIX and Linux* to migrate an existing LSF Version 7 or LSF 7 Update 1 cluster to LSF Version 8 on UNIX and Linux.

Migrate LSF Version 7 or LSF 7 Update 1 cluster to LSF 8 on UNIX and Linux

Follow the steps in *Migrating to Platform LSF Version 8 on UNIX and Linux* (`lsf_migrate_unix.pdf`) to migrate an *existing* LSF 7 cluster:

- Migrate an existing LSF Version 7 cluster to LSF Version 8 on UNIX and Linux
- Migrate an existing LSF 7 Update 1 cluster to LSF Version 8 on UNIX and Linux

Note:

DO NOT use these steps to migrate an existing LSF 7 Update 2 or higher cluster to LSF Version 8. Follow the steps in *Upgrading Platform LSF on UNIX and Linux* to upgrade LSF.

Migrate LSF Version 7 cluster to LSF Version 8 cluster on Windows

To migrate an *existing* LSF 7 Windows cluster to Platform LSF Version 8 on Windows, follow the steps in *Migrating Platform LSF Version 7 to Platform LSF Version 8 to Windows* (`lsf_migrate_windows_to_8.pdf`).

Note:

DO NOT use these steps to migrate a pre-version 7 cluster to LSF Version 8. Pre-version 7 clusters must first be migrated to LSF Version 7.

Bug fixes and solutions in this release

At release, LSF Version 8.0 includes all bug fixes up to and including November 29, 2010, and all solutions delivered up to and including September 30, 2010.

LSF 8.0.1 includes all bugs fixed between October 1 2010 and May 17 2011.

Bug fixes between May 20 2011 and LSF 8.0.1 release, and all solutions not included in the LSF 8.0.1 release will be available in the first LSF 8.0.1 quarterly maintenance pack.

Bug fixes and solutions delivered after LSF 8.0.1 release will be available in the next LSF release.

System requirements

Visit the Platform Computing Web site for information about supported operating systems and system requirements for Platform LSF:

http://www.platform.com/Products/platform-lsf/technical-information

API compatibility

To take full advantage of new Platform LSF Version 8 features, you should recompile your existing Platform LSF applications with Platform LSF Version 8.

Applications need to be rebuilt if they use APIs that have changed in Platform LSF Version 8.

New and changed Platform LSF APIs

See the *Platform LSF API Reference* for more information.

The following APIs have changed for LSF Version 8:

- `lsb_queueinfo`
- `lsb_serviceClassInfo`
- `lsb_modify`
- `lsb_submit`
- `checkUserGroupAdmin`
- `lsb_hostgrpinfo`
- `lsb_cu_info`
- `lsb_usergrpinfo`
- `lsb_geteventrec`
- `lsb_readjobinfo`
- `lsb_geteventrecbyline`

The following APIs were created for LSF Version 8:

- `lsb_liveconfig`: Live reconfiguration.
- `lsb_guaranteedResourcePoolInfo`: Returns information about guaranteed resource pools.
- `lsb_freeGuaranteedResourcePoolEntArray`: Frees the memory used by an array of type guaranteedResourcePoolEnt.
- `lsb_submitPack`: Submits multiple jobs using the pack method in the batch system.

SSH

Since LSF Version 7 Update 4, Platform LSF supports OpenSSH (SSH-1 and SSH-2).

FlexNet Publisher (formerly FLEXlm)

- All master LIM and FlexNet binaries on certified platforms are built with FlexNet 10.8.5.0 or above, except for Mac OS (FlexNet 10.8.6). If you are using your own license server, it must run FlexNet version 10.8.5.0 or later.
- If the master host is a certified platform, the FlexNet server must be 10.8.5.0.

  For a list of certified platforms, see

  http://www.platform.com/workload-management/high-performance-computing/technical-information

Platform License Scheduler Version 8.0

Platform License Scheduler Version 8.0 introduces several new enhancements to improve policy-driven application license sharing for Platform LSF clusters and projects. Platform License Scheduler enables:

- Significantly improved license utilization
- Increased user productivity
- Maximum return on license investments

See the *Platform License Scheduler Release Notes* for more information.

What's Changed in Platform LSF Version 8.0.1

New and changed behavior

Job pre-execution processing timeout

The new parameter JOB_PREPROC_TIMEOUT limits the time pre-execution processing is allowed to run. The timeout is defined in minutes. When the timeout is reached, LSF kills the pre-execution processes and requeues the job to the head of the queue.

The parameter can be defined in an application profile in `lsb.applications`, or at the cluster level in `lsb.params`. The value defined in the application profile overrides the value in `lsb.params`.

To enable job pre-execution processing timeout, set JOB_PREPROC_TIMEOUT in lsb.applications or lsb.params. By default, the parameter is not defined, and pre-execution processing does not time out.

Specify a timeout in minutes for job pre-execution processing. The specified timeout must be an integer greater than zero.

If the job's pre-execution processing takes longer than the timeout, LSF kills the job's pre-execution processes, kills the job with a pre-defined exit value of 98, and then requeues the job to the head of the queue. However, if the number of pre-execution retries has reached the limit, LSF suspends the job with PSUSP status instead of requeuing it.

JOB_PREPROC_TIMEOUT defined in an application profile in lsb.applications overrides the value in lsb.params. JOB_PREPROC_TIMEOUT cannot be defined in the user environment.

On UNIX and Linux, sbatchd kills the entire process group of the job's pre-execution processes.

On Windows, only the parent process of the pre-execution command is killed when the timeout expires, the child processes of the pre-execution command are not killed.

In the MultiCluster job forwarding model, JOB_PREPROC_TIMEOUT and the number of pre-execution retries defined in the receiving cluster apply to the job. When the number of attempts reaches the limit, the job returns to submission cluster and is rescheduled.

In the MultiCluster job lease model, JOB_PREPROC_TIMEOUT and the number of pre-execution retries defined in the submission cluster apply to jobs running on remote leased hosts, as if they were running on local hosts.

Automatic Time-based configuration in lsb.applications

Automatic time-based configuration is now supported in the file `lsb.applications`. It works the same as in other LSF configuration files.

Use if-else constructs and time expressions to define time windows in the file. Configuration defined within a time window applies only during the specified time period; configuration defined outside of any time window applies at all times. After editing the file, run `badmin reconfig` to reconfigure the cluster.

Time expressions in the file are evaluated by LSF every 10 minutes, based on `mbatchd` start time. When an expression evaluates true, LSF changes the configuration in real time, without restarting `mbatchd`, providing continuous system availability.

For example:

Begin application

NAME=app1

#if time(16:00-18:00)

CPULIMIT=180/hostA

#else

CPULIMIT=60/hostA

#endif

End application

In this example, for two hours every day, the configuration is:

Begin application

NAME=app1

CPULIMIT=180/hostA

End application

The rest of the time, the configuration is:

Begin application

NAME=app1

CPULIMIT=60/hostA

End application

Improved preemption performance

Performance has been greatly improved for clusters that have configured PREEMPT_FOR=LEAST_RUN_TIME.

