Using Platform License Scheduler

Platform License Scheduler
Version 8.0
June 2011

Platform Computing

Contents

CHAPTER

1

Introduction

About Platform License Scheduler

Applying policies to how licenses are shared can have significant benefits both in terms of productivity and cost savings.

Share licenses more easily

Platform License Scheduler makes it easy to share licenses between project teams and departments within the same design center or around the globe. With tools to allocate and monitor licenses, license owners can share licenses not in use, while still ensuring immediate access to licenses when needed. With more effective sharing, all users perceive a larger pool of licenses.

Ensure appropriate license allocation

Platform License Scheduler enables flexible, hierarchical sharing policies that reflect the needs of the business. During quiet periods, when licenses are not in contention, licenses can be allocated to anyone who needs them keeping utilization and throughput high. During busy periods, the supply of licenses can be allocated based on policy toward the most time or revenue critical projects.

Improve service levels and productivity

By ensuring access to a minimum share of licenses, and enabling allocation to flex between clusters based on need, licenses are more readily available and jobs are less likely to pend in queues awaiting license resources. This translates into reduced wait times and better productivity and contributes to a faster more efficient design environment.

Reduce or avoid cost

By being able to allocate scarce licenses to the most critical projects, and by being able to analyze license usage in the context of cluster resources, users and projects, planners are better able to find and remove bottlenecks, making their existing licenses more productive. With better visibility to how licenses are being used, they can plan license requirements more effectively ultimately helping to contain costs and boost productivity.

Platform License Scheduler controls the software license sharing in your organization. Platform License Scheduler works with FlexNet™ products to control and monitor license usage.

Glossary

blcollect

The LSF License Scheduler daemon that queries FlexNet licensing software for license usage. `blcollect` collects information from `lmstat`.

You can spread the load of license collection by running the license information collection daemon on multiple UNIX hosts.

Also called the collector.

bld

The LSF License Scheduler batch daemon.

cluster mode

License tokens are allocated to clusters by License Scheduler, and job scheduling within each cluster is managed by the local mbatchd. Not available before Platform License Scheduler version 8.0.

Each license feature can use either cluster mode or project mode, but not both.

lmgrd

The main FlexNet licensing daemon. Usually grouped into service domains inside License Scheduler.

project mode

License tokens are allocated to projects by License Scheduler, and job scheduling for license projects takes place across clusters following the license distribution policy configured for each project. Corresponds to Platform License Scheduler version 7.0.5 and earlier.

Each license feature can use either cluster mode or project mode, but not both.

service domain

A group of one or more FlexNet license servers.

You configure the service domain with the license server names and port numbers that serve licenses to a network.

taskman job

A job that is run by the LSF Task Manager (`taskman`) tool outside of LSF, but is scheduled by License Scheduler.

token

One license token represents one actual license, and is used by Platform License Scheduler to track license use and determine which job to dispatch next.

Platform License Scheduler manages license tokens instead of controlling the licenses directly. After reserving license tokens, jobs are dispatched, then the application that needs the license is started. The number of tokens available from LSF corresponds to the number of licenses available from FlexNet, so if a token is not available, the job is not dispatched.

Architecture

Platform License Scheduler manages license tokens instead of controlling the licenses directly. Using Platform License Scheduler, jobs receive a license token before starting the application. The number of tokens available from LSF corresponds to the number of licenses available from FlexNet, so if a token is not available, the job does not start. In this way, the number of licenses requested by running jobs does not exceed the number of available licenses.

When a job starts, the application is not aware of LSF License Scheduler. The application checks out licenses from FlexNet in the usual manner.

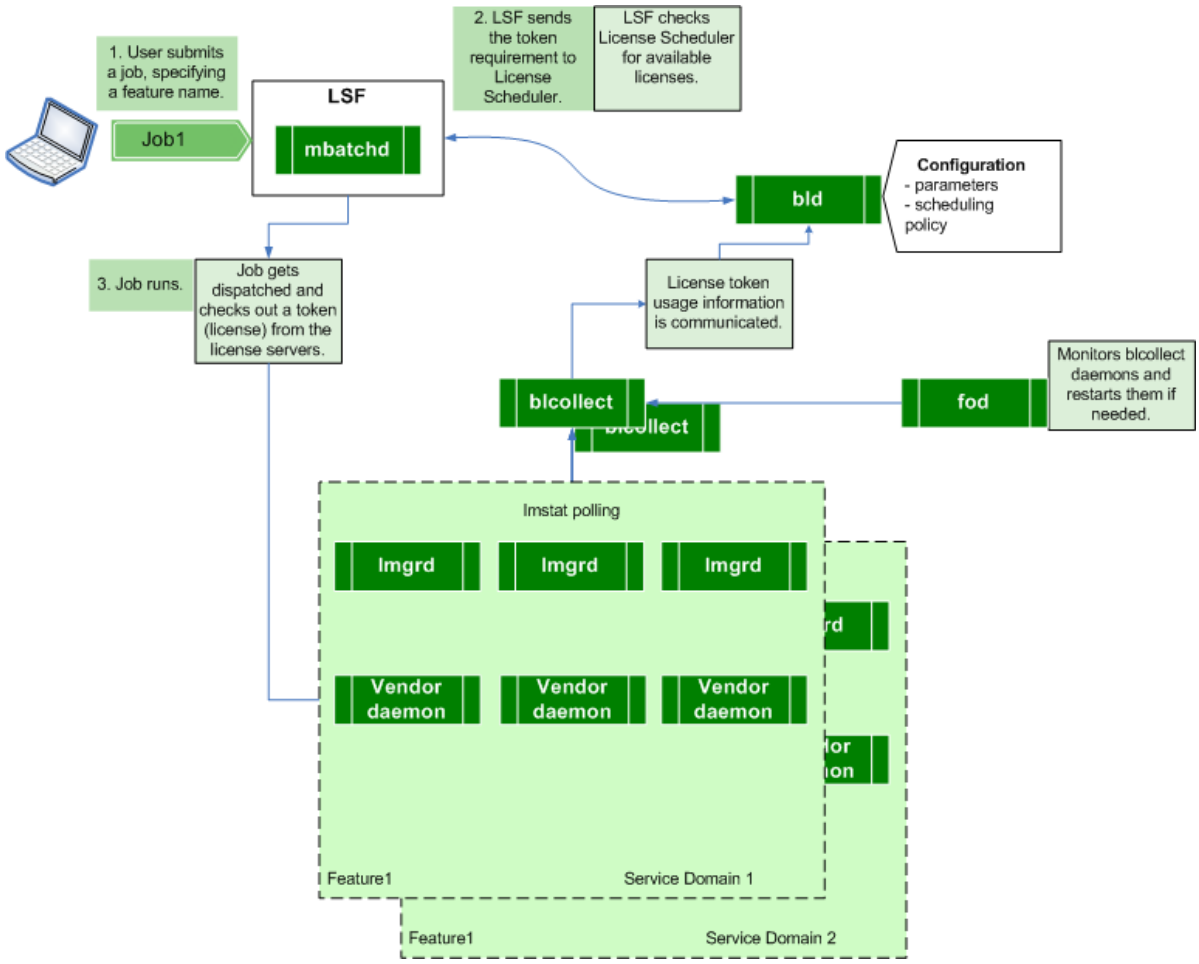


Figure 1: Daemon interaction

How scheduling policies work

With Platform License Scheduler, LSF gathers information about the licensing requirements of pending jobs to efficiently distribute available licenses. Other LSF scheduling policies are independent from Platform License Scheduler policies.

When starting a job, the basic LSF scheduling comes first. Platform License Scheduler has no influence on job scheduling priority. Jobs are considered for dispatch according to the prioritization policies configured in each cluster.

For example, a job must have a candidate LSF host on which to start before the License Scheduler fairshare policy (for the license project this job belongs to) will apply.

Other LSF fairshare policies are based on CPU time, run time, and usage. If LSF fairshare scheduling is configured, LSF determines which user or queue has the highest priority, then considers other resources. In this way, the other LSF fairshare policies have priority over License Scheduler.

When the mbatchd is offline

When a cluster is running, the mbatchd maintains a TCP connection to bld. When the cluster is disconnected (such as when the cluster goes down or is restarted) the bld removes all information about jobs in the cluster. License Scheduler considers licenses checked out by jobs in a disconnected cluster to be non-LSF use of licenses.

When `mbatchd` comes back online, the bld immediately receives updated information about the number of tokens currently distributed to the cluster.

When the bld is offline

If the `mbatchd` loses the connection with the `bld`, the `mbatchd` cannot get `bld`'s token distribution decisions to update its own.

However, because the `mbatchd` logs token status every minute in *`$LSF_TOP`*`/work/data/`*`featureName.ServiceDomainName`*`.dat` file, if the connection is lost, the `mbatchd` uses the last logged information to schedule jobs.

```
f3.LanServer1.dat
# f3 LanServer1 3 2
# p1 50 p2 50
  12/3         14:20:38        2    0    2    0         1    0    1    0
  12/3         14:21:39        2    0    2    0         1    0    1    0
  12/3         14:22:40        3    3    0    0         0    0    0    0
  12/3         14:23:41        3    3    0    0         0    0    0    0
  12/3         14:24:42        1    0    1    0         2    0    2    0
  12/3         14:25:43        1    0    1    0         2    0    2    0
  12/3         14:26:44        1    0    1    0         2    0    2    0
  12/3         14:27:55        1    0    1    0         2    0    2    0
```

f3 on LanServer1 has 3 tokens and 2 projects. Projects p1 and p2 share licenses 50:50.

At 14:27:55, the `bld` dispatched 1 token to p1, which has 0 in use, 1 free, 0 reserve. At the same time, the `bld` dispatched 2 tokens to p2, which has 0 in use, 2 free, and 0 reserve.

The `mbatchd` continues to schedule jobs based on the token distribution logged at 14:27:55 until the connection with the `bld` is re-established.

CHAPTER 2

Installing and Starting Platform License Scheduler

Install Platform License Scheduler

1. Perform the pre-installation steps.
2. Choose an installation plan:
   - UNIX: License Scheduler manages licenses for jobs that run through LSF and through applications other than LSF.
   - Windows, in a mixed cluster:

     A Platform Licenser Scheduler installation requires UNIX hosts to run the `bld`. Windows hosts in a mixed cluster can run License Scheduler commands.

     When you have License Scheduler UNIX machines working with LSF, run License Scheduler on Windows machines as well.

Before you install

LSF must be installed and running before installing Platform License Scheduler.

You must get a Platform License Scheduler license before installation.

1. Get an LSF License Scheduler license from Platform Computing:
   a) Send the host name and host identifier of the license server host to Platform at license@platform.com or to your LSF vendor.
   b) Check the `LSF_LICENSE_FILE` parameter in `lsf.conf` to locate the LSF license file.
   c) Add the LSF License Scheduler (`lsf_license_scheduler`) feature line to your existing LSF license file. For example:

   ```
   FEATURE lsf_license_scheduler lsf_ld 8.000 1-jun-0003 1 3C0733892E1683812345
   "Platform"
   ```

   d) For a permanent license, restart the LSF `lmgrd`.
2. Log on to any LSF host as `root` and use `lsid` to make sure the cluster is running. If you see the message "Cannot open `lsf.conf` file", the `$LSF_ENVDIR` environment variable may not be set correctly.

   To set your LSF environment:

   - For `csh` or `tcsh`:

     `%` **source LSF_TOP/conf/cshrc.lsf**
   - For `sh`, `ksh`, or `bash`:

     `$` **. LSF_TOP/conf/profile.lsf**

What the Platform License Scheduler setup script does

- Finds the appropriate `lsf.conf` for the running cluster.
- Copies the Platform License Scheduler files to your LSF directories:
  - $LSF_ENVDIR:
    - `lsf.licensescheduler`
    - `ls.users`

- $LSF_SERVERDIR:
  - `bld`
  - `blcollect`
  - `globauth`
  - `esub.ls_auth`
- $LSF_BINDIR:
  - `blstat`
  - `blcstat`
  - `blusers`
  - `blinfo`
  - `bladmin`
  - `blstartup`
  - `blhosts`
  - `blkill`
  - `bltasks`
  - `blparams`
  - `taskman`
- $LSF_LIBDIR:
  - `libglb.a`
  - `libglb.so`
  - liblic.so
- $LSF_MANDIR: various man pages

- Finds the appropriate `lsf.cluster.`*cluster_name* file for the running cluster.
- Creates the following additional directories:
  - `$LSB_SHAREDIR/`*cluster_name*`/db`
  - `$LSB_SHAREDIR/`*cluster_name*`/data`
- Sets your Platform License Scheduler administrators list in the `lsf.licensescheduler` file.
- Configures LSF to use License Scheduler.

Install Platform License Scheduler with Platform LSF (UNIX)

You must have write access to the LSF_TOP directories.

1. Log on as `root` to the installation file server host.
2. Download, uncompress, and extract the Platform License Scheduler packages for the platforms you need from the directory `distrib/8.0/platform_license_scheduler`.

   For example, for x86 64 bit systems running Linux Kernel 2.6.x and compiled with glibc 2.3.x:

   ```
   ftp> get lsf8.0_licsched_linux2.6-glibc2.3-x86_64.tar.Z
   ```

   Make sure that you download the License Scheduler distribution files to the same directory where you downloaded the LSF product distribution tar files.
3. Extract the distribution file.

   For example:

```
# zcat lsf8.0_licsched_linux2.6-glibc2.3-x86_64.tar.Z | tar xvf -
```

4. Change to the extracted distribution directory.

   For example:

   ```
   # cd lsf8.0_licsched_linux2.6-glibc2.3-x86_64
   ```

5. Edit `./setup.config` to specify the installation variables you want.

   Uncomment the options you want in the template file, and replace the example values with your own settings.

   Tip:

   The sample values in the `setup.config` template file are examples only. They are *not* default installation values.

6. Run the `setup` script as `root`:

   ```
   # ./setup
   ```

7. Enter `y` (yes) to confirm that the path to `lsf.conf` is correct.

   To enter a path to a different `lsf.conf`, type `n` (no) and specify the full path to the `lsf.conf` file you want to use.

8. Enter `y` to confirm that the path to `lsf.cluster.`*cluster_name* is correct.

   To enter a path to a different `lsf.cluster.`*cluster_name* file, type `n` (no) and specify the full path to the `lsf.cluster.`*cluster_name* file you want to use.

9. Enter `y` to confirm that you want to use the LSF Administrators list for License Scheduler with LSF.

   To enter a different list of administrators for License Scheduler, enter a space-separated list of administrator user names. You can change your License Scheduler administrators list later, if desired.

Install Platform License Scheduler on Windows

You can install License Scheduler on Windows hosts when your cluster includes both Windows and UNIX hosts.

The Platform License Scheduler Windows Client package includes:

- README
- Commands:
  - blstat.exe
  - blcstat.exe
  - blinfo.exe
  - blusers.exe
  - bladmin.exe
  - blhosts.exe
  - blkill.exe
  - bltasks.exe
  - blparams.exe
  - taskman.exe
- lsf.licensescheduler: Platform License Scheduler configuration file
- lsf.conf: LSF configuration file

Install Platform License Scheduler with Platform LSF (Windows)

You must already have LSF installed on all Windows hosts you intend to install License Scheduler on.

This installation option means that License Scheduler manages licenses for jobs submitted through LSF and through any other applications.

Install License Scheduler on Windows hosts only when your LSF cluster includes both UNIX and Windows hosts.

1. Download the License Scheduler Client for Windows package from the FTP site.
2. Copy all commands to $LSF_BINDIR (the `bin` subdirectory in your LSF installation directory) on your Windows hosts.
3. Copy `lsf.licensescheduler` to $LSF_ENVDIR.
4. Edit `lsf.licensescheduler` to suit your Platform License Scheduler Master host configuration.

Troubleshoot

1. If you receive the following message, configure your Windows host name and IP address in the `/etc/hosts` file on the master host:

   ```
   Failed in an LSF library call: Failed in sending/receiving a message: error 0: The operation completed successfully.
   ```

2. To enable the `blhosts` command, make sure your Windows host can resolve the master host IP address correctly.

Start Platform License Scheduler

You can configure LSF to start the License Scheduler daemon (`bld`) on the License Scheduler host as well as on candidate License Scheduler hosts that can take over license distribution in the case of a network failure. The LSF LIM daemon starts `bld` automatically.

1. Log on as the primary LSF administrator.
2. Set your LSF environment:
   - For `csh` or `tcsh`:

     `% source LSF_TOP/conf/cshrc.lsf`

   - For `sh`, `ksh`, or `bash`:

     `$ . LSF_TOP/conf/profile.lsf`
3. In `LSF_CONFDIR/lsf.conf`, specify a space-separated list of hosts for the LSF_LIC_SCHED_HOSTS parameters:

   LSF_LIC_SCHED_HOSTS="hostname_1 hostname_2 ... hostname_n"

   Where:

   hostname_1, *hostname_2*, ..., *hostname_n* are hosts on which the LSF LIM daemon starts the Platform License Scheduler daemon. The order of the host names is ignored.

   Note:

   Set the LSF_LIC_SCHED_HOSTS parameter to the same list of candidate hosts you used in the `lsf.licensescheduler` HOSTS parameter. The LSF_LIC_SCHED_HOSTS parameter is not used in any other function.
4. Run `lsadmin reconfig` to reconfigure the LIM.
5. Use `ps -ef` to make sure that `bld` is running on the candidate hosts.
6. Run `badmin mbdrestart` to restart `mbatchd`.
7. If you specified a LIC_COLLECTOR name in your service domains, start each license collector manually:

   blcollect -m "*host_list*" **-p** *lic_scheduler_port* **-c** *lic_collector_name*

   Where:

   - *host_list*

     Specifies a space-separated list of License Scheduler candidate hosts to which license information is sent. Use fully qualified host names.
   - *lic_scheduler_port*

     Corresponds to the License Scheduler listening port, which is set in `lsf.licensescheduler`.
   - *lic_collector_name*

     Specifies the name of the license collector you set for `LIC_COLLECTOR` in the service domain section of `lsf.licensescheduler`.