Preemption delay

This feature can provide flexibility to tune the system to reduce the number of preemptions. It is useful to get better performance and job throughput.

When the low priority jobs are short, if high priority jobs can wait a while for the low priority jobs to finish, preemption can be avoided and cluster performance is improved. This feature adds a parameter to control the preemptive jobs' waiting time.

Preemptive pending jobs wait for a specific time after submission.

Before the time expires, preemption will not be triggered, but the job can be dispatched by other scheduling policies. If the job is still pending after the time has expired, the preemption will be triggered.

The waiting time is for preemptive jobs in the pending status only. It will not impact the preemptive jobs that are suspended.

The time is counted from the submission time of the jobs. The submission time means the time `mbatchd` accepts a job, which includes newly submitted jobs, restarted jobs (by `brestart`) or forwarded jobs from a remote cluster.

When the preemptive job is waiting, the pending reason is
```
The preemptive job is allowing a grace period before preemption.
```
This pending reason is shown by newer versions of `bjobs`. The older `bjobs` command shows
```
Unknown pending reason code <6701>;
```
.

If your applications use LSF APIs, and they use a static link with LSF libraries, you must rebuild with the patched header file and libraries to let it show the correct pending reason. If your applications use a dynamic link with LSF libraries, they can show the correct pending reason after patching the libraries.

The parameter is defined in `lsb.params`, `lsb.queues` (overrides `lsb.params`), and `lsb.applications` (overrides both `lsb.params` and `lsb.queues`).

Set PREEMPT_DELAY=*seconds*

Preemptive jobs will wait the specified number of seconds from the submission time before preempting any low priority preemptable jobs. During the grace period, preemption will not be triggered, but the job can be scheduled and dispatched by other scheduling policies.

By default, the parameter is not defined anywhere, and preemption is immediate.

Improved startup and shutdown performance

Performance has been improved, and you can start, stop, and restart LSF daemons faster. The PDSH tool is required on master hosts and master candidates.

```
lsfstartup -pdsh [-num_hosts number] [-delay seconds]
```

The new options are:

- -pdsh: enable parallel remote command execution with PDSH.
- -num_hosts: number of hosts in one chunk. Valid values are 1 to 512. The default value is 250.
- -delay: time interval between chunks. Valid values are 1 to 60. The default value is 8 seconds.

`lsfstartup -pdsh` will:

- Divide slave hosts into chunks, each has *num_hosts* hosts
- Start slave hosts in parallel inside one chunk with PDSH
- Wait the number of seconds specified by -delay before starting the next chunk

For `lsfshutdown -pdsh`:

- You do not need to specify chunk size. The default value is 400.
- You do not need to specify chunk interval. The next chunk can begin to shut down as soon as the previous chunk is finished.

For `lsfrestart -pdsh`:

- You do not need to specify chunk size. The default value is 400.
- You do not need to specify chunk interval. The next chunk can begin to restart as soon as the previous chunk is finished.

Support for Gold v.2.2

Gold is a dynamic accounting system that tracks and manages resource usage in a cluster. LSF is now integrated with Gold v2.2. The Platform LSF integration allows dynamic accounting in Gold. The following GOLD features are supported:

Job quotations at the time of job submission

- Job reservations at the start time of jobs
- Job charges when jobs are completed

Gold v2.2 (or newer) is supported on Linux and UNIX. Complete the following steps to install the Gold integration:

1. Install Gold in the default location: `/opt/gold`, or export the PATH to var $GOLD_PATH. For example:
   ```
   setenv GOLD_PATH /opt/gold/bin/
   ```
2. Configure ABS_RUNLIMIT=Y in `lsf.params`. Factor scaling between different CPUs is not considered in Gold.
3. Synchronize all cluster host clocks to make sure clock times are identical on all hosts.
4. Enable `esub.gold` and `eexec.gold` to be executed by all jobs:
   1. Specify LSB_ESUB_METHOD=esub.gold in `lsf.conf`.
   2. Copy `eexec.gold` to your `LSF_SERVERDIR` and rename it to `eexec`.

When installation is complete, submitted jobs call `esub.gold` and `eexec.gold` scripts in order to:

- Check quotations in Gold at job submission
- Reserve resources in Gold when a job begins to run
- Charge in Gold when a job is complete

Complete the following steps to configure the GOLD integration:

a. Configure the same list of users and hosts in both GOLD and LSF.

The name of a machine registered in GOLD must be identical to the LSF execution host name when submitting jobs to this host.

b. Ensure all hosts in the cluster have access to the GOLD commands in order to run the `esub` and `eexec` scripts successfully.

Note the following:

- Detailed information logging can be configured using the environment variable LSF_CMD_LOGDIR on job submission.
- By default, there is no log file for the `esub.gold` script.

- If LSF_CMD_LOGDIR=*esub_log_path* is configured in the submission environment, a log file `esub.log.`*`user_name`* is generated. If the `esub` script cannot access the log directory, the `esub` log file is written to the `/tmp` directory.
- By default, only failure messages are recorded for the `eexec.gold` script.
- With logging enabled, LSF tries first to log to `LSF_LOGDIR/eexec.log.`*`user_name`*. If this directory cannot be accessed, the `eexec` logs to `/tmp/eexec.log.`*`user_name`* on the execution host.
- Use `bhist -l` to check the failure reasons for charge or reserve actions in Gold for LSF jobs.
- Job runtime is counted in Gold instead of job CPU time.
- Resizable jobs are not fully supported. Only processor usage at the end of the job is considered.
- The `bmod` command is not supported.
- The maximum number of processors specified by `bsub -n` is counted.

tssub and tspeek password cache

This enhancement allows you to run `tssub -I` or `tspeek` on Windows and view your remote desktop without being prompted for your password every time. You only need to input your password once.

This solution is supported on Windows (x64, x86).

Setting the LSF environment using a Perl script

`perl.lsf` is a Perl script that includes a function `setLSFEnv()`, which does two things: sources `profile.lsf`, and sets all LSF environment variables into the Perl environment hash table. In `perl.lsf`, the function `setLSFEnv()` is provided to set LSF environment variables.

The following LSF environments variables are set by calling `setLSFEnv()`.

PATH

MANPATH

LSF_BINDIR

LSF_SERVERDIR

LSF_LIBDIR

LD_LIBRARY_PATH

LSF_ENVDIR

EGO_TOP

EGO_BINDIR

EGO_SERVERDIR

EGO_LIBDIR

EGO_CONFDIR

EGO_LOCAL_CONFDIR

EGO_ESRVDIR

Steps:

1. Add
   ```
   require perl.lsf
   ```
   to your own Perl script. For example:
   ```
   "require ("/scratch/dev/lsf/elly/clusters/test/conf/perl.lsf");"
   ```

2. Call function `setLSFEnv()` when you need to set LSF environment, for example

   setLSFEnv();

   bsub sleep 99

Keep expired dynamic hosts

Expired dynamic hosts can now be kept in LSF memory and filtered from the output of LSF commands. `mbatchd` does not reconfigure automatically when dynamic hosts expire or are unavailable for more than the time set by LSF_DYNAMIC_HOST_TIMEOUT in `lsf.conf`. Expired hosts are not displayed by `lshosts`, `lsload`, `lsmon`, `bhosts`, etc.