     For example:

     blcollect -m "hostD.designcenter_b.com hostA.designcenter_a.com" -p 9581 -c CenterB

A file named `collectors/CenterB` is created in your `LSF_WORKDIR`.

Note:

If you do not specify a license collector name in a License Scheduler service domain, the master bld host starts a default blcollect.

Platform LSF parameters in Platform License Scheduler

Parameters in `lsf.conf` that start with LSF_LIC_SCHED are relevant to both LSF and License Scheduler:

- LSF_LIC_SCHED_HOSTS: LIM starts the License Scheduler daemon (`bld`) on candidate License Scheduler hosts.

Caution:

You cannot use LSF_LIC_SCHED_HOSTS if your cluster was installed with UNIFORM_DIRECTORY_PATH or UNIFORM_DIRECTORY_PATH_EGO. *Do not* set UNIFORM_DIRECTORY_PATH or UNIFORM_DIRECTORY_PATH_EGO for new or upgrade installations. They are for backwards compatibility only.

- LSF_LIC_SCHED_PREEMPT_REQUEUE : Requeues a job whose license is preempted by License Scheduler. The job will be killed and requeued instead of suspended.
- LSF_LIC_SCHED_PREEMPT_SLOT_RELEASE : Releases the slot of a job that is suspended when the its license is preempted by License Scheduler.
- LSF_LIC_SCHED_PREEMPT_STOP : Uses 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.
- LSF_LIC_SCHED_STRICT_PROJECT_NAME: Enforces strict checking of the License Scheduler project name upon job submission. If the project named is misspelled (case sensitivity applies), the job is rejected.

Platform LSF parameters used by Platform License Scheduler

- LSB_SHAREDIR: Directory where the job history and accounting logs are kept for each cluster
- LSF_LICENSE_FILE: One or more demo or FlexNet-based permanent license files used by LSF
- LSF_LICENSE_ACCT_PATH : Location for the license accounting files, including the license accounting files for LSF Family products
- LSF_LOG_MASK : Logging level of error messages for LSF daemons
- LSF_LOGDIR: LSF system log file directory

About submitting jobs

When you submit an LSF job, you must reserve the license using the resource requirement usage section (`bsub -R "rusage..."` option).

Tip:

You cannot successfully reserve a license using `bsub -R "select"`.

- Specify the license token name (same as specifying a shared resource).
- If using project mode, specify a license project name with the `bsub -Lp` option.

  If you also have `LSF_LIC_SCHED_STRICT_PROJECT_NAME=y` in `lsf.conf` and you have not configured a default project for the required feature, the job is rejected.

  Tip:

  Use the `blstat` command to view information about the `default` license project.
- Update resource requirements.

  If your queue or job starter scripts request a license that is managed by an LSF ELIM, you need to update the job submission scripts to request that license using the license token name.

Examples:

bsub -R "rusage[AppB=1]" -Lp Lp1 myjob

This submits a job called `myjob` to license project `Lp1` and requests one `AppB` license

bsub -R "rusage[AppC=1]" myjob

This submits a job called `myjob` and requests one `AppC` license.

After configuration changes

If you make configuration changes to License Scheduler or to LSF, you need to reconfigure.

1. Run **bld -C** to test for configuration errors.
2. Run **bladmin reconfig all** .
3. If making any change to `lsf.conf` or other LSF configuration files, run **badmin mbdrestart** and **lsadmin reconfig**.

Note:

After Platform License Scheduler configuration changes you may have to run `badmin mbdrestart` for changes to take effect. This applies to the following configuration changes:

- Project changes, additions or deletions
- Feature changes, additions, or deletions, including mode changes
- Cluster locations changes

You may also have to run `lsadmin reconfig` for any changes to the LIM to take effect (for example, if you changed LSF_LIC_SCHED_HOSTS).

Add a cluster to Platform License Scheduler

You must be a License Scheduler administrator.

You can add a new cluster to an existing Platform License Scheduler implementation.

1. Download Platform License Scheduler package from Platform's ftp site.

   Platform suggests you acquire the same version of master `bld` binaries and other architectures used in existing member clusters.
2. Install the Platform License Scheduler package on the new cluster.
3. Use an existing `lsf.licensescheduler` from *$LSF_ENVDIR* of another cluster using the same bld master.
4. Add new cluster name to the `Clusters` section of `lsf.licensescheduler`.
5. Add or modify license distribution policies defined in `lsf.licensescheduler`.
6. Maintain one central `lsf.licensescheduler` file and have all the clusters access it.

   Remember:

   It is essential that `lsf.licensescheduler` file in each cluster is identical.

   You can accomplish this using either of the following methods:

   - Create a symbolic link from each cluster's $LSF_ENVDIR to the central `lsf.licensescheduler` file.
   - Use a CRON-based synchronization script to synchronize the changes made from the central `lsf.licensescheduler` file to the corresponding `lsf.licensescheduler` files in all the clusters.
7. Check that there is no firewall or network issue with communication using the PORT in the `lsf.licensescheduler` file
8. Run **bladmin reconfig** on all hosts where `bld` is running.
9. On the newly added cluster, run **lsadmin limrestart** and then **badmin mbdrestart**.

Configure multiple administrators

The primary License Scheduler admin account must have write permissions in the LSF working directory of the primary LSF admin account.

The administrator account uses a list of users that you specified when you installed License Scheduler. Edit this parameter if you want to add or change administrators. The first user name in the list is the primary License Scheduler administrator. By default, all the working files and directories created by License Scheduler are owned by the primary License Scheduler account.

1. Log on as the primary License Scheduler administrator.
2. In `lsf.licensescheduler`, edit the ADMIN parameter if you want to change the License Scheduler administrator. You can specify multiple administrators separated by spaces.

   For example:

   ```
   ADMIN = lsfadmin user1 user2 root
   ```
3. Run `bld -C` to test for configuration errors.
4. Run `bladmin reconfig all` to make the changes take effect.

Upgrade License Scheduler

You must have License Scheduler installed before you can upgrade. You must be a cluster administrator.

You can upgrade to a new version of License Scheduler without uninstalling and re-installing.

1. Download the new version of the License Scheduler distribution tar files from the ftp site.
2. Get a license for the upgraded version of License Scheduler.
3. Deactivate all queues.

   This pends any running jobs and prevents new jobs from being dispatched.

   badmin qinact all
4. If you have the Platform Aplication Center installed, shut it down.

   pmcadmin stop
5. Back up your existing LSF_CONFDIR, LSB_CONFDIR, and LSB_SHAREDIR according to the procedures at your site.
6. Use the setup script to upgrade License Scheduler.
   a) Source `cshrc.lsf` or `profile.lsf` in old LSF cluster.
   b) Navigate to the location of your tar files and extract.
   c) Run the setup script.
7. Start License Scheduler.
   a) Source `cshrc.lsf` or `profile.lsf`.
   b) Run **lsadmin reconfig**.
   c) Run **ps -ef** to make sure the `bld` is running on the candidate hosts.
   d) Run **badmin mbdrestart**.
   e) Activate the queues.

      badmin qact all
8. If you have the Platform Application Center installed, restart it.

   pmcadmin start

Note:

Platform Application Center version 8.0.1 and later displays License Scheduler workload for both project mode and cluster mode.

Firewalls

Configuration for LSF, License Scheduler, and taskman interoperability.

Set up firewall communication

The `mbatchd` and `bld` listening ports (inbound connections) must be open on either side of the firewall.

- `mbatchd`: Set by LSB_MBD_PORT in `lsf.conf`
- `bld`: Set by PORT in `lsf.licensescheduler`

- If a firewall is between the `mbatchd` and `bld` hosts, both listening ports must be open.
- If a firewall is between `bld` and `blcollect` hosts (for example, `blcollect` is configured to run locally on the license servers and `bld` is on the LSF master host), the `bld` listening port must be open.
- If a firewall is between taskman and bld (where jobs use taskman to interface with License Scheduler), the `bld` listening port must be open.

CHAPTER

3

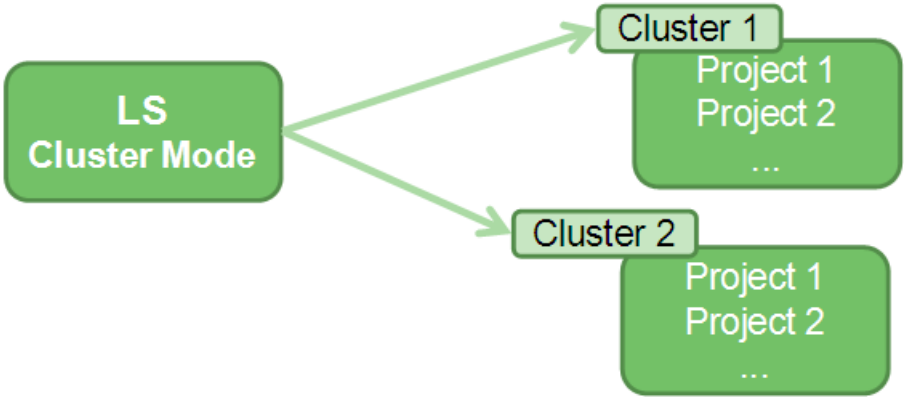
Platform License Scheduler Concepts

Platform License Scheduler modes

When configuring your installation of Platform License Scheduler, you must choose which of project mode and cluster mode best suits your needs for each license you use. Both project mode and cluster mode can be configured in one installation, however, all different licenses required by a job must belong to the same mode.

cluster mode

Distributes license tokens to clusters, where LSF scheduling takes over.



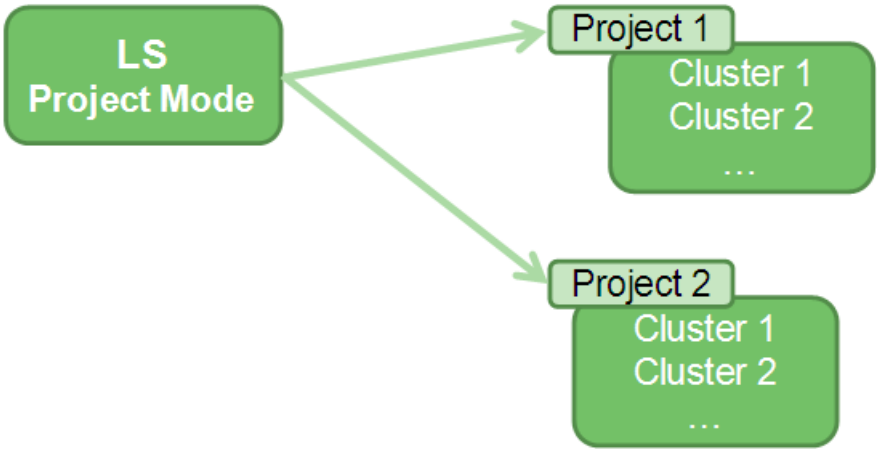
Cluster mode emphasizes high utilization of license tokens above other considerations such as ownership. License ownership and sharing can still be configured, but within each cluster instead of across multiple clusters. Preemption of jobs (and licenses) also occurs within each cluster instead of across clusters.

License tokens are re-used by LSF when a job finishes, without waiting for confirmation from `lmstat` that license tokens are available and reported in the next `blcollect` cycle. This results in higher license utilization for short jobs.

Cluster mode is new in Platform License Scheduler 8.0.

project mode

Distributes license token to projects configured across all clusters.

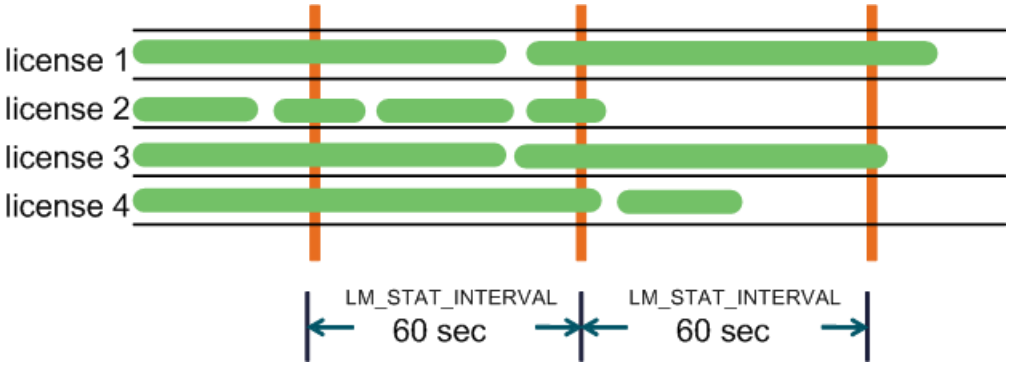


Project mode emphasizes ownership of license tokens by specific projects which span multiple clusters. When running in project mode, Platform License Scheduler checks demand from license owners across all LSF clusters before allocating license tokens. The process of collecting and evaluating demand for all projects in all clusters slows down each scheduling cycle. License tokens are distributed in the next scheduling cycle, once `lmstat` confirms license token availability.

Project mode was the only choice available before Platform License Scheduler 8.0.

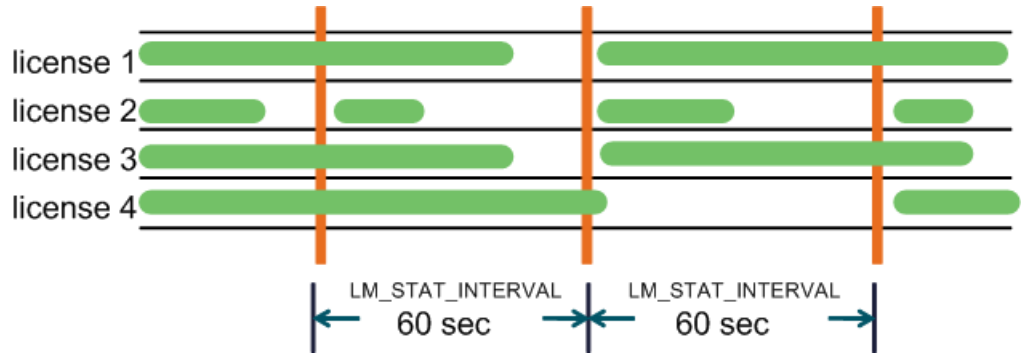
Difference between cluster mode and project mode

The following figure illustrates license utilization in cluster mode for short jobs with the corresponding `lmstat` reporting times:



In cluster mode, when one job finishes running, the next job gets its license immediately without having to wait for the next `lmstat` interval. For example, 4 jobs requiring license 2 are able to run without waiting for `lmstat` to report token distribution.

The following figure illustrates license utilization in project mode for short jobs with the `lmstat` reporting times:



In project mode, each job must wait for `lmstat` to report token distribution before it can get a license and start running. In this example, 3 jobs requiring liense 2 are able to start within the `lmstat` intervals illustrated.

When to use cluster mode

Cluster mode may suit your needs if:

- Your primary goal is to maximize license use.
- Ownership of licenses is a secondary consideration.
- Many jobs are short relative to the `blcollect` cycle (60 seconds by default, set by LM_STAT_INTERVAL).

When to use project mode

Project mode may suit your needs if:

- Your primary goal is to have licenses used by the group owning the licenses.
- Maximizing license use is a secondary consideration.
- Most jobs are long relative to the `blcollect` cycle (60 seconds by default, set by LM_STAT_INTERVAL).

Project groups

When configuring your installation of Platform License Scheduler in project mode, you can choose to configure projects, or extend your project configuration further to form hierarchical project groups.

Project groups pool multiple service domains together and treat them as one source for licenses, and distribute them in a hierarchical fairshare tree. The leaves of the policy tree are the license projects that jobs can belong to. Each project group in the tree has a set of values, including shares and limits.

License ownership is applied at the leaf level; that is, on individual license projects. Ownership of a given internal node equals to sum of the ownership of all of its direct children.

Each feature has its own hierarchical group, but features can share the same hierarchy. The hierarchical scheduling is done per feature across service domains.

projects

Projects alone apply one distribution policy within one service domain. The same local distribution policy can be applied to more than one service domain, but is implemented locally.

groups of projects

Groups of projects apply one distribution policy within one service domain, but assign shares and ownership to groups of projects for greater flexibility. With group license ownership, projects trigger preemption either when the project is using fewer licenses than it owns or when the group to which the project belongs is using fewer licenses than the group owns.

project groups

Projects groups apply one distribution policy across multiple service domains following the configured hierarchical structure. Project groups also allow you to apply hard limits to the number of licenses distributed to each project.

Once configured, the same project group hierarchy can be used for more than one feature.

When to use groups of projects

Grouping projects together in project mode may suit your needs if:

- Licenses are owned at multiple levels, for example by a department and also by projects within the department.
- License ownership is within one service domain. As for ungrouped projects, distribution policies are implemented locally for groups of projects.

When to use project groups

Extending your configuration to include project groups may suit your needs if:

- License ownership spans service domains.
- One distribution policy needs to be applied across several service domains.
- Project limits need to be applied across clusters.

Note:

If required, license project limits can be configured within one Platform LSF cluster using Platform LSF.

Service domains in Platform License Scheduler

A service domain is a group of one or more FlexNet license servers. License Scheduler manages the scheduling of the license tokens, but the license server actually supplies the licenses. You configure the service domain with the license server names and port numbers that serve licenses to a network.

- LAN: a service domain serving licenses to a single cluster
- WAN: a sevic e domain serving licenses to multiple clusters

License Scheduler assumes that any license in the service domain is available to any user who can receive a token from License Scheduler. Therefore, every user associated with a project specified in the distribution policy must meet the following requirements:

- The user is able to make a network connection to every FlexNet license server host in the service domain.
- The user environment is configured with permissions to check out the license from every FlexNet license server host in the service domain.

You must configure at least one service domain for Platform License Scheduler. It groups FlexNet license server hosts that serve licenses to LSF jobs and is used when you define a policy for sharing software licenses among your projects.

If a FlexNet license server host is not part of a License Scheduler service domain, its licenses are not managed by License Scheduler (the license distribution policies you configure in LSF do not apply to these licenses and usage of these licenses does not influence LSF scheduling decisions).

Service domain locality

License feature locality allows you to limit features from different service domains to a specific cluster, so that License Scheduler does not grant tokens to jobs from license that legally cannot be used on the cluster requesting the token. The LAN service domains used in cluster mode are configured using single-cluster locality.

Project mode

In project mode, a cluster can access the same license feature from multiple service domains.

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 for any features that cannot be shared across all the locations. This avoids having to define different distribution and allocation policies for different service domains, and allows hierarchical project group configurations.

To use Platform License Scheduler tokens in project mode, a job submission must specify the `-Lp` (license project) option. The project must be defined for the requested feature in `lsf.licensescheduler`.

Cluster mode

In cluster mode, each license feature in a cluster can access a single license feature from at most one WAN and one LAN service domain.

Platform License Scheduler does not control application checkout behavior. If the same license is available from both the LAN and WAN service domains, License Scheduler expects jobs to try to obtain the license from the LAN first.

Distribution policies

The most important part of License Scheduler is license token distribution. The license distribution policy determines how license tokens are shared among projects or clusters. Whenever there is competition, the configured share assignment determines the portion of license tokens each project or cluster is entitled to.

We refer to both licenses and license tokens because Platform License Scheduler does not control licenses directly. Instead it controls the dispatch of jobs requiring licenses submitted through LSF or `taskman` by tracking license tokens.

Total license tokens

The total number of license tokens managed by License Scheduler for a single feature in one service domain depends on the following:

- The number of active license servers in the service domain
- The number of licenses checked out by applications not managed by LSF

License shares

License shares assigned in the distribution policy determine what portion of total licenses a project (in project mode) or cluster (in cluster mode) receives. Each project or cluster able to use a license featurte must have a share of the license feature in the service domain.

The formula for converting a number of shares to a number of licenses for any given license feature is:

```
(shares assigned to project or cluster)
_______________________________          x (total number of licenses)
(sum of all shares assigned)
```

The number of shares assigned to a license project or cluster is only meaningful when you compare it to the number assigned to other projects or clusters, or to the total number of shares.

When there are no jobs in the system, each project or cluster is assigned license tokens based on share assignments.

Cluster mode distribution policies

static

A portion of the total licenses is allocated to the cluster based on the configured share. The amount is static, and does not depend on the workload in the system.

dynamic

Shares of the total licenses are assigned to each cluster, along with a buffer size. The configured shares set the number of licenses each cluster receives initially, but this is adjusted regularly based on demand from the cluster.

License distribution changes whenever a cluster requests an allocation update, by default every 15 seconds. In each update the allocation can increase by as much as the buffer size. There is no restriction on decreasing cluster allocation.

When dynamic license distribution is used in cluster mode, minimum and maximum allocation values can be configured for each cluster. The minimum allocation is like the

number of non-shared licenses for project mode, as this number of tokens is reserved for the exclusive use of the cluster.

If the minimum value configured exceeds the share assignment for the cluster, only the assigned share is reserved for the cluster.

Cluster shares take precedence over minimum allocations configured. If the minimum allocation exceeds the cluster's share of the total tokens, a cluster's allocation as given by `bld` may be less than the configured minimum allocation.

guarantees within a cluster

Guaranteed shares of licenses are assigned to projects within a cluster using LSF guarantee-type SLAs. Optionally, sharing of guaranteed licenses not in use can be configured.

Guarantees are like ownership for cluster mode, and can be used with both static and dynamic distribution policies.

Note:

Guarantee-type SLAs are only available in LSF version 8.0 or newer.

When to use static license distribution

Configure shares for all license features in cluster mode. Static license distribution is the basic license distribution policy, and is built on by adding additional configuration.

The basic static configuration may meet your needs if:

- Demand for licenses across clusters is predictable and unchanging, or licenses are strictly owned by clusters, or you always have extra licenses.

When to use dynamic license distribution

Dynamic license allocation may meet your needs if:

- Demand for licenses changes across clusters

When to use LSF guarantee SLAs with License Scheduler

Configuring guarantee SLAs within LSF clusters may meet your needs if:

- Licenses within a cluster are owned, and used either preferentially or exclusively by the license owners.

Project mode distribution policies

fairshare

Shares of the total licenses are assigned to each license project.

Unused licenses are shared wherever there is demand, however, when demand exceeds the number of licenses, share assignments are followed. Jobs are not preempted to redistribute licenses; instead licenses are redistributed when jobs finish running.

ownership and preemption

Shares of the total licenses are assigned to each license project. Owned shares of licenses are also assigned.

Unused licenses are shared wherever there is demand, however, when demand exceeds the number of licenses the owned share is reclaimed using premption.

Preemption occurs only while the specified number of owned licenses are not yet in use, and no free licenses are available. Once all owned licenses are being used, License Scheduler waits for licenses to become free (instead of using preemption) and then distributes additional tokens until the share is reached.

Jobs that are preempted by Platform License Scheduler are automatically resumed once licenses become available.

By default, LSF releases the job slot of a suspended job when License Scheduler preempts the license from the job.

Note:

For License Scheduler to give a license token to another project, the applications must be able to release their licenses upon job suspension.

active ownership

Active ownership allows ownership to automatically adjust based on project activity. Ownership is expressed as a percent of the total ownership for active projects. The actual ownership for each project decreases as more projects become active. Set percentage ownership values to total more than 100% to benefit from active ownership.

non-shared licenses

Some licenses are designated as non-shared, and are reserved for exclusive use instead of being shared when not in use.

The number of non-shared licenses is contained by the number of owned licenses, but this number is not included in share calculations for the project. To designate a certain number of licenses as non-shared, add the non-shared number to both the owned and the non-shared values.

When to use fairshare with project mode

Configure fairshare for all license features in project mode. Fairshare is the basic license distribution policy, and is built on by adding additional configuration.

The basic fairshare configuration may meet your needs without configuring additional distribution policies if:

- Licenses are assigned to specific license projects, but not strictly owned.

When to add ownership (and preemption)

Configure licenses as owned when:

- Licenses are owned by licenses projects, but can be loaned out when not in use.

- Maximizing license usage and license ownership are both important considerations. Loaned licenses must be returned to the owners as quickly as possible when needed (using preemption).
- Jobs borrowing licenses can be preempted.

When to add active ownership

Configure active ownership for owned licenses when:

- Ownership values are somewhat dynamic instead of being fixed values, and should decrease as more projects actively seek licenses.

When to add non-shared licenses

Configure licenses as non-shared when:

- Licenses are owned.
- Licenses are used exclusively by the owners.
- Having licenses available to the owners at all times is more important than maximizing license use.

Project mode preemption

Preemption only occurs when there are no free licenses. During preemption, a project releases a borrowed license to the project that owns the license (and now has demand).

Jobs using licenses that support job suspension release their tokens and automatically resume from where they were suspended. Jobs using licenses that do not support suspension are killed and restarted from the beginning.

Preemption only applies to project mode, and depending on your configuration takes the following into consideration:

- runtime (a job that has the smallest run time gets preempted first, in general)
- fairshare settings
- ownership
- priority
- minimal job preemption

Depending on how your projects are set up (whether they are all at the same level or not), your preemption is either flat or hierarchical.

Basic preemption with projects configured

When preemption occurs, License Scheduler calculates token usage for each project. The calculation considers tokens in use, tokens required, and token ownership value.

Based on the token usage, License Scheduler determines the projects that require tokens, and those that have too many.

- Jobs belonging to projects requiring tokens are scheduled first, ordered by project fairshare settings.
- Jobs belonging to projects with extra tokens are preempted first, if needed, ordered by project fairshare settings and the length os time each job has been running.

With PRIORITY

If project `PRIORITY` is configured in the `Project` section, the sort order of projects is based on priority, where a higher priority project is preempted last.

With PREEMPT_ORDER

If `PREEMPT_ORDER` is set to `BY_OWNERSHIP` in the `Feature` section, the projects are sorted by ownership.

- Projects with the highest ownership are scheduled first.
- Projects with the smallest ownership are preempted first.

This setting overrides basic preemption and PRIORITY.

With ENABLE_MINJOB_PREEMPTION

If `ENABLE_MINJOB_PREEMPTION=Y`, the number of preempted jobs is minimized. Projects with extra tokens are sorted by PRIORITY (if configured) or fairshare. The jobs are then sorted by `RUSAGE`.

Jobs with higher `RUSAGE` are preempted first to minimize the number of jobs preempted.

This setting is used in addition to basic preemption or PRIORITY.

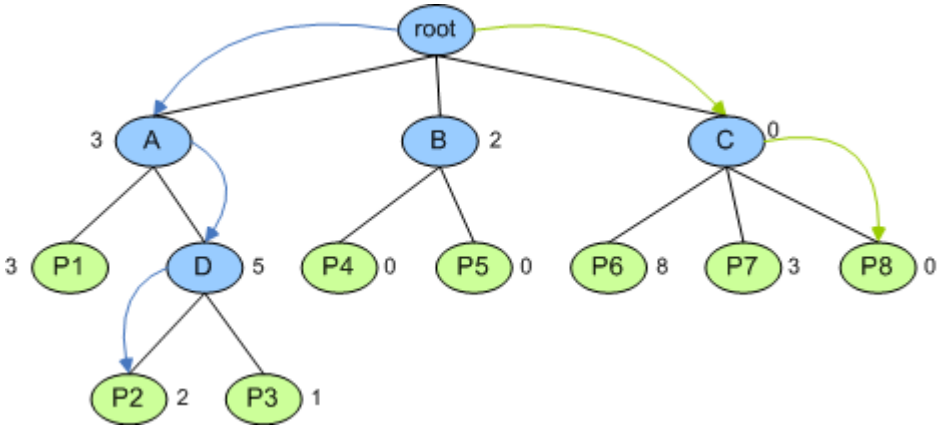
Hierarchical preemption with project groups configured

When project groups are configured, introducing a hierarchy into the project configuration, hierachical preemption applies.

There are two methods of hierarchical preemption:

1. Top-down (default): Preemption occurs between cousins rather than siblings. The result is to balance preemption between the entire hierarchy of projects.
2. Bottom-up (if `LS_PREEMPT_PEER=Y`): Siblings can preempt each other. The result is to balance preemption within a family of projects first.

For example, your projects are set up as follows:



In top-down preemption, if P8 needs a token, it preempts from P1, P2, or P3 (who are more distant relations), not from P6 or P7 (siblings of P8).

In bottom-up preemption, P8 preempts instead from its siblings (P6 or P7).

Limits

Hierarchical preemption is also affected by any limits placed on the projects. If a limit has already been reached (at any level of the hierarchy), License Scheduler considers the next possible node for preemption instead.

Preemption restrictions

A job cannot be preempted if:

- Preemption is restricted by a parameter such as: MAX_JOB_PREEMPT, PREEMPT_RESERVE, LM_REMOVE_INTERVAL, or LS_WAIT_TO_PREEMPT
- The preemptable job's server is not the current checking service domain.
- The job was submitted with a time duration and this time duration has expired.

Both LSF jobs and `taskman` jobs using licenses managed by License Scheduler can be preempted. To ensure lower priority jobs are not preempted too many times, maximum preemption time limits can be enabled with LS_ENABLE_MAX_PREEMPT.

License Scheduler taskman job preemption limits are controlled by the parameter LS_MAX_TASKMAN_PREEMPT in `lsf.licensescheduler`.

LSF preemption with License Scheduler preemption

For LSF jobs the parameter MAX_JOB_PREEMPT sets the maximum number of times a job can be preempted. MAX_JOB_PREEMPT can be defined in `lsb.params`, `lsb.queues`, or `lsb.applications`, with the application setting overriding the queue setting and the queue setting overriding the cluster-wide `lsb.params` definition.

Jobs belonging to a license project that has ownership in License Scheduler can trigger preemption even when no more slots are available in LSF. Configured together with LSF_LIC_SCHED_PREEMPT_SLOT_RELEASE, license job preemption works together with LSF slot-based preemption.

Example

Project `proj1` has ownership of 3 of the license `AppX`.

MXJ = 5, and LSF_LIC_SCHED_PREEMPT_SLOT_RELEASE=Y is configured in `lsf.conf`.

5 jobs are submitted and started using `AppX`, in `proj2`. Then 2 jobs are submitted to `proj1`, and pend waiting for a `AppX` license token. Although the slots are full, the request is sent to License Scheduler, which recognizes the ownership and preempts 2 jobs in `proj2`. The jobs are suspended, both their licenses and slots are released, and the 2 jobs in `proj1` can run.

LSF JOB_CONTROLS configuration

If the LSF administrator has defined JOB_CONTROLS in `lsb.queues` so that job controls (such as the signal SIGTSTP) take effect whe License Scheduler preemption occurs, LIC_SCHED_PREEMPT_STOP=Y in `lsf.conf` must also be defined for License Scheduler preemption to work.

License usage with FlexNet

Platform License Scheduler works differently with different types of applications depending on how the application uses the license features and whether these license features are known at the start of the job.

Known license requirements

For many applications, all license features needed to run its jobs are known before the start of the job.

1. The job submission passes a license usage request to the Platform LSF cluster.
2. LSF sends a query to License Scheduler to see if the license token can be given to the application.
3. When License Scheduler grants permission, LSF gives authorization to the user application.
4. The user application sends a request to FlexNet to check out a license.

Unknown license requirements

Some applications require an initial feature license to start a job and additional feature or sub-feature licenses during job execution. The user who submits the job knows the main license feature needed to start the job, but might not know the additional feature names or the number of additional features required. This additional license feature not specified at job submission is considered unknown license use.

At any time, the user application can either make a request to LSF without requesting verification from License Scheduler, or it can bypass LSF entirely by sending the license request directly to the FlexNet license servers.

1. The user application makes a request to LSF without requesting verification from License Scheduler.
2. LSF gives authorization to the user application because the request did not specify the need for License Scheduler verification.
3. The user application sends a request to FlexNet to check out a license.

Project mode

Known license requirements

Project mode supports known license requirements specified in the `rusage` section of job submissions. By default, each license feature is reserved for the full length of the job.

Optionally, use the `Feature` section parameter DYNAMIC=Y to enable the use of `duration` in the `rusage` string, and release license features after a specified `duration`.

Unknown license requirements

Unknown license requirements not in the `rusage` string are counted as jobs not managed by LSF, and license distribution policies are not applied by default.

Optionally, license requirements not included in the rusage string can be tracked as part of the managed workload in project mode, as long as there is at least one license feature specified in the job's rusage string. Set the parameter ENABLE_DYNAMIC_RUSAGE=Y in the `Feature` section to apply project distribution policies even when license `rusage` is not specified.

Cluster mode

Known license requirements

Cluster mode supports known license requirements specified in the `rusage` section of job submissions. Each license feature is reserved for the full length of the job.

In cluster mode, license requirements cannot be submitted with `duration` specified. If you have known license requirements for only a predetermined part of your job, you must choose between including them in the rusage and reserving for the entire job, or leaving them as unknown requirements.

Unknown license requirements

Unknown license requirements not in the rusage string are counted as part of the managed workload in cluster mode. License features not in the rusage string are not reserved for the job, however, distribution policies do apply. (This is equivalent behavior to ENABLE_DYNAMIC_RUSAGE=Y in project mode.)

CHAPTER

4

Configuring Platform License Scheduler

Configure cluster mode

Use cluster mode to distribute licenses across LSF clusters, leaving the scheduler for each LSF cluster to schedule jobs, allocate licenses to projects within the cluster, and preempt jobs.

Configure parameters

1. Cluster mode can be set globally, or for individual license features. Set individually when using cluster mode for some features and project mode for some features.
   a) If using cluster mode for all license features, define **CLUSTER_MODE=Y** in the `Parameters` section of `lsf.licensescheduler`.
   b) If using cluster mode for some license features, define **CLUSTER_MODE=Y** for individual license features in the `Feature` section of `lsf.licensescheduler`.

   The `Feature` section setting of CLUSTER_MODE overrides the global `Parameter` section setting.
2. List the License Scheduler hosts.

   By default with an LSF installation, the HOSTS parameter is set to the LSF_MASTER_LIST.

   - List the hosts in order from most preferred to least preferred. The first host is the master license scheduler host.
   - Specify a fully qualified host name such as `hostX.mycompany.com` unless all your License Scheduler clients run in the same DNS domain.

   HOSTS=host1 host2
3. Specify the data collection frequency between License Scheduler and FlexNet.

   The default is 60 seconds.

   LM_STAT_INTERVAL=*seconds*
4. Specify the path to the FlexNet command `lmstat`.

   For example, if `lmstat` is located in `/etc/flexlm/bin`:

   ```
   LMSTAT_PATH=/etc/flexlm/bin
   ```

   Tip:

   If the `lmstat` command is not included in the `flexlm/bin` directory, you can find it packaged with your LSF distribution in `$LSF_SERVERDIR`.