This feature is supported on 64-bit X86 Linux 2.6 glibc 2.3.

When LSF_DYNAMIC_HOST_KEEP=Y, LIM keeps expired dynamic host information in memory, but still removes entries for expired hosts in the `hostcache` file. When expired dynamic hosts rejoin the cluster, LIM records them in the `hostcache` file once again.

If dynamic hosts rejoin the cluster with different names, LIM considers these as new hosts and adds them to the host list and `hostcache` file. `mbatchd` also adds these as new hosts, and `lshosts`, `lsload` and `bhosts` display information for these hosts.

To clean up unnecessary host information in LIM and `mbatchd` memory, you must restart LIM. Restart causes automatic `mbatchd` reconfiguration automatically if dynamic hosts have become unavailable and have not rejoined the cluster. Master LIM enforces license checking when dynamic hosts expire or when expired dynamic hosts rejoin the cluster. LIM licenses expired dynamic hosts using current licensing logic for client hosts.

There are some limitations to this feature:

- When expired dynamic hosts rejoin the cluster and change local resources, LIM cannot update local resource information without running `lim reconfig`.
- Output displayed by `lshosts -s`, `lsload -s`, `bhosts -s`, `brsvs`, and `bjobs` does not filter out expired dynamic hosts.
- `mbatchd` only returns a warning message if expired dynamic hosts are defined in `lsb.*` configuration files when you run `lim reconfig`.

Detailed pending reason display for external scheduler plugin

This solution provides a way for an external scheduler plugin to provide custom pending reason information, in the traditional PENDING REASON area of `bjobs`.

In LSF 8.0.1, an external scheduler pending reason parameter PEND_EXTSCHED_REASON is introduced in `lsb.params`, which displays a detailed pending reason in `bjobs -p` and `bjobs -l` output.

When implementing an external scheduler plugin, developers can specify a main pending reason as PEND_EXTSCHED_REASON and specify a custom detailed pending reason when a job cannot be dispatched. The main reason is displayed as

```
External Scheduler reason:
```

and the custom detailed pending reason is displayed below the main pending reason. For example:

```
# bjobs -p
JOBID   USER    STAT  QUEUE      FROM_HOST   EXEC_HOST   JOB_NAME   SUBMIT_TIME
110     admin   PEND  normal     hostA       sleep 1                May 11  13:32
 External scheduler reason:
 Detail reason: custom_pending_reason_display
```

Custom detailed pending reason display only supports the main pending reason. It does not not support a host pending reason.

This feature is supported on all LSF platforms.

What's Changed in Platform LSF Version 8

New and changed behavior

Live reconfiguration

You can use live reconfiguration to make configuration changes in LSF active memory that take effect immediately, faster than if you had to run `badmin reconfig`. Live reconfiguration requests use the new `bconf` command, and generate updated configuration files in the directory set by LSF_LIVE_CONFDIR in `lsf.conf`.

Live configuration changes made using the `bconf` command are recorded in the history file `liveconf.hist` located under $LSB_SHAREDIR, and can be queried using the `bconf hist` command. Not all configuration changes are supported by `bconf`.

When a file exists in the directory set by LSF_LIVE_CONFDIR, all LSF restart and reconfig commands read the file in this directory instead of the equivalent configuration file in LSF_CONFDIR.

Guarantee SLAs

You can now guarantee resources such as hosts or slots by configuring guaranteed resource pools with new guarantee-type SLAs.

Each guaranteed resource pool configured in `lsb.resources` contains resources, guarantee distribution policies, and (optionally) loan policies. Guarantees are made to Service Classes defined in `lsb.serviceclasses`, identifying which consumers are guaranteed the resources. Only guarantee-type SLAs can be used with guaranteed resource pools.

LSF reserves sufficient resources to honor guarantees without changing the scheduling priority in any other way. Jobs are considered for dispatch according to whatever other scheduling features are considered, and then also have access to the guaranteed resources. You can automatically attach jobs to guarantee SLAs for complete transparency to users.

By default guaranteed resources are reserved and left idle when not in use; configuring loan policies allows jobs without guarantees access to guaranteed resources. Resource pools can loan to specific queues, to jobs shorter than a configured length (based on runtime estimate or runtime limit), or allow unlimited loans.

Fairshare

Fairshare scheduling can now calculate users' dynamic priority based on a decayed runtime by setting the parameter RUN_TIME_DECAY in `lsb.params`. The decay rate used is the same as for cumulative CPU time and historical runtime.

All parameters in the user dynamic priority calculation for fairshare scheduling can now be set at the queue level.

Queue-based fairshare using slot pools can now limit the number of slots available from the slot pool using the parameter MAX_SLOTS_IN_POOL in `lsb.queues`. After job scheduling occurs for each queue, LSF can dispatch jobs to any remaining slots in the pool across queues if the parameter USE_PRIORITY_IN_POOL is set in `lsb.queues`.

Performance and scalability

Several internal improvements have been made affecting LSF performance and scalability. `mbatchd` and `mbschd` memory consumption and communications have been enhanced, and `mbatchd` startup is faster. Time to reconfigure `mbatchd` is reduced. Queries, job submissions and job dispatch have all been optimized.

Some new features improve performance, for example, job packs (`bsub -pack`) improves job submission rate, guarantee SLAs improve resource usage, `bjobs` enhancements improve query performance, and live reconfiguration improves daemon response time.

Licensing

LSF 8.0 uses a consistent core-based licensing model. This replaces the former CPU-based policy. In addition, the notion of license classes is no longer supported. Client licensing has also changed. All client types including floating clients are now supported by the same client license (`lsf_client`).

Additional license keys are required to run Platform Session Scheduler, Platform License Scheduler, or Platform Make.

EGO default

During LSF installation, EGO is now disabled by default.

If you want to use EGO, configure the related installation parameters before installing:

- UNIX (`install.config`):

  ENABLE_EGO

  EGO_DAEMON_CONTROL
- Windows

  ENABLE_EGO (Enable EGO window)

  SERVICETYPE (SBD and RES Control window)

User group new and changed behavior

Default user group

Using the new parameter DEFAULT_USER_GROUP in `lsb.params`, you can now assign a default user group to all jobs submitted without a user group specified.

Enforce user group tree

For ease of administration, you can now reject any `UserGroup` configuration in `lsb.users` that does not follow a tree-like structure. Enable by setting ENFORCE_UG_TREE in `lsb.params`, and the second and subsequent appearance of a user group in GROUP_MEMBER is ignored. This makes it easier to manage inherited rights.

New user group administrator rights

User group administrators can now have `usershares` or `full` rights, allowing them to adjust user group shares or user group membership using the `bconf` command, in addition to controlling jobs within the user group.