Configure clusters

Configure the clusters permitted to use Platform License Scheduler in the `Clusters` section of the `lsf.licensescheduler` file.

This is only required if you are using more than one cluster.

1. In the Clusters section, list all clusters that can use Platform License Scheduler.

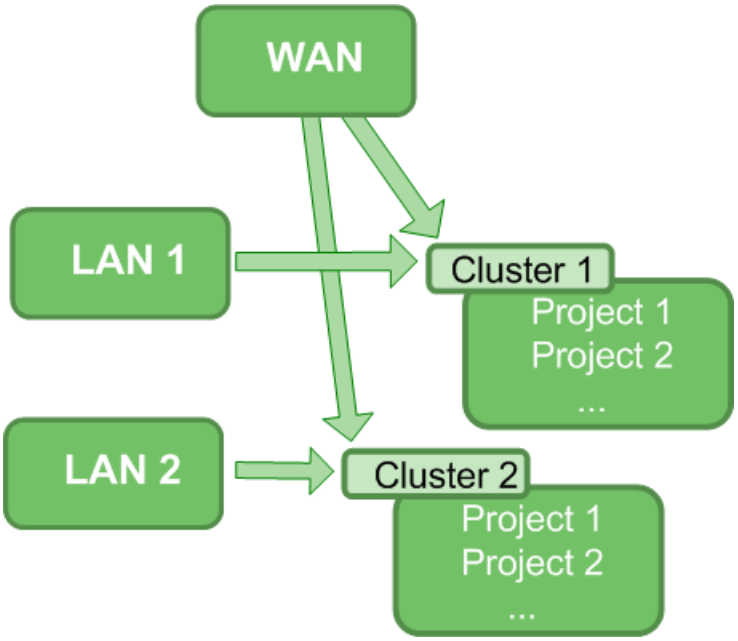
For example:

```
Begin Clusters
CLUSTERS
cluster1
cluster2
End Clusters
```

Cluster mode service domains

A service domain is a group of one or more FlexNet license servers. Platform License Scheduler manages the scheduling of the license tokens, but the license server actually supplies the licenses.

In cluster mode, each cluster can access licenses from one WAN and one LAN service domain.



Platform License Scheduler does not control application checkout behavior. If the same license is available from both the LAN and WAN service domains, License Scheduler expects jobs to try to obtain the license from the LAN first.

Configure ServiceDomain sections

You configure each service domain, with the license server names and port numbers that serve licenses to a network, in the `ServiceDomain` section of the `lsf.licensescheduler` file.

Whether the service domain is a WAN or LAN service domain is specified later in the `Feature` section.

1. Add a `ServiceDomain` section, and define NAME for each service domain.

   For example:

   ```
   Begin ServiceDomain
   NAME=DesignCenterA
   End ServiceDomain
   ```

2. Specify the FlexNet license server hosts for that domain, including the host name and FlexNet port number.

For example:

```
Begin ServiceDomain
NAME=DesignCenterA
LIC_SERVERS=((1700@hostA))
End ServiceDomain
```

For multiple license servers:

```
LIC_SERVERS=((1700@hostA)(1700@hostB))
```

For redundant servers, the parentheses are used to group the three hosts that share the same `license.dat` file:

```
LIC_SERVERS=((1700@hostD 1700@hostE 1700@hostF))
```

Note:

If FlexNet uses a port from the default range, you can specify the host name without the port number. See the FlexNet documentation for the values of the default port range.

```
LIC_SERVERS=((@hostA))
```

Configure LAN service domain

You configure LAN service domains in the `Feature` section of `lsf.licensescheduler`. Only a single cluster and service domain can be specified in each LAN `Feature` section. Licenses from the LAN service domain are statically allocated to the cluster.

1. In the Feature section, set

   ```
   CLUSTER_DISTRIBUTION=service_domain(cluster_name share)
   ```

   Use the service domain name defined in the `ServiceDomain` section.

   For example:

   ```
   Begin Feature
   NAME=verilog
   CLUSTER_DISTRIBUTION=MyLanServer(tokyo_cluster 1)
   End Feature
   ```

Configure WAN service domain

WAN configuration includes all clusters sharing the WAN service domain. As for a LAN service domain, you set this in the CLUSTER_DISTRIBUTION parameter in the `Feature` section of the `lsf.licensescheduler` file.

For a WAN service domain, you can optionally configure dynamic license sharing based on past license use across all clusters served by the WAN service domain, and if required set minimum and maximum allocations for each cluster.

1. Set the WAN service domain name in the CLUSTER_DISTRIBUTION parameter.

   ```
   CLUSTER_DISTRIBUTION = service_domain(cluster share/min/max...)
   ```

   Use the service domain name defined in the `ServiceDomain` section.
2. Configure each cluster.

   All clusters with access to the WAN service domain licenses must be included.

   a) Set the cluster name.
   b) Set the share for each cluster.

The share is a non-negative integer representing the share of licenses each cluster should receive in a static license allocation, and the starting share in a dynamic license allocation.

3. Optionally, set ALLOC_BUFFER in the `Feature` section of the `lsf.licensescheduler` file. When set, this enables a dynamic sharing policy.

   ```
   ALLOC_BUFFER = buffer
   ```

   or

   ```
   ALLOC_BUFFER = cluster1 buffer1 cluster2 buffer2...default buffer
   ```

   - When extra license tokens are available, each cluster's allocation increases to as much as PEAK +BUFFER.

     The value BUFFER is set by ALLOC_BUFFER in the `Feature` section, and the value `PEAK` is the peak value of dynamic license token use over a time interval set by PEAK_INUSE_PERIOD in the `Parameters` or `Feature` section.
   - When allocated tokens are not being use in a cluster, the cluster's allocation goes down to PEAK +BUFFER.

     Since tokens are not being used in the cluster, the peak use value PEAK decreases, thus PEAK +BUFFER also decreases.

   The allocation buffer sets both the rate at which the cluster allocation can grow, and the number of licenses that can go unused, depending on demand.

   Allocation buffers help determine the maximum rate at which tokens can be transferred to a cluster as demand increases in the cluster. The maximum rate of transfer to a cluster is given by the allocation buffer divided by MBD REFRESH INTERVAL. Be careful not to set the allocation buffer too large so that licenses are not wasted because they are be allocated to a cluster that cannot use them.

4. Optionally, when dynamic sharing is enabled (ALLOC_BUFFER is defined) you can set the minimum and maximum allocation for each cluster.

   The minimum allocation reserves license tokens for exclusive use by the cluster; the maximum allocation limits the total number of license tokens received by the cluster.

   Cluster shares take precedence over minimum allocations configured. If the minimum allocation exceeds the cluster's share of the total tokens, a cluster's allocation as given by `bld` may be less than the configured minimum allocation.

To allow a cluster to be able to use licenses only when another cluster does not need them, you can set the cluster distribution for the cluster to 0, and specify an allocation buffer for the number of tokens that the cluster can request.

For example:

```
Begin Feature
CLUSTER_DISTRIBUTION=Wan(CL1 0 CL2 1)
ALLOC_BUFFER=5
End Feature
```

When no jobs are running, the token allocation for CL1 is 5. CL1 can get more than 5 tokens if CL2 does not require them.

Examples

Static example (no allocation buffer set):

```
Begin Feature
NAME=verilog
CLUSTER_DISTRIBUTION=MyWanServer(tokyo_cl 1 newyork_cl 1 toronto_cl 2)
```

```
End Feature
```

In this example, licenses are statically allocated based solely on the number of shares assigned to each cluster. If the number of licenses is not evenly divisible by the number of shares, the additional licenses are distributed round-robin to clusters in the order they appear in CLUSTER_DISTRIBUTION. Thus if there are 98 licenses in total, `tokyo_cl` receives 25, `newyork_cl` receives 25, and `toronto_cl` receives 48. Each cluster limits the total `rusage` of running jobs based on the allocated license tokens.

Dynamic example (allocation buffer set):

```
Begin Feature
NAME=verilog
CLUSTER_DISTRIBUTION=MyWanServer(tokyo_cl 1 newyork_cl 1 toronto_cl 2/10/50)
ALLOC_BUFFER=tokyo_cl 5 newyork_cl 1 toronto_cl 2
End Feature
```

In this example, licenses are initially distributed according to the assigned shares. Since allocation buffers are set, dynamic sharing based on past use is enabled. Based on the allocation buffers, `toyko_cl` receives license tokens the fastest when there is demand within the cluster. Minimum amd maximum allocations of 10 and 50 respectively are set for `toronto_cl`, which also has the largest share.

LAN and dynamic WAN example:

```
Begin Feature
NAME=verilog
CLUSTER_DISTRIBUTION=MyWan(c1 1/1/25 c2 1/1/30 c3 2/5/100) MyLan(c1 1)
ALLOC_BUFFER=c3 5 default 2
End Feature
```

In this example the `verilog` license feature is available from both WAN and LAN service domain, however only cluster c1 receives the license feature from both servers. Licenses from the WAN service domain are initially distributed according to the assigned shares. Since allocation buffers are set, dynamic sharing based on past use is enabled. Based on the allocation buffers cluster `c3` receives license tokens the fastest when there is demand within the cluster.

Configure license features

Each type of license requires a `Feature` section in the `lsf.licensescheduler` file.

1. Define the feature name used by FlexNet to identify the type of license.

   You only need to specify this parameter if the License Scheduler token name is not identical to the FlexNet feature name.

   ```
   Begin Feature
   FLEX_NAME=201-AppZ
   End Feature
   ```

2. (Optional) Use the NAME parameter to define an alias between License Scheduler and FlexNet feature names.

   LSF does not support names that start with a number, or names containing a dash or hyphen character (-), which may be used in the FlexNet feature name. In these cases, define a NAME for the feature as well.

   In this example, the FlexNet feature name 201-AppZ has an alias of AppZ201.

   ```
   Begin Feature
   FLEX_NAME=201-AppZ
   NAME=AppZ201
   End Feature
   ```

Configure taskman jobs in cluster mode

Optionally, to run taskman (interactive) jobs in cluster mode, include the dummy cluster `interactive` in your service domain configuration.

- In the `Feature` section:
  a) Include the dummy cluster `interactive` in the CLUSTER_DISTRIBUTION parameter.
  b) Set a share for the dummy cluster `interactive` .
  c) Optionally, set an allocation buffer for the dummy cluster `interactive` to enable dynamic allocation.

Examples

```
Begin Feature
NAME=licenseA
CLUSTER_DISTRIBUTION=MyLanServer(tokyo_cl 1 interactive 1)
End Feature
```

```
Begin Feature
NAME=licenseB
CLUSTER_DISTRIBUTION=MyWanServer(tokyo_cl 1 newyork_cl 1 interactive 2)
End Feature
```

Allocate licenses to non-LSF jobs

Applies to WAN service domains only.

1. Set WORKLOAD_DISTRIBUTION in the `Feature` section to allocate licenses for non-LSF use.

   ```
   WORKLOAD_DISTRIBUTION=service_domain_name(LSF lsf_distribution NON_LSF
   non_lsf_distribution)
   ```

   If WORKLOAD_DISTRIBUTION is set for a LAN service domain in cluster mode, the parameter is ignored.

For example, to set aside 20% of licenses for use outside of LSF:

```
Begin Feature
NAME=licenseB
CLUSTER_DISTRIBUTION=MyWanServer(tokyo_cl 1 newyork_cl 1)
WORKLOAD_DISTRIBUTION=MyWanServer(LSF 8 NON_LSF 2)
End Feature
```

Restart to implement configuration changes

1. Run **lsadmin limrestart** or **bladmin reconfig** to restart the `bld`.
2. If you have added, changed, or deleted any `Feature` sections, you may need to restart `mbatchd`. In this case a message is written to the log file prompting the restart.

   If required, run **badmin mbdrestart** to restart each LSF cluster.

View license allocation

1. Run **blstat -t *token_name*** to view information for a specific license token (as configured in a `Feature` section).

`blstat` output differs for cluster mode and project mode.

Configure cluster mode with guarantees

Cluster mode distributes licenses across LSF clusters. To guarantee license resources to projects within a cluster and allow loaning of license resources when not in use, use LSF guarantee-type SLAs. Guarantees and loans in cluster mode are similar to non-shared licenses and ownership in project mode.

A guarantee provides jobs belonging to set consumers with specific resources (such as hosts). Jobs run using guaranteed resources when possible. Once the guaranteed resources are used, jobs run outside the guarantee following whatever other scheduling features are configured. Guarantees are configured within a guaranteed resource pool.

Guarantee SLAs are configured in Platform LSF. For more information see *Administering Platform LSF* and *Platform LSF Configuration Reference*.

Configure service classes

Service classes allow access to guaranteed resources. Configure a service class for each license project in the cluster.

1. Configure each `ServiceClass` section in the `lsb.serviceclasses` file. Begin with the line `Begin ServiceClass` and end with the line `End ServiceClass`. For each service class you must specify:
   a) `NAME`: the name of the service class.
   b) `GOALS = [GUARANTEE]`
   c) Optional parameters for the `ServiceClass` section are `ACCESS_CONTROL`, `AUTO_ATTACH`, and `DESCRIPTION`.

   You can configure as many service class sections as you need.

   Important:

   The name you use for your service class cannot be the same as an existing host partition or user group name.

   For example:

```
Begin ServiceClass
NAME = sla1
GOALS = [GUARANTEE]
ACCESS_CONTROL=LIC_PROJECTS[ proj1 ]
DESCRIPTION = A guarantee SLA with access restricted to the license project proj1.
End ServiceClass
```

Automatically attach jobs to service classes

When the optional parameter AUTO_ATTACH is set, jobs are automatically attached to the service class.

When automatic attachment is not set, jobs can be submitted to the service class using **bsub -sla *serviceclass_name***.

If a job can access more than one SLA with automatic attachment set, it is attached to the first valid SLA in the order of the configuration file.

1. Set **AUTO_ATTACH=Y** in the `ServiceClass` section in the `lsb.serviceclasses` file. For example:

```
Begin ServiceClass
NAME = sla1
GOALS = [GUARANTEE]
ACCESS_CONTROL=LIC_PROJECTS[ proj1 ]
```

```
AUTO_ATTACH=Y
DESCRIPTION = A guarantee SLA with access restricted to the license project proj1.
Jobs submitted to proj1 are attached to the SLA automatically and run on guaranteed
resources if possible.
End ServiceClass
```

Configure a resource pool of license tokens

Guaranteed resource pools provide a minimum resource guarantee to consumers, and can optionally loan out guaranteed resources not in use.

Guaranteed resource pools are defined in `lsb.resources` and used by consumers defined within `ServiceClass` sections in `lsb.serviceclasses`.

1. Configure a `GuaranteedResourcePool` section in `lsb.resources`. Begin with the line `Begin GuaranteedResourcePool` and end with the line `End GuaranteedResourcePool`. Specify the following:
   a) `NAME`: the name of the guaranteed resource pool.
   b) `TYPE`: the guarantee type. For licenses, use the type `resources` and include the name of the license feature.
   c) `DISTRIBUTION`: share assignments for all service classes using the resource pool. Can be percent or absolute numbers.
   d) Optional parameters for GuaranteedResourcePool sections of resources are `LOAN_POLICIES`, and `DESCRIPTION`.

   You can configure as many resource pools as you need. One resource pool can be used by several SLAs, and one SLA can access multiple resource pools.

   For example:

```
Begin GuaranteedResourcePool
NAME = hspice_guarantees
TYPE = resource[hspice]
DISTRIBUTION = ([proj1_sc,50%][proj2_sc,50%])
DESCRIPTION = A resource pool of hspice licenses controlled by License Scheduler and
used by proj1_sc and proj2_sc.
End GuaranteedResourcePool
```

Configure loans

Loans from unused guarantees are recommended when using cluster mode. When loans are disabled, use a static license distribution policy.

When configured, unused license resources are loaned out based on the loan policy. The loan policy allows specific queues to access unused resources from guaranteed resource pools.

1. Configure a guaranteed resource pool in `lsb.resources` with the required `NAME`, `TYPE`, and `DISTRIBUTION` parameters.
2. Add a loan policy to the guaranteed resource pool.

   `LOAN_POLICIES= QUEUES[queue_name]` allows you to specify which queues can access loaned resources. Use the keyword `all` to loan to jobs from any queue.

   For example, to allow loans to jobs from the queue `my_queue`:

```
Begin GuaranteedResourcePool
...
LOAN_POLICIES = QUEUES[my_queue]
...
End GuaranteedResourcePool
```

Configure loans to short jobs

Loans can be restricted based on job runtime, or estimated runtime.

1. Add the policy `DURATION[minutes]` to the guaranteed resource pool configuration in `lsb.resources`, where minutes is an integer.

   `DURATION` allows you to set a maximum job runtime limit (or estimated runtime, whichever is shorter) for jobs to borrow resources. Omit DURATION completely to allow jobs with any runtime to borrow from the gurantee.

   For example, to allow loans to jobs from any queue with a runtime of 10 minutes or less:

```
Begin GuaranteedResourcePool
...
LOAN_POLICIES = QUEUES[all] DURATION[10]
...
End GuaranteedResourcePool
```

Configure loans to stop when jobs are waiting for guaranteed resources

Loans can be restricted so that jobs have access to the loaned resources only when consumers with unused guaranteed resources do not have pending loads.

Restricting loans is useful when running jobs that require several licenses. With restricted loans enabled, loaning out single licenses will not delay jobs waiting for license resources to accumulate.