The parameter STRICT_UG_CONTROL in `lsb.params` allows you to enable user group administrators for groups with the special member `all`. The parameter also allows you to limit the rights of the user

group administrator: by default, a user group administrator has control over all jobs belonging to group members, but with the parameter set, a user group administrator can only control the jobs that are submitted to that user group (`bsub -G`).

Group administrator output expanded

The commands `bugroup` and `bmgroup` expand the group administrator list to show individual users, even if user groups are the configured administrators.

Functionality enhancements

Job packs

The purpose of this feature is to speed up submission of a large number of jobs. When the feature is enabled, you can submit jobs by submitting a single file containing multiple job requests.

VBScript ELIMs

LSF running on Windows hosts now supports VBScript ELIMs with the extension `.vbs`. Full paths can be up to 230 characters in length.

Host-based rusage

Host-based rusage is now available for parallel jobs (`blaunch` or `pam`/`taskstarter` jobs only), allowing tracking of processes across hosts and providing detailed resource usage information. Host-based rusage includes the runtime `mem`, `swp`, `cputime`, PIDs, and PGIDs, and the finished job `mem`, `swp`, and `cputime`.

When LSF_HPC_EXTENSIONS="HOST_RUSAGE" in `lsf.conf`, runtime host-based rusage is included in output from `bjobs -l`, finished host-based rusage is included in output from `bjobs -l`, `bhist -l`, and `bacct -l`, and in the `lsb.acct` , `lsb.events` and `lsb.stream` files, and the bhosts -l option output displays accurate host-based memory reservation values.

We recommend that you configure cumulative rusage with host rusage (without cumulative rusage, every `blaunch` or `pam`/`taskstarter` command overwrites the rusage value). Set LSF_HPC_EXTENSIONS="CUMULATIVE_RUSAGE HOST_RUSAGE" in `lsf.conf`.

Modify swap limit

The command `bmod` now has options -v and -vn to modify or remove the swap limit of running or pending jobs.

Swap and memory amounts

When LSF_PIM_LINUX_ENHANCE is enabled in `lsf.conf`, support for exact memory usage tracking for shared memory segments reporting has been extended from Linux kernel versions 2.6.25 and above to Linux kernel version 2.6.14 and above.

When EGO_PIM_SWAP_REPORT is also enabled, the reporting of swap amount is also extended.

Preemption time limits

You can now limit the uninterrupted run time before preemption occurs as well as the maximum total accumulated preemption time. The parameters NO_PREEMPT_INTERVAL and MAX_TOTAL_TIME_PREEMPT can be set in `lsb.applications`, `lsb.queues`, or `lsb.params`.

NICE

The `NICE` value can now be defined at the application level in `lsb.applications`, and overrides the queue level value, if set.

Job starter extension

You can now have a job starter that runs on a Windows execution host and has symbols (like && or ||).

Optimize job dependency by job name

Job dependencies specified by job name have an optimized search index created when **LSB_INDEX_BY_JOB="JOBNAME"** in `lsf.conf`.

nVidia GPU Solution

This solution allows the use of nVidia GPUs in a managed manner with LSF. You can submit jobs to LSF, and LSF will make sure that the job is scheduled to machines with the required nVidia GPUs in them.

Usability enhancements

Command output includes year

The time strings displayed in output from the commands `bjobs -l`, `bacct -l`, and `bhist -l`, `-b`, or `-t` can now include the year. Enable by setting **LSB_DISPLAY_YEAR=Y** in `lsf.conf`.

Display execution host

`bsub -K` now displays the execution host in the command output when LSB_SUBK_SHOW_EXEC_HOST is defined in `lsf.conf`.

lspasswd

`lspasswd` can now be run from Linux and UNIX machines.

Display exit reason

Output from `bhist -l` now includes the exit reason for terminated jobs.

Configurable lsrcp fallback

`lsrcp` can now be configured to use `rcp`, `scp`, or a self-defined shell script as a fallback command if copying files through RES fails, using the parameter LSF_REMOTE_COPY_CMD in `lsf.conf`.

Array name element with job dependency

You can now specify an array name element (rather than an entire array) when submitting a job with a dependency. For example: **bsub -w "ended(jobArray[1])" sleep 1000**. The new job is not dispatched until the first element from jobArray has completed.

In locations where *job_id* can be specified, it can be replaced with *job_name*.

Maximum advance reservations

Administrators can set the maximum number of advance reservations any user or user group can make using the ADVRSV_USER_LIMIT in `lsb.params`.

Display exclusive resources

Output from `lshosts` now indicates exclusive resources by prefixing with '!'.

Fault tolerance and error handling enhancements

NIOS

You can now log any NIOS errors to a specific log in a configurable location using LSF_NIOS_ERR_LOGDIR in `lsf.conf`. Applies to Windows only.

Redispatch job when pre-execution scripts fail

If a pre-execution script fails to run successfully on a host, LSF now sensibly reschedules the job on other hosts.

New and changed configuration parameters and environment variables

The following configuration parameters and environment variables are new or changed for LSF Version 8:

lsb.applications

- MAX_TOTAL_TIME_PREEMPT: Sets the accumulated preemption time in minutes after which a job cannot be preempted again (overrides any queue-level and `lsb.params` setting).
- NO_PREEMPT_INTERVAL: Prevents preemption of jobs for the specified number of minutes of uninterrupted run time (overrides any queue-level and `lsb.params` setting).
- NICE: Sets an application-level NICE value, which overrides any queue-level NICE values.

lsb.params

- ADVRSV_USER_LIMIT: Limits the number of advanced reservations users or user groups can make.
- DEFAULT_USER_GROUP: If used, jobs submitted without a user group specified will be associated with the defined default user group.
- ENFORCE_UG_TREE: Enables strict checking for the user group configuration, such that user groups must form tree-like structures.
- JOB_DEP_LAST_SUB: If used, jobs submitted with dependency conditions using a job name that belongs to multiple jobs, evaluate only the most recently submitted job.
- MAX_TOTAL_TIME_PREEMPT: Sets the accumulated preemption time in minutes after which a job cannot be preempted again.
- NO_PREEMPT_INTERVAL: Prevents preemption of jobs for the specified number of minutes of uninterrupted run time.
- PEND_EXTSCHED_REASON: displays a detailed pending reason in `bjobs -p` and `bjobs -l` output.
- PREEXEC_EXCLUDE_HOST_EXIT_VALUES: Specify one or more values (between 1 and 255, but not reserved value 99) that correspond to the exit code your pre-execution scripts exits with in the case of failure. LSF excludes any hosts that attempt to run the pre-execution script and exit with the value specified.
- RUN_TIME_DECAY: Enables decay for runtime at the same rate as the decay set by HIST_HOURS for cumulative CPU time and historical runtime. Used only with fairshare scheduling.