1. Add the policy `CLOSE_ON_DEMAND` to the guaranteed resource pool configuration in `lsb.resources`. For example:

```
Begin GuaranteedResourcePool
...
LOAN_POLICIES = QUEUES[queue1] CLOSE_ON_DEMAND
...
End GuaranteedResourcePool
```

Configure a queue with access to all guaranteed resources

Queues with very high priority (such as administrator test queues) can be configured with access to all guaranteed resources, regardless of SLA demand.

1. Configure a queue in `lsb.queues` with `SLA_GUARANTEES_IGNORE = Y`.

Note:

Using SLA_GUARANTEES_IGNORE=Y defeats the purpose of guaranteeing resources. This should be used sparingly for low traffic queues only.

Restart for changes to take effect

Cluster mode must be enabled, and LSF clusters must be restarted for LSF configuration changes to take effect.

1. In the `Parameters` section of `lsf.licensescheduler`, confirm cluster mode is enabled (**CLUSTER_MODE=Y**).

2. Run **badmin mbdrestart** to restart each LSF cluster.
3. Run **lsadmin limrestart** or **bladmin reconfig** to restart the `bld`.

View guaranteed resource pools

Guaranteed resource pool configuration includes the resource type, and distribution among consumers defined in the corresponding service classes.

1. Run `bresources -g -l -m` to see details of the guaranteed resource pool configuration, including a list of hosts currently in the resource pool.

Project mode using projects

License projects allow you to configure license distribution when running in project mode. Each distribution policy is applied locally, within service domains.

Tip:

Although license projects are not the same as LSF projects, you can map your license project names to LSF project names for easier monitoring.

Configure parameters

1. Project mode can be set globally, or for individual license features. Set individually when using project mode for some features and cluster mode for some features.
   a) If using project mode for all license features, define **CLUSTER_MODE=N** in the `Parameters` section of `lsf.licensescheduler`.
   b) If using project mode for some license features, define **CLUSTER_MODE=N** for individual license features in the `Feature` section of `lsf.licensescheduler`.

   The `Feature` section setting of CLUSTER_MODE overrides the global `Parameter` section setting.
2. List the License Scheduler hosts.

   By default with an LSF installation, the HOSTS parameter is set to the LSF_MASTER_LIST.

   - List the hosts in order from most preferred to least preferred. The first host is the master license scheduler host.
   - Specify a fully qualified host name such as `hostX.mycompany.com` unless all your License Scheduler clients run in the same DNS domain.

   HOSTS=host1 host2
3. Specify the data collection frequency between License Scheduler and FlexNet.

   The default is 30 seconds.

   LM_STAT_INTERVAL=*seconds*
4. Specify the path to the FlexNet command `lmstat`.

   For example, if `lmstat` is located in `/etc/flexlm/bin`:

   ```
   LMSTAT_PATH=/etc/flexlm/bin
   ```

Configure clusters

Configure the clusters permitted to use Platform License Scheduler in the `Clusters` section of the `lsf.licensescheduler` file.

This is only required if you are using more than one cluster.

1. In the Clusters section, list all clusters that can use Platform License Scheduler.

For example:

```
Begin Clusters
CLUSTERS
cluster1
cluster2
End Clusters
```

Configure projects

Each project defined in a `Projects` section of `lsf.licensescheduler` can have a distribution policy applied in the `Feature` section, where projects can be associated with license features.

1. Define the projects with or without priority.

```
Begin Projects
PROJECTS       PRIORITY
Lp1             3
Lp2             1
Lp3             2
default         0
End Projects
```

The higher the number, the higher the priority. When 2 projects have the same priority number configured, the first listed project has a higher priority. Priority is taken into account when license preemption occurs, where lower priority projects are prempted first.

If not explicitly configured, the default project has the priority of 0. A default project is used when no license project is specified during job submission.

Add project description

Optionally, you can add a project description of up to 64 characters to your projects to help identify them.

1. In the `Project` section of `lsf.licensescheduler`, find the project and add a description in the `DESCRIPTION` column.

For example:

```
Begin Projects
PROJECTS PRIORITY DESCRIPTION
p1       10       "Engineering project 123"
p2       9        "QA build project 2C"
P3       8        ""
End Projects
```

When running `blinfo -Lp` or `blinfo -G`, any existing project descriptions display.

Project mode service domains

A service domain is a group of one or more FlexNet license servers. Platform License Scheduler manages the scheduling of the license tokens, but the license server actually supplies the licenses. You must configure at least one service domain for Platform License Scheduler.

In project mode, each cluster can access licenses from multiple WAN and LAN service domains. Platform License Scheduler collects license availability and usage from FlexNet license server hosts, and merges this with license demand and usage information from LSF clusters to make distribution and preemption decisions.

Note:

Unless you require multiple service domains for some specific reason, we recommend configuring both modes with at most one LAN and one WAN for each feature in a cluster. Because Platform License Scheduler does not control license checkout, running with one cluster accessing multiple service domains is not optimal.

Configure service domains

You configure each service domain, with the license server names and port numbers that serve licenses to a network, in the `ServiceDomain` section of the `lsf.licensescheduler` file.

1. Add a `ServiceDomain` section, and define NAME for each service domain.

   For example:

   ```
   Begin ServiceDomain
   NAME=DesignCenterA
   End ServiceDomain
   ```

2. Specify the FlexNet license server hosts for that domain, including the host name and FlexNet port number.

   For example:

   ```
   Begin ServiceDomain
   NAME=DesignCenterA
   LIC_SERVERS=((1700@hostA))
   End ServiceDomain
   ```

   For multiple license servers:

   ```
   LIC_SERVERS=((1700@hostA)(1700@hostB))
   ```

   For redundant servers, the parentheses are used to group the three hosts that share the same `license.dat` file:

   ```
   LIC_SERVERS=((1700@hostD 1700@hostE 1700@hostF))
   ```

Note:

If FlexNet uses a port from the default range, you can specify the host name without the port number. See the FlexNet documentation for the values of the default port range.

```
LIC_SERVERS=((@hostA))
```

Configure license features

Each type of license requires a `Feature` section in the `lsf.licensescheduler` file.

The `Feature` section includes the license distribution policy.

1. Define the feature name used by FlexNet to identify the type of license.

   You only need to specify this parameter if the License Scheduler token name is not identical to the FlexNet feature name.

   ```
   Begin Feature
   FLEX_NAME=201-AppZ
   End Feature
   ```

2. Optionally, use the NAME parameter to define an alias between License Scheduler and FlexNet feature names.

LSF does not support names that start with a number, or names containing a dash or hyphen character (-), which may be used in the FlexNet feature name. In these cases, define a NAME for the feature as well.

In this example, the FlexNet feature name 201-AppZ has an alias of AppZ201.

```
Begin Feature
FLEX_NAME=201-AppZ
NAME=AppZ201
End Feature
```

3. Define a distribution policy.

   A distribution policy defines the license fairshare policy in the format:

   ```
   DISTRIBUTION = ServiceDomain1 (project1 share_ratio project2 share_ratio ...)
   ServiceDomain2 (project3 share_ratio ...)
   ```

   For example, a basic configuration assigns shares:

   ```
   Begin Feature
   FLEX_NAME=201-AppZ
   NAME=AppZ201
   DISTRIBUTION = DesignCenterA (LpA 2 LpB 1 default 1)
   End Feature
   ```

   LpA has the right to twice as many licenses as LpB. Jobs submitted without a license project specified can run under the default project.

4. Optionally, add owned licenses to the distribution policy in the format:

   ```
   DISTRIBUTION = ServiceDomain1 (project1 share_ratio/number_owned project2
   share_ratio/number_owned ...) ServiceDomain2 (project3 share_ratio ...)
   ```

   If `LS_FEATURE_PERCENTAGE=Y` or `LS_ACTIVE_PERCENTAGE=Y` in `lsf.licensescheduler`, *number_owned* is expressed as a percentage of the total licenses.

   Example 1:

   ```
   DISTRIBUTION = LanServer(Lp1 1 Lp2 1/10)
   ```

   This example assumes there are 10 licenses in total, all owned by Lp2.

   The two Platform License Scheduler projects, `Lp1` and `Lp2`, and share the licenses, but grant ownership of the licenses to one of the projects (`Lp2`).

   When `Lp2` has no work to be done, `Lp1` can use the licenses. When `Lp2` has work to do, `Lp1` must return the license immediately to `Lp2`. The license utilization is always at the maximum, showing that all licenses are in use even while the license distribution policies are being enforced.

   Example 2:

   ```
   DISTRIBUTION=LanServer1(Lp1 1 Lp2 2/6)
   ```

   `Lp1` is set to use one third of the available licenses and `Lp2` to use two thirds of the licenses. However, `Lp2` is always entitled to six licenses and preempts other license project jobs when licenses are needed immediately.

   If the projects are competing for a total of 12 licenses, `Lp2` is entitled to eight (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.

Track partial and unspecified license use

When you want to manage licenses not included in job resource requirements or have applications you know use licenses for only part of the length of each job, use these optional settings.

1. Optionally, specify DYNAMIC=Y to consider the license feature as a dynamic resource when it is only used for part of the job.

   Set DYNAMIC=Y for applications with known license use that do not use the license for the entire length of the job. Jobs submitted with duration specified then release the license when not in use.

   ```
   Begin Feature
   NAME = p1_2
   DISTRIBUTION= Lan1 (a 1 b 1 c 1 default 1)
   DYNAMIC=Y
   End Feature
   ```

   For example, a taskman job submission with duration:

   taskman -R "rusage[p1_2=1:duration=2]" myjob

2. Optionally, set ENABLE_DYNAMIC_RUSAGE=Y in the Feature section of `lsf.licensescheduler` to track license use of license features not specified at job submission. For example:

   ```
   Begin Feature
   NAME = feat2
   DISTRIBUTION = LanServer(proj1 1 default 1)
   ENABLE_DYNAMIC_RUSAGE = y
   End Feature
   ```

   Submit a job to run the application, specifying the license feature name:

   bsub -R "rusage[feat1=1]" -Lp proj1 app1

   The job runs and license `feat1` is checked out:

   blstat

   ```
   FEATURE: feat1
   SERVICE_DOMAIN: LanServer TOTAL_INUSE: 1    TOTAL_RESERVE: 0    TOTAL_FREE: 4
   OTHERS: 0
     PROJECT                 SHARE   OWN  INUSE RESERVE FREE   DEMAND
    proj1                    50.0 %   0     1    0      2       0
    default                  50.0 %   0     0    0      3       0
   FEATURE: feat2
   SERVICE_DOMAIN: LanServer TOTAL_INUSE: 0    TOTAL_RESERVE: 0    TOTAL_FREE: 10
   OTHERS: 0
   PROJECT                SHARE   OWN  INUSE RESERVE FREE   DEMAND
    proj1                   50.0 %   0     0    0      5       0
    default                 50.0 %   0     0    0      5       0
   ```

blusers -l

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

blusers -J

```
JOBID   USER     HOST     PROJECT          CLUSTER        START_TIME
1896    user1    hostA    proj1            cluster1       Aug  9 10:01:25
RESOURCE        RUSAGE        SERVICE_DOMAIN
feat1           1             LanServer
```

Later, `app1` checks out feature `feat2`. Since it was not specified at job submission, `feat2` is a class C license checkout. But since it is configured with ENABLE_DYNAMIC_RUSAGE=Y, jobs that require `feat2` are considered managed workload, and subject to the distribution policies of project `proj1`:

blstat

```
FEATURE: feat1
SERVICE_DOMAIN: LanServer
TOTAL_INUSE: 1    TOTAL_RESERVE: 0    TOTAL_FREE: 4    OTHERS: 0
PROJECT                SHARE   OWN  INUSE RESERVE FREE   DEMAND
 proj1                   50.0 %   0     1    0      2       0
 default                 50.0 %   0     0    0      2       0
```

```
FEATURE: feat2
SERVICE_DOMAIN: LanServer
```

```
TOTAL_INUSE: 1    TOTAL_RESERVE: 0    TOTAL_FREE: 9    OTHERS: 0
PROJECT                 SHARE   OWN  INUSE RESERVE FREE   DEMAND
 proj1                    50.0 %   0     1    0      4       0
 default                  50.0 %   0     0    0      5       0
```

blusers -l

```
FEATURE  SERVICE_DOMAIN  USER   HOST    NLICS  NTASKS  OTHERS  DISPLAYS   PIDS
feat1     LanServer       user1  hostA  1      1       0    (/dev/tty)  (16326)
feat2     LanServer       user1  hostA  1      1       0    (/dev/tty)  (16344)
```

blusers -J

```
JOBID   USER      HOST      PROJECT             CLUSTER        START_TIME
1896    user1     hostA     proj1               cluser1        Aug  9 10:01:25
RESOURCE        RUSAGE        SERVICE_DOMAIN
feat1           1             LanServer
feat2           1             LanServer
```

Restart to implement configuration changes

1. Run **lsadmin limrestart** or **bladmin reconfig** to restart the `bld`.
2. If you have added, changed, or deleted any `Feature` sections, you may need to restart `mbatchd`. In this case a message is written to the log file prompting the restart.

   If required, run **badmin mbdrestart** to restart each LSF cluster.

View projects and descriptions

1. Run `blinfo -Lp` to view projects and descriptions.

   For example:

   blinfo -Lp

   ```
   PROJECT PRIORITY DESCRIPTION
   p1      10       Engineering project 123
   p2      9        QA build project 2C
   P3      8
   ```

View license allocation

1. Run **blstat -t *token_name*** to view information for a specific license token (as configured in a `Feature` section).

   `blstat` output differs for cluster mode and project mode.

Project mode optional settings

Once you have configured Platform License Scheduler in project mode with projects or project groups, you may want to include some additional configuration that is not required, but can be useful.

Active ownership

With ownership defined, projects with demand for licenses are able to reclaim licenses up to the assigned ownership share for the project. With active ownership enabled, ownership is expressed as a percent of the total ownership for active projects, and the actual ownership for each project decreases as more projects become active. This allows ownership to automatically adjust based on project activity.

Active ownership can be used with projects, groups of projects, and project groups. Set percentage ownership values to total more than 100% to benefit from active ownership.

Configure active ownership

When active ownership is enabled, ownership settings for inactive projects are disregarded during license token distribution.

1. Set **LS_ACTIVE_PERCENTAGE=Y** in the `Feature` section.

   All ownership values for inactive projects are set to zero, and total ownership percent is adjusted if it exceeds 100%.

   LS_FEATURE_PERCENTAGE=Y is automatically set, and owned and non-shared values are expressed in percent. If used with project groups, OWNERSHIP, LIMITS and NON_SHARED are expressed in percent.
2. Set the percentage of owned licenses in the DISTRIBUTION parameter (`Feature` section) for a total percentage exceeding 100%.

   For example:

   ```
   ...
   DISTRIBUTION=wanserver (Lp1 2/50 Lp2 1/30 Lp3 2/30 Lp4 3/30)
   LS_ACTIVE_PERCENTAGE=Y
   ...
   ```

   In this example, all four license projects are configured with a share and an owned value. Lp1 has the greatest number of owned licenses, and can use preemption to reclaim the most licenses.

   If only Lp1 is active, Lp1 owns 50% of licenses. Total active ownership is 50%, so no adjustment is made.

   If Lp1 and Lp2 are active, Lp1 owns 50% and Lp2 owns 30%. Total active ownership is 80%, so no adjustment is made.

   If Lp1, Lp2, and Lp3 are active, Lp1 owns 50%, Lp2 owns 30%, and Lp3 owns 30%. Total active ownership is 110%, so ownership is scaled to result in Lp1 owning 46%, Lp2 owning 27%, and Lp3 owning 27%. (Exact numbers have been rounded.)

   If all projects are active, the total active ownership is 140%. Ownership is scaled to result in Lp1 owning 37%, Lp2 owning 21%, Lp3 owning 21%, and Lp4 owning 21%. (Exact numbers have been rounded.)

Default projects

Jobs requiring a license feature but not submitted to a license project for that feature are submitted to the `default` project. For jobs to run, a share of license tokens must be assigned to the `default` project.

If you do not want the `default` project to get shares of license tokens, you do not need to define a `default` project in the distribution policy for a feature, however jobs in the `default` project will pend by default.

To avoid having jobs submitted without a project pend, either assign shares to the `default` project, or disable default projects so jobs are rejected.

Configure default project shares

Jobs cannot run in the `default` project unless shares are assigned.

1. Define a default project in the `Feature` section DISTRIBUTION parameter.

   Any job submitted without a project name specified by -Lp can now use tokens from the default project.

Disable default projects

License token jobs submitted without a project specified are accepted and assigned to the `default` project, unless your configuration specifies that such jobs be rejected.

1. Optionally, set `LSF_LIC_SCHED_STRICT_PROJECT_NAME=y` in `lsf.conf`.

   Jobs submitted without a project specified are rejected, and the `default` license project is not used.

Groups of projects

Configuring groups of projects lets you set shares and ownership for each group and distribute license features to groups of projects. A license project should only belong to one group. Preemption first occurs between groups of projects, and then occurs between projects.

Preemption with groups of projects

The following tables show changes in preemption behavior based on ownership configured for groups of projects, with a total of 20 licenses. With groups of projects configured, `GroupA` is able to preempt in order to reclaim 10 owned licenses. Since `Lp2` is not using all 5 owned licenses, `Lp1` can use more than the share it owns.

Project license ownership only

License project	Licenses owned	Licenses used
Lp1	5	6
Lp2	5	0
Lp3	5	7
Lp4	5	7

Groups of projects with license ownership

Group	License projects	Project licenses owned	Licenses used after preemptions
GroupA	Lp1	5	9
	Lp2	5	1
GroupB	Lp3	5	6
	Lp4	5	4

Configure group license ownership

1. In `lsf.licensescheduler`, set the GROUP parameter in the `Feature` section.
   a) Set up groups and members.

   For example:

```
Begin Feature
NAME = AppY
DISTRIBUTION = LanServer1(Lp1 5/5 Lp2 5/5 Lp3 5/5 Lp4 5/5)
GROUP = GroupA(Lp1 Lp2) GroupB (Lp3 Lp4)
End Feature
```

   In this example, `Lp1` and `Lp2` belong to the group `GroupA`. `Lp3` and `Lp4` belong to the `GroupB` group.

Configure interactive (taskman) jobs

By default, interactive (taskman) jobs do not receive a share of the license token allocation, while all clusters receive equal shares.

You can allocate a share of all license features to interactive jobs in the `Parameters` section.

1. To globally enable a share of the licenses for interactive tasks, you must set the `ENABLE_INTERACTIVE` in `lsf.licensescheduler`.

   In `lsf.licensescheduler`, edit the `Parameters` section:

```
Begin Parameters
...
ENABLE_INTERACTIVE = y
...
End Parameters
```

   When the change in configuration takes effect, interactive tasks are allocated the same share (by default) as each cluster.

Configure cluster and interactive allocations

By default in project mode, each cluster receives one allocation share from a license feature, and interactive tasks receive no shares.

You can modify the allocation of license shares across clusters and to interactive tasks in individual `Feature` sections.

1. In the `Features` section of `lsf.licensescheduler`, set the ALLOCATION parameter.

ALLOCATION=*project_name* **(***cluster_name* [*number_shares*] ... **)**

Allocation examples

For example, this ALLOCATION setting matches the default when ALLOCATION is undefined and interactive jobs are enabled with **ENABLE_INTERACTIVE=Y**. An equal share is allocated to each cluster and to interactive jobs.

```
Begin Feature
NAME = AppX
DISTRIBUTION = LanServer1 (Lp1 1)
ALLOCATION = Lp1 (Cluster1 1 Cluster2 1 interactive 1)
End Feature
```

In this example licenses are shared equally between `cluster1` and interactive tasks, with `cluster2` receiving nothing:

```
Begin Parameters
...
ENABLE_INTERACTIVE = y
...
End Parameters
Begin Feature
NAME = AppY
DISTRIBUTION = LanServer (Lp1 1)
ALLOCATION = Lp1(cluster1 2 cluster2 0 interactive 2)
End Feature
```

In the following example, even though the global allocation to interactive jobs is disabled (**ENABLE_INTERACTIVE = N**), ALLOCATION defined in the `Feature` section can assign a share to interactive jobs for this license feature.

```
Begin Feature
NAME = AppZ
DISTRIBUTION = LanServer (Lp1 1)
ALLOCATION = Lp1(cluster1 0 cluster2 1 interactive 2)
End Feature
```

Given a total of 12 licenses, 4 are allocated to `cluster2` and 8 are allocated to interactive tasks.

Configure feature groups

Feature groups configured in one `FeatureGroup` section allow you to view the information for multiple features, grouped together.

1. In `lsf.licensescheduler`, configure a `FeatureGroup` section, listing the license features associated with that license.
   - Each `FeatureGroup` section must have a unique name.
   - The feature names in FEATURE_LIST must already be defined in `Feature` sections.
   - FEATURE_LIST cannot be empty or contain duplicate feature names.
   - Features can appear in more than one `FeatureGroup` section.

   For example:

   ```
   Begin FeatureGroup
   NAME = Corporate
   FEATURE_LIST = ASTRO VCS_Runtime_Net Hsim Hspice
   End FeatureGroup
   ```

   ```
   Begin FeatureGroup
   NAME = Offsite
   ```

```
FEATURE_LIST = Encounter NCSim  NCVerilog
End FeatureGroup
```

Restart to implement configuration changes

Changes made in `lsf.licensescheduler` require restarting the `bld`.

Changes made in `lsf.conf` require restating the LSF clusters.

1. Run **badmin restart** to restart each LSF cluster.
2. Run **lsadmin limrestart** or **bladmin restart** to restart the `bld`.

View license feature group information

When FEATURE_LIST is configured for a group of license features in `lsf.licensescheduler`, you can view detailed information about the groups.

1. Run `blinfo -g` or `blstat -g`.

   For example, if the feature group called myFeatureGroup1 has the members feature2 and feature3:

   blstat -g "myFeatureGroup1"

   Information displays for feature2 and feature3 in descending alphabetic order.

   Run `blstat -g` alone or with options `-Lp`, `-t`, `-D` ,`-G'`, `-s`.

   Run `blinfo '-g'` alone or with options `-a`, `-t`, `-C`, and `-A`.

License feature locality

License feature locality allows you to limit features from different service domains to a specific cluster, so that License Scheduler does not grant tokens to jobs from license that legally cannot be used on the cluster requesting the token.

How locality works

Setting locality means that license resources requested from different clusters are mapped to different tokens in License Scheduler

Features with different locality are treated a different tokens by License Scheduler. You must configure separate feature sections for same feature with different localities.

Note:

You must make sure that your features are configured so that the applications always first try to check out licenses locally.

When License Scheduler receives license requests from LSF, it knows where the request is from, and it interprets the request into demands for tokens usable by that cluster. For example, if `clusterA` sends a request to the `bld` asking for 1 `hspice` license, License Scheduler marks the demand for both `hspice@clusterA` and `hspice`. When the job gets either token to run, the demand is cleaned up for both tokens.

Configure locality

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 as different resources.

1. In `lsf.licensescheduler`'s Feature section, configure LOCAL_TO.

   For example: `LOCAL_TO=Site1(clusterA clusterB)` configures the feature for more than one cluster, where the cluster names are already defined in the Clusters section of `lsf.licensescheduler`.

   `LOCAL_TO=clusterA` configures locality for only one cluster. This is the same as `LOCAL_TO=clusterA(clusterA)`.

   License Scheduler now 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.

2. (Optional) View locality settings.
   a) Run **blinfo -A**.

      The feature allocation by cluster locality displays.

```
FEATURE          PROJECT   ALLOCATION
hspice           Lp1       [clusterA, 25.0%] [clusterB, 25.0%]
                           [clusterC, 25.0%] [interactive, 25.0%])
                 Lp2       [clusterA, 50.0%] [clusterB, 50.0%])
hspice@clusterA  Lp1       [clusterA, 100.0%])
                 Lp2       [clusterA, 100.0%])
hspice@siteB     Lp1       [clusterB, 80.0%] [clusterC, 20%])
                 Lp2       [clusterB, 80.0%] [clusterC, 20%])
hspice@clusterC  Lp1       [clusterC, 60.0%] [interactive, 40.0%)
                 Lp2       [clusterC, 60.0%] [interactive, 40.0%)
                 Lp3       [clusterC, 60.0%] [interactive, 40.0%)
vcs              Lp1       [clusterA, 33.0%] [clusterB, 33.0%]
                           [interactive, 33.0%])
                 Lp2       [clusterA, 50.0%] [clusterB, 50.0%])
vcs@clusterA     Lp1       [clusterA, 100.0%])
                 Lp2       [clusterA, 100.0%])
vcs@siteB        Lp1       [clusterB, 80.0%] [clusterC, 20%])
                 Lp2       [clusterB, 80.0%] [clusterC, 20%])
vcs@clusterC     Lp1       [clusterC, 60.0%] [interactive, 40.0%)
                 Lp2       [clusterC, 60.0%] [interactive, 40.0%)
                 Lp3       [clusterC, 60.0%] [interactive, 40.0%)
```

   b) Run **blinfo -C**.

      The cluster locality information for the features displays.

```
NAME: hspice     FLEX_NAME: hspice
 CLUSTER_NAME    FEATURE          SERVICE_DOMAINS
 clusterA        hspice           SD3 SD4
                 hspice@clusterA  SD1
 clusterB        hspice           SD3 SD4
                 hspice@siteB     SD3
 clusterC        hspice           SD3 SD4
                 hspice@siteB     SD3
                 hspice@clusterC  SD5

NAME: vcs        FLEX_NAME: VCS_Runtime
 CLUSTER_NAME    FEATURE          SERVICE_DOMAINS
 clusterA        vcs              SD3 SD4
                 vcs@clusterA     SD1
 clusterB        vcs              SD3 SD4
                 vcs@siteB        SD3
 clusterC        vcs              SD3 SD4
                 vcs@siteB        SD3
                 vcs@clusterC     SD5
```

c) Run **blusers**.

```
FEATURE          SERVICE_DOMAIN  USER     HOST       NLICS    NTASKS
hspice@clusterA  SD1             user1    host1      1        1
hspice@siteB     SD2             user2    host2      1        1
```

d) Run **blstat**.

```
FEATURE: hspice
 SERVICE_DOMAIN: SD3 SD4
 TOTAL_INUSE: 0    TOTAL_RESERVE: 0    TOTAL_FREE: 22   OTHERS: 0
 PROJECT                 SHARE   OWN  INUSE RESERVE FREE   DEMAND
 Lp1                    50.0 %   0     0    0       11       0
 Lp2                    50.0 %   0     0    0       11       0
FEATURE: hspice@clusterA
 SERVICE_DOMAIN: SD1
 TOTAL_INUSE: 0    TOTAL_RESERVE: 0    TOTAL_FREE: 25   OTHERS: 0
 PROJECT                 SHARE   OWN  INUSE RESERVE FREE   DEMAND
 Lp1                    50.0 %   0     0    0       12       0
 Lp2                    50.0 %   0     0    0       13       0

FEATURE: hspice@siteB
 SERVICE_DOMAIN: SD2
 TOTAL_INUSE: 0    TOTAL_RESERVE: 0    TOTAL_FREE: 65   OTHERS: 0
 PROJECT                 SHARE   OWN  INUSE RESERVE FREE   DEMAND
 Lp1                    50.0 %   0     0    0       32       0
 Lp2                    50.0 %   0     0    0       33       0
```

e) Run **bhosts -s**.

Different resource information displays depending on the cluster locality of the features.

From `clusterA`:

```
RESOURCE                TOTAL       RESERVED       LOCATION
hspice                  36.0        0.0            host1
```

From `clusterB` in `siteB`:

```
RESOURCE                TOTAL       RESERVED       LOCATION
hspice                  76.0        0.0            host2
```

Example configuration: 2 sites and 4 service domains

Some of your service domains may have geographical restrictions when serving licenses. In this example, two clusters in one location can run `hspice` jobs. and 4 service domains are defined for the `hpsice` feature:

- `SD1` is a local license file for `clusterA` with 25 `hspice` licenses
- `SD2` is a local license file for `clusterB` with 65 `hspice` licenses
- `SD3` is a WANable license with 15 `hspice` licenses
- `SD4` is a globally WANable license with 7 `hspice` licenses

The geographical license checkout restrictions are:

- Jobs in `clusterA` can check out licenses from SD1 SD3 and SD4 but not SD2

- Jobs in `clusterB` can check out licenses from SD2 SD3 and SD4 but not SD1

```
Begin Feature
NAME = hspice
DISTRIBUTION = SD1 (Lp1 1 Lp2 1)
LOCAL_TO = clusterA
End Feature

Begin Feature
NAME = hspice
DISTRIBUTION = SD2 (Lp1 1 Lp2 1)
LOCAL_TO = clusterB
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 = clusterA
End Feature

Begin Feature
NAME = hspice
GROUP_DISTRIBUTION = group1
SERVICE_DOMAINS = SD2
LOCAL_TO = clusterB
End Feature

Begin Feature
NAME = hspice
GROUP_DISTRIBUTION = group1
SERVICE_DOMAINS = SD3 SD4
End Feature
```

Submit jobs that use locality

LOCAL_TO is configured in `lsf.licensescheduler`.

Job submission is simplified when locality is configured.

1. Specify the resource usage string with the same resource name you see in `bhosts -s`.

   No OR rusage string is needed.

   For example:

   bsub -Lp Lp1 -R "rusage[hspice=1]" myjob

How locality works with other settings

The following table shows various combinations of LOCAL_TO and other feature section parameters:

	NAME	FLEX_NAME
1	AppX	-
2	AppZ201	201-AppZ
3	AppB_v1	AppB

1. You can define different License Scheduler tokens for the same FlexNet feature. The the service domain names (in either the DISTRIBUTION line or the SERVICE_DOMAINS for group configurations) of the same FlexNet feature in different feature sections must be exclusive. They cannot overlap.
2. When LOCAL_TO is configured for a feature, you can define different License Scheduler tokens for the same FlexNet feature with different localities. The constraints are:
   - For the same FlexNet feature, service domains must be exclusive.
   - The location name of LOCAL_TO defines the locality of that feature, so the name must be unique for all tokens with same FlexNet feature.
   - You should use same location name for different FlexNet features with the same pattern of locality, but License Scheduler does not check whether the same location name of a different feature contains the same list of clusters.
3. Features must either have a different NAME or have LOCAL_TO defined. The service domains for each License Scheduler token of same FlexNet feature must be exclusive.

How locality works with ALLOCATION and ENABLE_INTERACTIVE

The LOCAL_TO parameter simplifies the ALLOCATION configuration. Most of the time you are only interested in who can participate to share a particular token. LOCAL_TO gives the equal share for all the clusters defined in LOCAL_TO and applies to all the projects. Use ALLOCATION to fine tune the shares for individual projects between different clusters:

- Except for the keyword `interactive`, all the cluster names defined in ALLOCATION must also be defined in the LOCAL_TO parameter.
- The global parameter ENABLE_INTERACTIVE and ALLOCATION with interactive share defined works same as before. If ALLOCATION is configured, it ignores the global setting of the ENABLE_INTERACTIVE parameter.
- If ALLOCATION is not defined, but LOCAL_TO is defined, the default value for ALLOCATION will be equal shares for all the clusters defined in LOCAL_TO parameter. This applies to all license projects defined in DISTRIBUTION or GROUP_DISTRIBUTION.
- If both ALLOCATION and LOCAL_TO are defined, ALLOCATION parameter can be used to fine tune the shares between the clusters for different projects.

The following table shows example configurations with two clusters and 12 `hspice` licenses distributed as follows:

```
DISTRIBUTION = LanServer (Lp1 1 Lp2 1)
```

ENABLE_INTERACTIVE	LOCAL_TO	ALLOCATION
No	SiteA(clusterA interactive)	—
No	clusterA	Lp1(clusterA 1 clusterB 0)
No	clusterA	Lp1(clusterA 1)\ Lp2(clusterA 1)

About interactive taskman jobs

The License Scheduler command `taskman` is a job starter for taskman jobs to use License Scheduler without `bsub`. `taskman` checks out a license token and manages interactive UNIX applications.

If LOCAL_TO is specified for a feature, `taskman` jobs need to specify feature names with locality information similar to submission with `bsub`. You need to know which token can be used from the location where task is going to run. For example:

```
taskman -Lp P1 -R "rusage[hspice@siteB=1]" myjob
taskman -Lp P1 -R "rusage[hspice=1]" myjob
taskman -Lp P1 -R "rusage[hspice@clusterA=1]" myjob
```

Project mode using project groups

Project groups use a `ProjectGroup` section to build a hierarchical project structure, and allow you to set limits on projects spanning multiple clusters.

Depending on your license usage, you can configure different project groups for different license features, or reuse the same hierarchical structure.

Each license feature in project mode can either use projects or project groups. Changing from projects to project groups involves adding a `ProjectGroup` section and changing the license token distribution configured in the `Feature` section. Other configuration remains the same.

Configure project groups

ProjectGroup sections use configured projects (each with a `Projects` section in the `lsf.licensescheduler` file) to form a hierarchical structure for each feature.

Note:

The `Feature` section GROUP parameter is used to group projects together, simplifying configuration, and is not the same as a `ProjectGroup` section.

1. Add a `ProjectGroup` section to the `lsf.licensescheduler` file:

```
Begin ProjectGroup
GROUP     SHARES    OWNERSHIP   LIMITS     NON_SHARED
End Projectgroup
```

   If `LS_FEATURE_PERCENTAGE=Y` or `LS_ACTIVE_PERCENTAGE=Y` in `lsf.licensescheduler`, values for OWNERSHIP, LIMITS, and NON_SHARED are expressed as a percentage of the total licenses, not as an absolute number.

2. For each branch in the hierarchy, add a line to the `ProjectGroup` section.
   a) Under the heading GROUP, indicate the project that branches, and direct decendants in the hierarchy **(*group*(*member* ...))**.
   b) Under the heading SHARES, set the integer share for each member project.
   c) Under the heading OWNERSHIP, set the integer ownership for each bottom-level group member (leaf node), with '-' representing no ownership. The `OWNERSHIP` value must be greater than or equal to the `NON_SHARED` value.
   d) Under the heading LIMITS set the integer license limit for each member project, with '-' representing unlimited. The `LIMITS` value must be greater than or equal to the `OWNERSHIP` value.
   e) Under the heading NON_SHARED, set the integer number of non-shared licenses each bottom-level group member (leaf node) uses exclusively, with '-' representing none.
   f) Optionally, under the heading DESCRIPTION, add a description up to 64 characters long, using '\' to extend to multiple lines.