- STRICT_UG_CONTROL: Enables user group administrators for groups containing the special member all. Limits control to the administrator of the specified user group for jobs submitted with user group specified (`bsub -G`).

lsb.hosts

- Some of the configuration in `lsb.hosts` can be modified using the `bconf` command. The updated `lsb.hosts` file is written under the directory set by LSF_LIVE_CONFDIR in `lsf.conf`.

lsb.queues

- Some of the configuration in `lsb.queues` can be modified using the `bconf` command. The updated `lsb.queues` file is written under the directory set by LSF_LIVE_CONFDIR in `lsf.conf`.
- The following fairshare user priority parameters can now be configured at the queue level:
  - RUN_TIME_FACTOR
  - CPU_TIME_FACTOR
  - ENABLE_HIST_RUN_TIME
  - RUN_TIME_DECAY
  - HIST_HOURS
  - FAIRSHARE_ADJUSTMENT_FACTOR
  - RUN_JOB_FACTOR
  - COMMITTED_RUN_TIME_FACTOR
- JOB_STARTER now takes the keyword `preservestarter`, for use with JOB_STARTER_EXTEND in lsf.conf.
- MAX_SLOTS_IN_POOL: Maximum number of job slots available in the slot pool the queue belongs to for queue-based fairshare.
- MAX_TOTAL_TIME_PREEMPT: Sets the accumulated preemption time in minutes after which a job cannot be preempted again.
- NO_PREEMPT_INTERVAL: Prevents preemption of jobs for the specified number of minutes of uninterrupted run time.
- SLA_GUARANTEES_IGNORE: Allows jobs in the queue to use resources guaranteed to SLAs to which the queue does not belong. Use with a guaranteed resource pool in `lsb.resources` and a guarantee SLA in `lsb.serviceclasses`.
- USE_PRIORITY_IN_POOL: Queue-based fairshare only. Enables LSF to dispatch jobs to any remaining slots in a slot pool across queues after job scheduling for each queue is complete.

lsb.resources

- Some of the configuration in `lsb.resources` can be modified using the `bconf` command. The updated `lsb.resources` file is written under the directory set by LSF_LIVE_CONFDIR in `lsf.conf`.
- The `Limit` section has two new consumers: LIC_PROJECTS to enforce limits on specific license projects, and PER_LIC_PROJECT to enforce per-project limits on license projects.
- A new `GuaranteedResourcePool` section configures guaranteed resource pools. A resource pool can be split between several guarantee SLAs (configured in `lsb.serviceclasses`). Parameters for the new section are:
  - NAME (required) - a name for the pool.
  - TYPE (required) - slots or hosts.
  - HOSTS - a list of hosts and host groups in the pool.

- RES_SELECT - a resource requirement string hosts must satisfy.
- DISTRIBUTION (required) - resource distribution among SLAs.
- LOAN_POLICIES - loan policies for the pool.
- DESCRIPTION - description of the pool.
- SLOTS_PER_HOST - maximum number of slots each host can contribute to the pool.

lsb.serviceclasses

- Some of the configuration in `lsb.serviceclasses` can be modified using the `bconf` command. The updated `lsb.serviceclasses` file is written under the directory set by LSF_LIVE_CONFDIR in `lsf.conf`.
- A new resource-based service class with a guarantee goal uses shares in one or more guaranteed resource pools (configured in `lsb.resources`) to guarantee resources. Service classes with a guarantee goal (**GOALS = [GUARANTEE]**) can have the following parameters defined:
  - ACCESS_CONTROL - restricts access to listed users, groups, queues and so on. Can be used with AUTO_ATTACH.
  - AUTO_ATTACH - enables guarantee SLAs to automatically attach to applicable jobs. Used with ACCESS_CONTROL.
  - DESCRIPTION - description of the guarantee SLA.
  - GOALS=[GUARANTEE] - guarantee SLAs do not allow combined goals.
  - NAME - name of the guarantee SLA.

lsb.users

- Some of the configuration in `lsb.users` can be modified using the `bconf` command. The updated `lsb.users` file is written under the directory set by LSF_LIVE_CONFDIR in `lsf.conf`.

lsf.cluster

- Some of the configuration in `lsf.cluster` can be modified using the `bconf` command. The updated `lsf.cluster` file is written under the directory set by LSF_LIVE_CONFDIR in `lsf.conf`.

lsf.conf

- EGO_DEFINE_NCPUS: The default setting has changed from procs to cores.
- JOB_STARTER_EXTEND: For job starters that have symbols (like && or ||) and run on Windows execution hosts. Used in conjunction with JOB_STARTER=preservestarter in `lsb.queues`.
- LSB_DISPLAY_YEAR: Includes the year in the time strings displayed in output from the commands `bjobs -l`, `bacct -l`, and `bhist -l`, `-b`, or `-t`.
- LSB_INDEX_BY_JOB: Enables the creation of a job index of job names for optimized job name searches when specifying job dependencies using job names.
- LSB_MAX_PACK_JOBS: Enables the job packs feature and specifies the maximum number of jobs in one pack.
- LSB_PACK_MESUB: If LSB_PACK_MESUB=N, `mesub` will not be executed for any jobs in the job submission file, even if there are `esubs` configured at the application level (`-a` option of `bsub`), or using LSB_ESUB_METHOD in `lsf.conf`, or through a named `esub` executable under LSF_SERVERDIR.
- LSB_PACK_SKIP_ERROR: If LSB_PACK_SKIP_ERROR=Y, all requests in the job submission file are submitted, even if some of the job submissions fail. Otherwise job submission stops at the first error.

- LSB_SACCT_ONE_UG: Minimizes `mbatchd` startup memory use during fairshare accounting at job submission by only creating share accounts for active users.
- LSB_SUBK_SHOW_EXEC_HOST: Enables display of the execution host in the output of the `bsub -K` command.
- LSF_HPC_EXTENSIONS: When defined as "HOST_RUSAGE", host-based rusage (of jobs created with `blaunch` or `pam`/`taskstarter`) is displayed by `bjobs -l`, `bhist -l`, `bacct -l`, `lsb.events`, `lsb.acct`, and `lsb.stream`. Suggested use is LSF_HPC_EXTENSIONS="CUMULATIVE_RUSAGE HOST_RUSAGE".
- LSF_LIVE_CONFDIR: Specifies a directory for configuration files changed by `bconf` commands. All restart and reconfiguration operations will use files in this directory if they exist.
- LSF_NIOS_ERR_LOGDIR: Specifies a directory for all NIOS errors to be output to. Applies to Windows only.
- LSF_PIM_LINUX_ENHANCE: Support for exact memory usage tracking for shared memory segments reporting has been extended from Linux kernel versions 2.6.25 and above to Linux kernel version 2.6.14 and above

  When EGO_PIM_SWAP_REPORT is also enabled, the reporting of swap amount is also extended.
- LSF_REMOTE_COPY_CMD: If defined, specifies the fallback remote copy command used by `lsrcp` if copying files through RES fails.
- LSF_LSRUN_CWD_USE_TMP=Y: When the current working directory does not exist on an execution host, by default, jobs submitted by `lsrun` and `lsgrun` will fail because of lack of current directory. LSF_LSRUN_CWD_USE_TMP=Y allows the job try to run under `/tmp`. The LSF_LSRUN_CWD_USE_TMP is supported on Unix and Linux.
- LSF_DYNAMIC_HOST_KEEP=Y: Allows LIM to keep dynamic host information in memory when dynamic hosts become unavailable for longer than the period specified by LSF_DYNAMIC_HOST_TIMEOUT. This parameter also disables automatic `mbatchd` reconfiguration triggered by dynamic hosts becoming unavailable for longer than LSF_DYNAMIC_HOST_TIMEOUT. Valid values for LSF_DYNAMIC_HOST_KEEP are Y or N. The default value is N.

New commands

bconf

The new `bconf` command allows LSF administrators and user group administrators to submit live reconfiguration requests, updating configuration settings in active memory without restarting daemons. Updated configuration files are written to the directory set by LSF_LIVE_CONFDIR in `lsf.conf`.

Changed commands, options, and output

The following command options and output are new or changed for LSF Version 8:

bacct

The `-l` option output time string now includes the year when LSB_DISPLAY_YEAR=Y in `lsf.conf`.

The `-l` option displays host-based accounting information for completed jobs when LSF_HPC_EXTENSIONS="HOST_RUSAGE" in `lsf.conf`.

bhist

The `-l`, `-b`, and `-t` option output time strings now include the year when LSB_DISPLAY_YEAR=Y in `lsf.conf`.

The `-l` option displays host-based CPU time used for completed jobs when LSF_HPC_EXTENSIONS="HOST_RUSAGE" in `lsf.conf`.

The `-l` option now includes the exit reason for terminated jobs.

bhosts

The `-l` option output now displays accurate host-based memory reservation values when LSF_HPC_EXTENSIONS="HOST_RUSAGE" in `lsf.conf`.

bjobs

The `-l` option output time string now includes the year when LSB_DISPLAY_YEAR=Y in `lsf.conf`.

The `-l` option output now includes the host-based resource usage when LSF_HPC_EXTENSIONS="HOST_RUSAGE" in `lsf.conf`.

If a job has been submitted with an SLA (using `bsub -sla`) or automatically attached to a guarantee SLA, the `-l` option shows the SLA.

blimits

New option `-Lp` displays license projects on which limits are enforced. This information is not shown by default.

bmgroup

Now expands the group administrator list to show individual users, even if user groups are the configured administrators.

bmod

The new `bmod` options `-v` and `-vn` modify or remove the swap limit of a running or pending job.

`bmod -Gn` now moves the job to the default user group, if DEFAULT_USER_GROUP in `lsb.params` is configured.

You cannot modify a job such that it no longer satisfies the assigned guarantee SLA. Jobs auto-attached to guarantee SLAs and modified before they run re-attach to another SLA as required, but running jobs must continue to satisfy the auto-attached SLA.

bqueues

The `-l`and`-r` options show the new parameters such as SLA_GUARANTEES_IGNORE, MAX_SLOTS_IN_POOL, and USE_PRIORITY_IN_POOL, if defined, and the queue-level fairshare factors, if defined.

bresources

The new `-g` option displays information about configured guaranteed resource pools:

- POOL_NAME - name of guaranteed resource pool.
- TYPE - hosts or slots.
- STATUS - Whether guarantee is being met. Possible values are `ok`, `unknown`, `overcommitted` (more resources guaranteed than in pool), and `close_loans` (new loans suspended due to pending demand).
- TOTAL - number of resources included in guaranteed resource pool.

- FREE - number of unused resources in guaranteed resource pool.
- GUAR CONFIG - configured number of guaranteed resources.
- GUAR UNUSED - unused number of guaranteed resources.

The new `-l` option (used with `-g`) displays long format detailed information about guaranteed resource pools with the following additional fields:

- GUARANTEED RESOURCE POOL - name and description of guaranteed resource pool.
- DISTRIBUTION - configured distribution of guarantee among SLAs.
- LOAN_POLICIES - configured loan policies.
- HOSTS - configured host list.
- RESOURCE SUMMARY:
  - SLA - Name of each SLA guarantee made from the guaranteed resource pool.
  - GUARANTEED - number of resources in the pool guaranteed to the SLA.
  - USED - number of resources in the pool currently in use by the SLA.

The new `-m` option (used with `-g` and `-l`) displays the hosts currently in guaranteed resource pools. This includes configured hosts in the states `ok`, `closed_Busy`, `closed_Excl`, `closed_cu_Excl`, and `closed_Full`.

bsla

Now displays information about resource-based guarantee SLAs as well as time-based velocity, deadline, and throughput SLAs.

bsub

The `-K` option now displays the execution host in the command output when LSB_SUBK_SHOW_EXEC_HOST is defined in `lsf.conf`.

The new -pack *job_submission_file* option allows submission of jobs from a file. The job packs feature must be enabled.

bswitch

When switching a job that has been auto-attached to a guarantee SLA, a running job is only switched if the new queue satisfies the SLA, while a job that has not started is switched and the auto-attachment changed if required.

bugroup

Output from `bugroup` now shows the user group administrator rights. Group administrators are expanded to show individual users, even if user groups are the configured administrators.

lsadmin

The `-s` option for the `lsadmin lsflic` command is no longer supported.

lshosts

Output from `lshosts` now indicates exclusive resources with the prefix '!'.

The `-l` option no longer displays the LICENSES_NEEDED field.

lsmake

Platform Make has improved performance and efficiency, and now supports large make files.

The new `-a` and `-x` options help you to avoid errors from file system latency.

Use `-x` to automatically rerun a command that has failed, and specify how many times to retry the command. The interval between attempts automatically increases each time.

Use `-a` when you have dependent targets that may run on different hosts, and you want to allow time for the shared file system to synchronize client and server. Specify longer times for slower file systems. Used together with `-x`, the new `-a` option also affects the timing of retry attempts, so the interval between attempts is longer for slower file systems.

The new `-y` option displays summary information after the job is done.

The new `-u` option generates a data file tracking the number of tasks running over time.

The `-m` option syntax has been improved, so you can simply specify the number of cores (slots) after the host name when you want to use multiple cores on a host.

The `-j` option now considers the number of cores on multi-core hosts.

Platform Make now supports the following standard LSF debug options: LSF_CMD_LOGDIR, LSF_CMD_LOG_MASK, LSF_DEBUG_CMD, LSF_TIME_CMD, LSF_NIOS_DEBUG.

lspasswd

`lspasswd` can now be run from Linux and UNIX machines.

By default, LSF uses host type `nt` to search for Windows servers for user authentication. If you have configured a different LSF host type for your Windows server hosts, use the `-t` option.

lsrcp

`lsrcp` can now be configured to use `rcp`, `scp`, or a self-defined shell script as a fallback command for res-copy, using parameter LSF_REMOTE_COPY_CMD in `lsf.conf`.