   For example, the branch `g4` splits into three members:

```
GROUP                 SHARES    OWNERSHIP   LIMITS     NON_SHARED
(g4 (p4 p5 p6))       (1 1 1)   (1 1 1)     ()         (- 3 -)
```

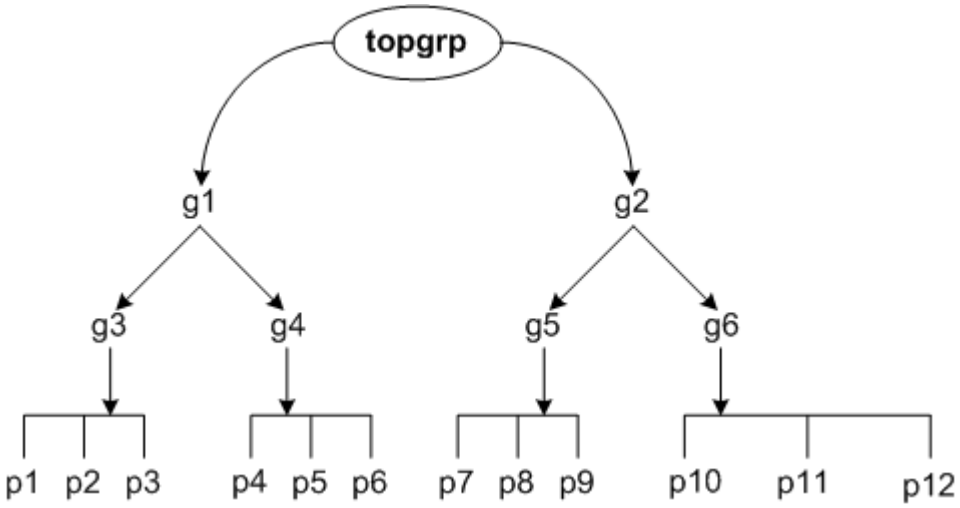
3. In the `Feature` section, set parameter GROUP_DISTRIBUTION to the top level of the `ProjectGroup` section hiearchy.

   The DISTRIBUTION parameter used for projects is no longer used.

4. In the `Feature` section, list service domains in the SERVICE_DOMAINS parameter.

Unlike for projects, service domains are not included in the distribution for project groups.

Project group examples



This hierarchy is implemented by the project group configuration:

```
Begin ProjectGroup
GROUP                SHARES    OWNERSHIP   LIMITS     NON_SHARED
(topgrp (g1 g2))     (1 1)     (- -)       (10 10)    (4 4)
(g1 (g3 g4))         (1 1)     (- -)       (10 10)    (- 4)
(g2 (g5 g6))         (1 1)     (- -)       (- 5)      (2 2)
(g3 (p1 p2 p3))      (1 1 2)   ()          (3 4 5)    ()
(g4 (p4 p5 p6))      (1 1 1)   (1 1 1)     ()         (- 3 -)
(g5 (p7 p8 p9))      (1 1 1)   (2 - 2)     ()         (1 - 1)
(g6 (p10 p11 p12))   (1 1 1)   (2 2 2)     (4 4 4)    (1 - 1)
End ProjectGroup
```

License feature configuration using this project group:

```
Begin Feature
NAME = AppZ
GROUP_DISTRIBUTION = topgrp
SERVICE_DOMAINS = LanServer WanServer
End Feature
```

Using the LIMITS column allows you to limit token use, so sometimes tokens are not distributed even if they are available. By default, License Scheduler distributes all available tokens if possible. For example, if total of 6 licenses are available:

```
Begin ProjectGroup
GROUP       SHARES OWNERSHIP LIMITS NON_SHARED
(Root(A B)) (1 1)   ()        ()     ()
(A (c d))   (1 1)   ()        (1 1)  ()
(B (e f))   (1 1)   ()        ()     ()
End ProjectGroup
```

When there is no demand for license tokens, License Scheduler only allocates 5 tokens according to the distribution. License Scheduler gives 3 tokens to group `A` and 3 tokens to group `B`, but project `c` and project `d` are limited to 1 token each, so 1 token will not be allocated within group `A`. As more demand comes in for project `e` and project `f`, the unallocated tokens are distributed to group `B`.

Configure preemption priority within project groups

The optional PRIORITY parameter in the `ProjectGroup` section, if defined, is used for preemption instead of basing preemption on the accumulated `inuse` for each project.

1. Under the heading PRIORITY, set the integer priority for each group member, with '0' being the lowest priority.

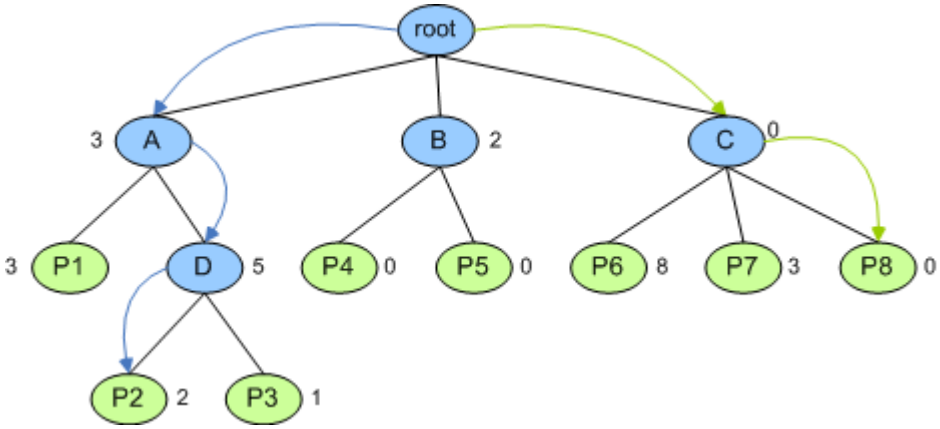
   PRIORITY can be set for all members in the project group hierarchy.

For example:

```
Begin ProjectGroup
GROUP             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 -)
(D (P2 P3))       (1 1)    ()         ()        ()       (2 1)
End ProjectGroup
```

By default, priority is evaluated from top to bottom. The priority of a given node is first decided by the priorities of its parent nodes. The values are only comparable between siblings.

The following figure illustrates the example configuration:



The priority of each node is shown beside the node name. If priority is not defined, by default is set to 0 (nodes P4 and P5 under node B).

To find the highest priority leaf node in the tree, Platform License Scheduler traverses the tree from root to node A to node D to project P2.

To find the lowest priority leaf node in the tree, License Scheduler traverses the tree from root to node C to project P8.

When two nodes have the same priority, for example, projects P4 and P5, priority is determined by accumulated `inuse` usage at the time the priorities are evaluated.

When a leaf node in branch A wants to preempt a token from branch B or C, branch C is picked because it has a lower priority than branch B.

View hierarchical configuration

1. Use `blinfo -G` to view the hierarchical configuration:

For the previous example:

blinfo -G

```
GROUP                 SHARES    OWNERSHIP   LIMITS     NON_SHARED
(topgrp (g1 g2))      (1 1)     (4 4)       (10 10)    (4 4)
(g1 (g3 g4))          (1 1)     (2 4)       (10 10)    (- 4)
(g2 (g5 g6))          (1 1)     (2 2)       (- 5)      (2 2)
(g3 (p1 p2 p3))       (1 1 2)   ()          (3 4 5)    ()
(g4 (p4 p5 p6))       (1 1 1)   (1 3 1)     ()         (- 3 -)
(g5 (p7 p8 p9))       (1 1 1)   (2 - 2)     ()         (1 - 1)
(g6 (p10 p11 p12))    (1 1 1)   (2 2 2)     (4 4 4)    (1 - 1)
```

Viewing information about project groups

1. Use `blstat -G` to view the hierarchical dynamic license information.

blstat -G

```
FEATURE: p1_f1
SERVICE_DOMAINS:
TOTAL_INUSE: 0    TOTAL_RESERVE: 0    TOTAL_FREE: 5    OTHERS: 0
SHARE_INFO_FOR:  /topgrp
GROUP/PROJECT           SHARE   OWN  INUSE RESERVE FREE   DEMAND
g2                      100.0 %   4     0    0        4     0
SHARE_INFO_FOR:  /topgrp/g2
GROUP/PROJECT           SHARE   OWN  INUSE RESERVE FREE   DEMAND
p3                      50.0 %   0     0    0        2     0
p4                      50.0 %   0     0    0        2     0
FEATURE: p1_f2
SERVICE_DOMAINS:
TOTAL_INUSE: 0    TOTAL_RESERVE: 0    TOTAL_FREE: 10   OTHERS: 0
SHARE_INFO_FOR:  /topgrp
GROUP/PROJECT          SHARE   OWN  INUSE RESERVE FREE   DEMAND
g2                     100.0 %   4     0    0        4     0
SHARE_INFO_FOR:  /topgrp/g2
GROUP/PROJECT          SHARE   OWN  INUSE RESERVE FREE   DEMAND
p3                     50.0 %   0     0    0        2     0
p4                     50.0 %   0     0    0        2     0
```

Using automatic time-based configuration

Variable time-based configuration is used in both project mode and cluster mode to automatically change configuration set in `lsf.licensescheduler` based on time windows. For example, if you have design centers in remote locations, one use of time-based configuration is to switch ownership of license tokens based on local time of day.

You define automatic configuration changes in `lsf.licensescheduler` by using `if-else` constructs and time expressions. After you change the files, reconfigure the cluster with the `bladmin reconfig` command.

The expressions are evaluated by License Scheduler every 10 minutes based on `bld` start time. When an expression evaluates true, License Scheduler dynamically changes the configuration based on the associated configuration statements and restarts `bld`

The `#if`, `#else`, `#endif` keywords are not interpreted as comments by License Scheduler, but as if-else constructs.

Syntax

```
time = hour | hour:minute | day:hour:minute
```

hour

integer from 0 to 23, representing the hour of the day.

minute

integer from 0 to 59, representing the minute of the hour.

If you do not specify the minute, License Scheduler assumes the first minute of the hour (:00).

day

integer from 0 to 7, representing the day of the week, where 0 represents every day, 1 represents Monday, and 7 represents Sunday.

If you do not specify the day, License Scheduler assumes every day. If you do specify the day, you must also specify the minute.

Specify time values

1. Specify at least the hour.

   Day and minutes are optional.

Specify time windows

1. Specify two time values separated by a hyphen (`-`), with no space in between.

   ```
   time_window = time1-time2
   ```

   time1 is the start of the window and *time2* is the end of the window. Both time values must use the same syntax.

   Use one of the following ways to specify a time window:

- *hour*-*hour*
- *hour*:*minute*-*hour*:*minute*
- *day*:*hour*:*minute*-*day*:*hour*:*minute*

For example:

- Daily window

  To specify a daily window omit the day field from the time window. Use either the `hour-hour` or `hour:minute-hour:minute` format. For example, to specify a daily 8:30 a.m. to 6:30 p.m window:

  ```
  8:30-18:30
  ```
- Overnight window

  To specify an overnight window make *time1* greater than *time2*. For example, to specify 6:30 p.m. to 8:30 a.m. the following day:

  ```
  18:30-8:30
  ```
- Weekend window

  To specify a weekend window use the day field. For example, to specify Friday at 6:30 p.m to Monday at 8:30 a.m.:

  ```
  5:18:30-1:8:30
  ```

Specify time expressions

Time expressions use time windows to specify when to change configurations.

1. Define a time expression.

   A time expression is made up of the `time` keyword followed by one or more space-separated time windows enclosed in parenthesis. Time expressions can be combined using the `&&`, `||`, and `!` logical operators.

   ```
   expression = time(time_window[ time_window ...])
              | expression && expression
              | expression || expression
              | !expression
   ```

   For example:

   Both of the following expressions specify weekends (Friday evening at 6:30 p.m. until Monday morning at 8:30 a.m.) and nights (8:00 p.m. to 8:30 a.m. daily).

   ```
   time(5:18:30-1:8:30 20:00-8:30)
   time(5:18:30-1:8:30) || time(20:00-8:30)
   ```

Create if-else constructs

The if-else construct can express single decisions and multi-way decisions by including elif statements in the construct.

- Define an if-else expression.

  ```
  #if time(expression)
  statement
  #else
  statement
  #endif
  ```

  The `#endif` part is mandatory and the #else part is optional.

- Define an elif expression.

  The `#elif` expressions are evaluated in order. If any expression is true, the associated statement is used, and this terminates the whole chain.

  The `#else` part handles the default case where no other conditions are satisfied.

```
#if time(expression)
statement
#elif time(expression)
statement
#elif time(expression)
statement
#else
statement
#endif
```

  When you use `#elif`, the `#else` and `#endif` parts are required.

Restart to implement configuration changes

All time-based configuration is within the `lsf.licensescheduler` file, so restarting the `bld` applies all changes.

1. Run **bladmin ckconfig** to check configuration.
2. Run **lsadmin limrestart** or **bladmin restart** to restart the `bld`.

Verify configuration

Verify time-based configuration by viewing Platform License Scheduler information.

1. Run **blinfo**.
2. Run **blstat**.

Examples

Project configuration in project mode

```
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
```

Project group configuration in project mode

```
#
# ProjectGroup section
#
Begin ProjectGroup
GROUP             SHARES   OWNERSHIP  LIMITS     NON_SHARED
(group1 (A B))    (1 1)    (5 -)         ()         ()
End ProjectGroup

Begin ProjectGroup
GROUP             SHARES   OWNERSHIP  LIMITS     NON_SHARED
(group2 (A B))    (1 1)    (- 5)         ()         ()
End ProjectGroup
```

```
#
# Feature section
#
Begin Feature
NAME = f1
#if time(5:16:30-1:8:30 20:00-8:30)
GROUP_DISTRIBUTION=group1
#elif time(3:8:30-3:18:30)
GROUP_DISTRIBUTION=group2
#else
GROUP_DISTRIBUTION=group2
#endif
SERVICE_DOMAINS=Lan1 Lan2
End Feature
```

Cluster distribution configuration in cluster mode

```
Begin Feature
NAME = f1
#if time(5:16:30-1:8:30 20:00-8:30)
CLUSTER_DISTRIBUTION=Wan(Cl1 1 Cl2 1)
#elif time(3:8:30-3:18:30)
CLUSTER_DISTRIBUTION= Wan(Cl1 2 Cl2 1/2/100) Lan(Cl2 1)
#else
CLUSTER_DISTRIBUTION= Wan(Cl1 10 Cl2 1/1/10) Lan(Cl1 1)
#endif
End Feature
```

Failover

License maximization

The built-in functionality of License Scheduler helps ensure that your licenses are always being used efficiently. For example, if the `sbatchd` encounters any problems, the job acquires the state `UNKNOWN`. However, License Scheduler ensures that any in use licenses continue to be allocated, but charges them to the `OTHERS` category until the `sbatchd` recovers and the job state is known again.

failover host

A master candidate host that runs the License Scheduler daemon (`bld`), and can take over license management if the master License Scheduler host fails or loses its connection to the network (in either a LAN or WAN environment).

failover provision

The configuration of a list of failover hosts in the event of a host failure or network breakdown. License Scheduler can be configured for failover provision in both LANs and WANs.

Failover provisioning for LANs

Configuring failover ensures enhanced performance and reliable license distribution.

You only need one host to run Platform License Scheduler, but you can configure your site for a failover mechanism with multiple candidate hosts to take over the scheduling in case of a failure. This configuration can be used in a local network or across multiple sites in a wider network.

1. Define the list of License Scheduler hosts in `LSF_CONFDIR/lsf.conf` and lsf.licensescheduler for your LAN (Designer Center A in this example).
   a) `lsf.conf`: Specify a space-separated list of hosts for the LSF_LIC_SCHED_HOSTS parameter:

      LSF_LIC_SCHED_HOSTS="hostA.designcenter_a.com hostB.designcenter_a.com hostC.designcenter_a.com"

      Tip:

      List the hosts in order of preference for running Platform License Scheduler, from most preferred to least preferred.

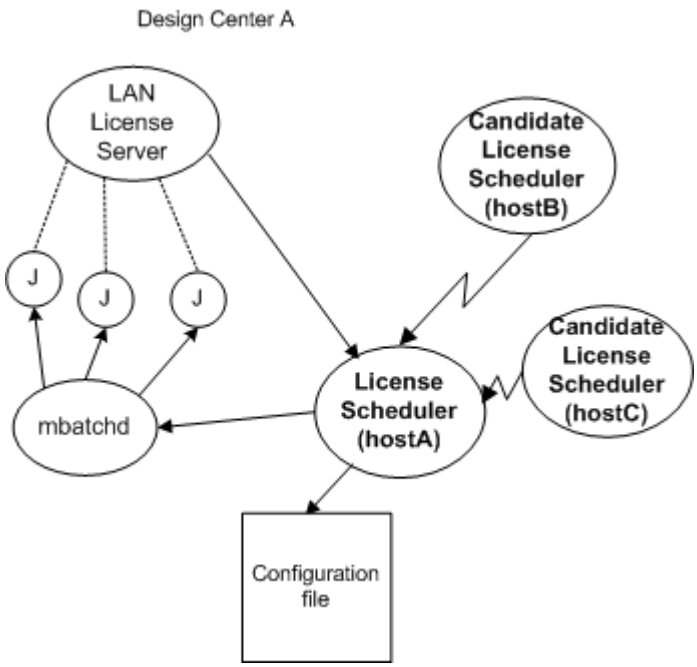
   b) `lsf.licensescheduler`: Specify a space-separated list of hosts for the `HOSTS` parameter:

      HOSTS=hostA.designcenter_a.com hostB.designcenter_a.com hostC.designcenter_a.com

      List the hosts in the same order as `lsf.conf`.

   The LIM starts the `bld` (License Scheduler daemon) on each host in the LSF_LIC_SCHED_HOSTS list.

   Every host in defined in LSF_LIC_SCHED_HOSTS is a failover candidate and runs the `bld` daemon.