New configuration files

No configuration files are new for LSF Version 8.

New and changed accounting and job event fields

liveconf.hist

All changes to configuration files made by the new `bconf` command are recorded in the `liveconf.hist` file located under `$LSB_SHAREDIR`. The `bconf hist` command queries this file.

lsb.acct

The JOB_FINISH record now includes new host-based rusage fields when LSF_HPC_EXTENSIONS="HOST_RUSAGE" in `lsf.conf`.

lsb.events

The JOB_STATUS record for completed jobs now contains new host-based rusage fields when LSF_HPC_EXTENSIONS="HOST_RUSAGE" in `lsf.conf`.

Bugs fixed since September 2009 (Platform LSF 7 Update 6)

Bugs fixed in the June 2011 release (LSF Version 8.0.1) since the September 2009 update (LSF Version 7 Update 6) until May 17 2011 are listed in the document *Fixed Bugs for Platform LSF Version 8.0*.

Known Issues

- `bmod` cannot change the memory requirement for a running job if a MEM general resource limit is defined for the user in `lsb.resources`.
- Application checkpointing is not supported on 64-bit Windows 7.
- MacOS X is supported only as LSF slave hosts. MacOS X hosts cannot be LSF master or master candidate hosts.
- The order of sections and the syntax used in the configuration file templates must be maintained in all configuration files used with live reconfiguration. If configuration files use irregular syntax, the result of using the `bconf` command may result in undesirable behavior.

  For example, if a `User` section is defined before a `UserGroup` section in `lsb.users`, when using `bconf`, changes to user group job limits are inserted in the `User` section before the `UserGroup` definition. To avoid this problem, do not change the order of the `User` and `UserGroup` sections in the `lsb.users` template. Make sure that the `UserGroup` section remains before the `User` section.

  Note that the `badmin ckconfig` command does not detect all syntax errors.
- For both host partition and queue-level fairshare, if you specify user share assignments with the keyword "`others`", there can be a problem using the `bconf rmmember` command to modify the share assignment. The "`others`" keyword is not intended to be used by itself, but you can use the `bconf rmmember` command to define an assignment this way, for example:

  ```
  FAIRSHARE=USER_SHARES[others,10]
  ```

  After you run `bconf rmmember`, the bad syntax is temporarily interpreted as equal share configuration, so fairshare continues to work, as if the syntax was:

  ```
  FAIRSHARE=USER_SHARES[default,1]
  ```

  After you restart or reconfigure `mbatchd`, the entire line is ignored and fairshare is disabled, because the syntax is illogical.

  If equal shares is what you want, you must modify the file manually.
- If you use the `bconf` command to modify membership of a host group, a problem can occur, but only if your host group excludes one or more hosts.

  Attention:

  The problem can only occur if you have a host group that contains at least one excluded host, AND this host group contains at least two other host groups, AND those two other host groups have at least one host in common.

  This example shows the problem.

  1. Define groups so `Group1` and `Group2` have a host in common, and `Group3` specifies an excluded host:

     ```
     Begin HostGroup
     ```

```
GROUP_NAME  GROUP_MEMBER
Group1     (hostA hostB hostC hostX)
Group2     (hostA hostQ)
Group3     (Group1 Group2 ~hostX)
End HostGroup
```

At this point, `Group3` contains four hosts {`hostA` `hostB` `hostC` `hostQ`}.

2. The problem will occur if you use the `bconf` command to remove `Group2` from `Group3`.

```
Group3     (Group1 ~hostX)
```

After you run `bconf`, all the hosts in `Group2` are removed from `Group3`, including `HostA` (temporarily).

At this point, `Group3` contains two hosts {`hostB` `hostC`}.

3. After you restart or reconfigure `mbatchd`, `Group3` includes `HostA` again, because `HostA` belongs to `Group1`.

At this point, `Group3` contains three hosts {`hostA` `hostB` `hostC`}.

If `Group3` did not specifically exclude `hostX` (or any other host), the system would use a different method to determine group membership during live reconfiguration, and the problem would not occur at all.

- If you configure `lsb.resources` and define HOSTS in a limit with host type or model, do not use the `bconf` command to modify the HOSTS membership, or the result may be inconsistent with `badmin reconfig`.
- When SLOTS, HOSTS, or USERS is defined as a limit in `lsb.resources`, you cannot use the `bconf` command to change the limit to SLOTS_PER_PROCESSOR, PER_HOST, or PER_USER. You need to delete and recreate the limit.
- Administrators must move files from LSF_LIVE_CONFDIR to LSF_CONFDIR manually before upgrading LSF, or applying patches to LSF. This is limitation of the LSF installer.
- Platform Analytics 7.6 is compatible with LSF 8.0, but it was developed for use with LSF 7.0, and for full data collection functionality it should be configured to work with the LSF version 7.0 library (not 8.0). However, the Platform Analytics 7.6 node installer cannot detect LSF version 8.0 or later, and by default it configures Platform Analytics to work with LSF version 6.2.

  After installing Platform Analytics, take these steps to fix the configuration:

  1. Stop the data collection.
  2. Edit the file:

     `$PA_ROOT/conf/perf.conf`
  3. Configure the LSF_VERSION parameter to 7.0, as shown (not 8.0 or 6.2):

     LSF_VERSION=7.0
  4. Save your changes, source the environment, and start the data collection again.

Limitations

- You cannot use the `bconf` command to modify an LSF user group if a UNIX user group exists with the same name. The workaround is to modify configuration manually.
- You cannot use the `bconf` command to modify Windows users or nonexistent users in a mixed cluster. The workaround is to modify users manually.
- Terminal Services jobs, submitted with the `tssub` command, cannot start on an execution server host running a Windows desktop operating system (XP, Vista, or Windows 7).

Download the Platform LSF Version 8.0.1 Distribution Packages

Download the LSF distribution packages two ways:

- Through FTP at `ftp.platform.com`
- Through the World Wide Web at `my.platform.com`

Download Platform LSF from my.platform.com

You must provide your Customer Support Number and register a user name and password on `my.platform.com` to download LSF.

To register at `my.platform.com`, click New User? and complete the registration form. If you do not know your Customer Support Number or cannot log in to `my.platform.com`, send email to *support@platform.com*.

1. Navigate to *http://my.platform.com*.
2. Choose Products > Platform LSF Family > Platform LSF > LSF 8.0.1.
3. Raise the Download tab, and select the Updates, Packages, and Documentation you wish to download.
4. Log out of `my.platform.com`.

Download Platform LSF through FTP

Access to the Platform FTP site is controlled by login name and password. If you cannot access the distribution files for download, send email to *support@platform.com*.

1. Log on to the LSF file server.
2. Change to the directory where you want to download the LSF distribution files. Make sure that you have write access to the directory. For example:

   ```
   # cd /usr/share/lsf/tarfiles
   ```
3. FTP to the Platform FTP site:

   ```
   # ftp ftp.platform.com
   ```
4. Provide the login user ID and password provided by Platform.
5. Change to the directory for the LSF Version 8 release:

   ```
   ftp> cd /distrib/8.0/platform_lsf
   ```
6. Set file transfer mode to binary:

   ```
   ftp> binary
   ```
7. For LSF on UNIX and Linux, get the installation distribution file.

   ```
   ftp> get lsf8.0.1_lsfinstall.tar.Z
   ```

   Tip:

   Before installing LSF on your UNIX and Linux hosts, you must uncompress and extract lsf8.0.1_lsfinstall.tar.Z to the same directory where you download the LSF product distribution tar files.
8. Get the distribution packages for the products you want to install on the supported platforms you need. For example:

- For the Linux 2.6 (glibc 2.3) 64-bit version of LSF Version 8.0.1:

```
ftp> get lsf8.0.1_linux2.6-glibc2.3-x86_64.tar.Z
```

Tip:

Put the LSF distribution files in the same directory as the installation tar files. *Do not* uncompress and extract the distribution files.

- For 32-bit LSF Version 8 on Windows:

```
ftp> get lsf8.0.1_win32.msi
```

9. Download the Platform LSF Version 8 documentation from `/distrib/8.0/platform_lsf/docs/`.

```
ftp> get lsf8_documentation.zip
ftp> get lsf8_documentation.tar.Z
```

Tip:

After installing LSF, you should extract the Platform LSF Version 8 documentation files to LSF_TOP/docs/lsf. Navigate in your browser to LSF_TOP/docs/lsf/index.html to access the LSF Version 8 documentation.

10. Exit FTP.

```
ftp> quit
```

Install Platform LSF Version 8.0.1

Installing Platform LSF involves the following steps:

1. Get a demo license (`license.dat` fie).
2. Run the installation programs.

Get a Platform LSF demo license

Before installing Platform LSF Version 8, you must get a demo license key.

Contact *license@platform.com* to get a demo license.

Put the demo license file `license.dat` in the same directory where you downloaded the Platform LSF product distribution tar files.

Run the UNIX and Linux installation

Use the `lsfinstall` installation program to install a new LSF Version 8 cluster, or upgrade from an earlier LSF version.

See *Installing Platform LSF on UNIX and Linux* for new cluster installation steps.

See the *Platform LSF Command Reference* for detailed information about `lsfinstall` and its options.

Important:

DO NOT use the UNIX and Linux upgrade steps to migrate an existing LSF 7 cluster or LSF Version 7 Update 1 cluster to LSF Version 8. Follow the manual steps in the document *Migrating to Platform LSF Version 8 on*

UNIX and Linux to migrate an existing LSF 7 Update 1 cluster to LSF Version 8 on UNIX and Linux.

Run the Windows installation

Platform LSF on Windows 2003, Windows 2008, Windows 7, and Windows 2008R2 is distributed in the following packages:

- `lsf8.0.1_win32.msi`
- `lsf8.0.1_win-x64.msi`

Platform LSF is not supported on Windows for IA64 (`lsf8.0.1_win-ia64.msi`).

See *Installing Platform LSF on Windows* for new cluster installation steps.

To migrate your existing LSF Version 7 cluster on Windows to LSF Version 8, you must follow the manual steps in the document *Migrating Platform LSF Version 7 to Platform LSF Version 8 on Windows* (`lsf_migrate_windows_to_8.pdf`).

Platform Application Center

Platform Application Center provides a web-based user interface for job submission, job and LSF host monitoring, and reporting. Additional functionality is licensed in Platform Application Center for managing job flows.

You must install Platform LSF before installing Platform Application Center. See *Installing Platform Application Center* for installation and configuration steps.

Note:

The Platform Application Center is installed separately from LSF, and is no longer installed at the same time as LSF.

Install Platform License Scheduler

See *Using Platform License Scheduler* for installation and configuration steps.

Learn About Platform LSF Version 8

Information about Platform LSF is available from the following sources:

- World Wide Web and FTP
- Platform LSF documentation
- Platform training

World Wide Web and FTP

Information about Platform LSF Version 8 is available in the LSF area of the Platform FTP site (`ftp.platform.com/distrib/8.0/`).

The latest information about all supported releases of Platform LSF is available on the Platform Web site at *www.platform.com*.

If you have problems accessing the Platform web site or the Platform FTP site, send email to *support@platform.com*.

my.platform.com

`my.platform.com`—Your one-stop-shop for information, forums, e-support, documentation and release information. `my.platform.com` provides a single source of information and access to new products and releases from Platform Computing.

On the Platform LSF Family product page of `my.platform.com`, you can download software, patches, updates and documentation. See what's new in Platform LSF Version 8.0.1, check the system requirements for Platform LSF, or browse and search the latest documentation updates through the Platform LSF Documentation page.

Platform LSF documentation

The Platform LSF Documentation page is your entry point for all LSF documentation.

Get the latest LSF documentation from `my.platform.com`. Extract the LSF documentation distribution file to the directory `LSF_TOP/docs/lsf`.

Note:

Current Platform LSF 8.0 documentation also applies to Platform LSF Version 8.0.1. It contains some enhancements and corrections to the documentation released with Platform LSF Version 8.0. in January 2011.

Platform training

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

Customized on-site course delivery is also available.

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

Get Technical Support

Contact Platform

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

Web Portal eSupport

You can take advantage of our Web-based self-support available 24 hours per day, 7 days a week ("24x7") by visiting http://my.platform.com. The Platform eSupport and Support Knowledgebase site enables you to search for solutions, submit your support request, update your request, enquire about your request, as well as download product manuals, binaries, and patches.

Email

support@platform.com

Get patch updates and other notifications

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

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

We'd like to hear from you

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

Email

doc@platform.com

Mail

Information Development
Platform Computing Inc.
3760 14th Avenue Markham
Ontario Canada L3R 3T7

Be sure to tell us:

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

Copyright

Document redistribution policy

Internal redistribution

You may only redistribute this document internally within your organization (for example, on an intranet) provided that you continue to check the Platform Web site for updates and update your version of the documentation. You may not make it available to your organization over the Internet.

Trademarks

LSF is a registered trademark of Platform Computing Corporation in the United States and in other jurisdictions.

POWERING HIGH PERFORMANCE, PLATFORM COMPUTING, PLATFORM SYMPHONY, PLATFORM JOBSCHEDULER, and the PLATFORM and PLATFORM LSF logos are trademarks of Platform Computing Corporation in the United States and in other jurisdictions.

UNIX is a registered trademark of The Open Group in the United States and in other jurisdictions.

Linux is the registered trademark of Linus Torvalds in the U.S. and other countries.

Microsoft is either a registered trademark or a trademark of Microsoft Corporation in the United States and/or other countries.

Windows is a registered trademark of Microsoft Corporation in the United States and other countries.

FlexNet is a registered trademarks or trademarks of Flexera Software Corporation in the United States of America and/or other countries.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates.

Intel, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

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

Third Party License Agreements

www.platform.com/legal-notices/third-party-license-agreements