- hostA.designcenter_a.com is the License Scheduler host, and the remaining hosts are candidate hosts running the `bld` daemon, ready to take over the management of the licenses in case of a network failure
- Each host contains the list of candidate hosts in memory
- Each candidate License Scheduler host communicates with the License Scheduler host, License Scheduler (hostA)
- If the License Scheduler host fails, each candidate host checks to see if a more eligible host is running the License Scheduler daemon. If not, it becomes the failover host and inherits the communication links that existed between the original License Scheduler host and each candidate host. In this example, if License Scheduler on hostA fails, Candidate License Scheduler hostB is the next most eligible host, and takes over the license scheduling.

Failover provisioning for WANs

Similar to LANs, you can configure your site for a failover mechanism across multiple sites in a wide network.

You only need one host to run License Scheduler, but you can configure your site for a failover mechanism with multiple candidate hosts to take over the scheduling in case of a failure.

License scheduling across sites can be streamlined because License Scheduler supports service provisioning during breaks in wide area network connections. This allows you to run License Scheduler from one host that controls license scheduling across multiple sites.

Configure and start Platform License Scheduler in a WAN

In a WAN configuration:

1. As the root user, install Platform License Scheduler on each cluster in the WAN configuration and select one cluster to be the main cluster.

2. In the cluster that contains the WAN license server, log on as the primary License Scheduler administrator.
3. Edit the following items in `LSF_CONFDIR/lsf.licensescheduler`:
   a) Specify a space-separated list of hosts for the `HOSTS` parameter:

      `HOSTS=`hostname_1 hostname_2 ... hostname_n

      Where:

      hostname_1 is the most preferred host for running Platform License Scheduler.

      hostname_n is the least preferred host for running Platform License Scheduler.
   b) In the `Clusters` section, specify the names of the clusters in the WAN.

      For example:

      ```
      Begin Clusters
      CLUSTERS
      design_SJ
      design_BOS
      End Clusters
      ```
4. In the cluster that contains the WAN license server, as the LSF primary administrator, edit `LSF_CONFDIR/lsf.conf`. Lines that begin with # are comments:

   Specify a space-separated list of hosts for the `LSF_LIC_SCHED_HOSTS` parameter:

   `LSF_LIC_SCHED_HOSTS="`*hostname_1 hostname_2 ... hostname_n*`"`

   Where:

   hostname_1, *hostname_2*, ..., *hostname_n* are hosts on which the LSF LIM daemon starts the Platform License Scheduler daemon (`bld`).

   The first host listed in the `HOSTS` list is the default master License Scheduler host for the WAN.

   The order of the host names in `LSF_LIC_SCHED_HOSTS` is ignored.
5. In the other clusters in the WAN:
   a) Configure the `LSF_LIC_SCHED_HOSTS` parameter in `lsf.conf` with a local list of candidate hosts.
   b) Configure the `HOSTS` parameter in the `Parameters` section `lsf.licensescheduler` with the following list of hosts:

      - Start the list with the same list of candidate hosts as the `HOSTS` parameter in the cluster that contains the WAN license server.
      - Continue the list with the local cluster's list of hosts from the `LSF_LIC_SCHED_HOSTS` parameter in `lsf.conf`.
6. In the cluster that contains the WAN license server and the other clusters in the WAN, run the following commands:
   a) Run `bld -C` to test for configuration errors.
   b) Run `bladmin reconfig` to configure License Scheduler.
   c) Run `lsadmin reconfig` to reconfigure LIM.
   d) Use `ps -ef` to make sure that `bld` is running on the candidate hosts.
   e) Run `badmin reconfig` to reconfigure `mbatchd`.

Tip:

Although the `bld` daemon is started by LIM, `bld` runs under the account of the primary License Scheduler administrator. If you did not

configure the LIM to automatically start the `bld` daemon on your License Scheduler hosts, run `$LSF_BINDIR/blstartup` on each host to start the `bld` daemon.

WAN example

A design center contains the following hosts configuration in a WAN:

LIM starts `bld` on the following hosts:

- `lsf.conf` in Design Center A

```
LSF_LIC_SCHED_HOSTS="hostA1.designcenter_a.com hostA2.designcenter_a.com
hostA3.designcenter_a.com"
```

- `lsf.conf` in Design Center B

```
LSF_LIC_SCHED_HOSTS="hostB1.designcenter_b.com hostB2.designcenter_b.com
hostB3.designcenter_b.com"
```

License Scheduler candidate hosts are listed in the following order of preference:

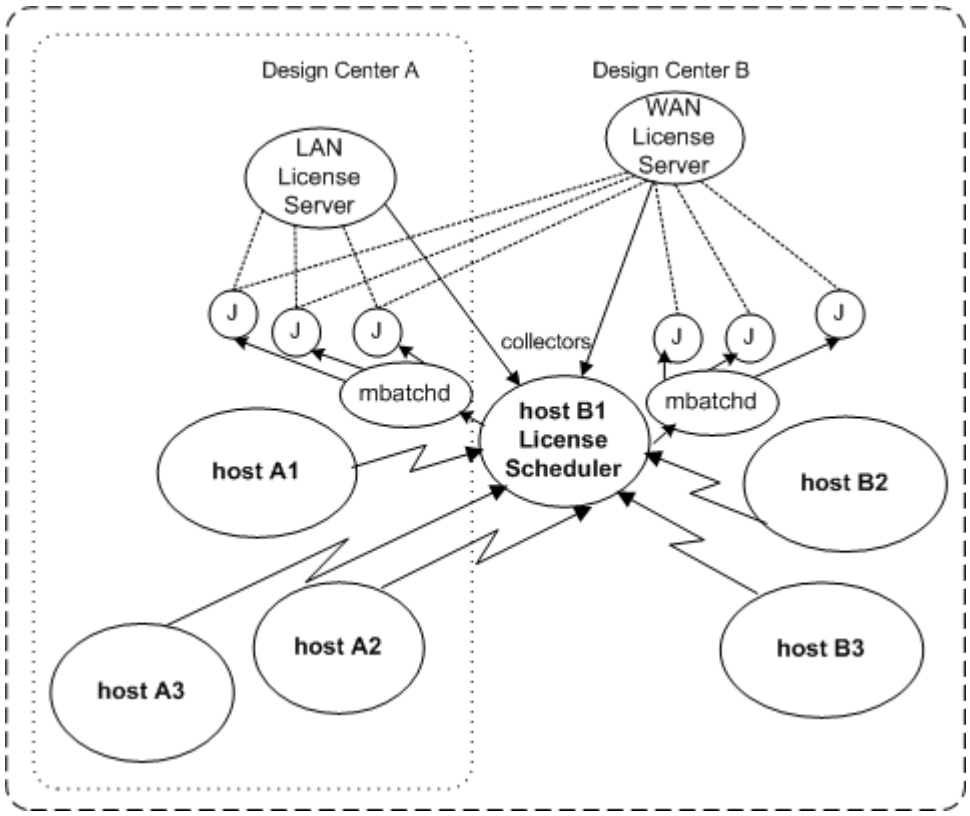
- `lsf.licensescheduler` in Design Center A

```
HOSTS=hostB1.designcenter_b.com hostB2.designcenter_b.com
hostA1.designcenter_a.com hostA2.designcenter_a.com hostA3.designcenter_a.com
```

- `lsf.licensescheduler` in Design Center B

```
HOSTS=hostB1.designcenter_b.com hostB2.designcenter_b.com hostB3.designcenter_b.com
```

The following diagram shows hostB1.designcenter_b.com, the License Scheduler host for the WAN containing Design Center A and Design Center B.



How it works

The LSF LIM daemon starts the License Scheduler daemon (`bld`) on each host listed in LSF_LIC_SCHED_HOSTS in Design Center A and Design Center B.

Each host in the HOSTS list in Design Center A is a potential License Scheduler candidate in Design Center A and is running the `bld` daemon, but only one host becomes the License Scheduler host—the first host in the `HOSTS` list that is up and that is running the `bld` daemon. Similarly, the License Scheduler host in Design Center B is the first host in the `HOSTS` list that is up and that is running the `bld` daemon.

License Scheduler manages the licenses in Design Center A and Design Center B as follows:

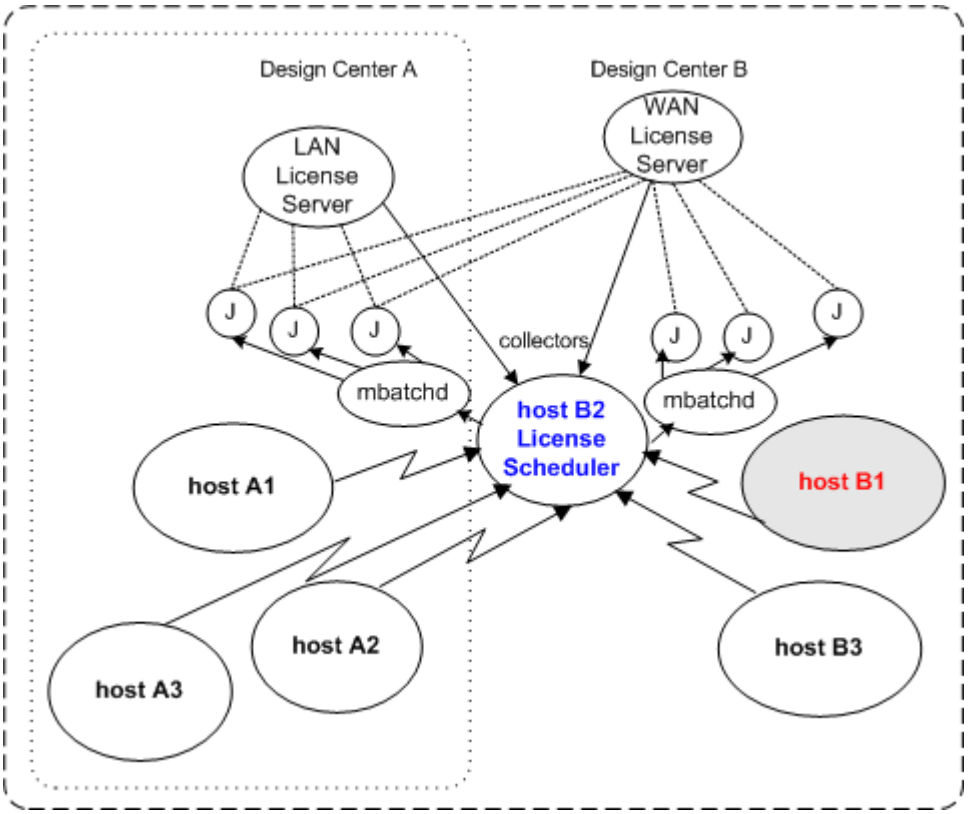
- Both design centers list hostB1.designcenter_b.com at the top of their HOSTS lists.
- hostB1.designcenter_b.com is the License Scheduler host for Design Center A and for Design Center B.
- The rest of the hosts in both design centers remain on standby as candidate License Scheduler hosts.
- License Scheduler manages the license scheduling across the WAN connection.

Service provisioning at the host and network levels

In the above example configuration, there are two potential points of failure: host and network.

Host failure

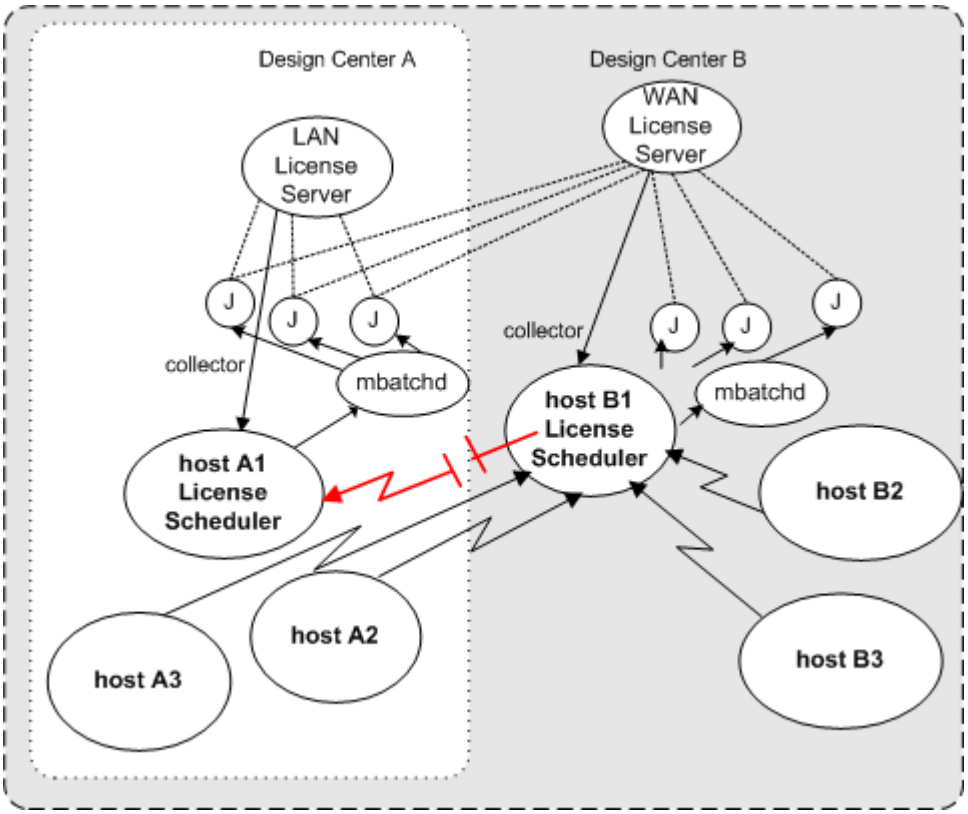
If hostB1.designcenter_b.com fails, and `bld` stops running, a candidate License Scheduler host must take over the license management. The next host on the HOSTS list in both Design Center A and Design Center B is hostB2designcenter_b.com. License Scheduler fails over to this host if it is up and running.



Network failure

If the network connection between Design Center A and Design Center B breaks, Design Center A can no longer communicate with the hosts in Design Center B, so hostB1.designcenter_b.com and hostB2.designcenter_b.com are no longer candidate license scheduling hosts for Design Center A. The next candidate host for Design Center A is hostA1.designcenter_a.com. License management then runs locally in Design Center A on hostA1.designcenter_a.com. In Design Center B, hostB1.designcenter_b.com continues to run License Scheduler, but only manages the local network as long as the wide area network connection is down.

The local License Scheduler host, hostA1.designcenter_a.com, checks for a heartbeat from hostB1.designcenter_b.com at regular intervals, then returns license management back to hostB1.designcenter_b.com when the network connection returns.



Set up fod

The `fod` daemon manages failover for the `blcollect` daemons. `fod` can restart any failed `blcollect` processes if the local host (and thus the local `fod`) is down . The failover host `fod` starts new `blcollect` daemons until the primary host comes back online and the primary `fod` contacts the secondary `fod`.

The fod files are in the Platform License Scheduler package, but must be copied, configured, and started manually.

1. Install the failover daemon (`fod`) files on each host.
   a) Create a new directory to hold the fod files, with subdirectories `bin`, `conf`, `etc`, and `man`.

For example: `/usr/local/fod`

b) Copy all user command files and the `fod.shell` file to `.../bin`.
c) Copy the `fod.conf` file to `.../conf`.
d) Copy the `fod` file to `.../etc`.
e) Copy the `fodapps.1`, `fodhosts.1` and `fodid.1` files to `.../man/man1`.
f) Copy the `fod.conf.5` file to `.../man/man5`.
g) Copy the `fodadmin.8` file to `.../man/man8`.

2. Edit the `fod.shell` file, and set the FOD_ROOT parameter to the name of your new directory.

   For example: **FOD_ROOT=/usr/local/fod**

3. Set environment variables.
   a) Set the PATH environment variable to include the `/bin` directory.
   b) Set the FOD_ENVDIR environment variable to `$FOD_ROOT/conf`.
   c) Set the MANPATH environment variable to include the `/man` directory.
4. In `fod.conf`, set required parameters.

   - FOD_ADMIN: The License Scheduler administrator
   - FOD_PORT: The TCP listening port and UDP port for the failover daemon
   - FOD_WORK_DIR: The working directory
   - FOD_LOG_DIR: The log directory

   For example:

   FOD_CLUSTERNAME = fod

   FOD_ADMIN = lsadmin

   FOD_PORT = 9583

   FOD_WORK_DIR = /usr/local/fod/work

   FOD_LOG_DIR = /usr/local/fod/work

5. In the Hosts section of `fod.conf` specify the hosts where the failover daemons runs.

   If your hosts run in different DNS domains, you must use a fully-qualified domain name when specifying the host name. The first host in the Hosts section is the first host on which the failover daemon will run (the master failover daemon host).

   For example:

```
Begin Hosts
HOSTNAME
fodhost1.domain_name
fodhost2
End Host
```

6. Modify the Applications section of `fod.conf`.

```
Begin Applications
NAME         PATH                                    PARAMS     FATAL_EXIT_VALUE
blcollect   /pcc/apps/lsf6/6.0/sparc-sol7-64/etc   (-2 -m "sasun3 augustus claudius" -p 9581 -c
lan -i 20 -D /sparc-sol7-64/etc)   (-)
End Applications
```

7. Start the fod on each host.
   a) Log on as the Platform License Scheduler administrator.
   b) Source the LSF environment.

      - For `csh` or `tsh` run **source LSF_TOP/conf/cshrc.lsf**

- For `sh`, `ksh`, or `bash`, run **. LSF_TOP/conf/profile.lsf**

c) Launch the failover daemons by running the `fod.shell` file.

Check the progress of a successful launch using `ps -ef`.

View the `fod` log under $LSF_LOGDIR.

Check configuration from `$FOD_ROOT/etc` using `fod -C`.

CHAPTER

5

Viewing information and troubleshooting

About viewing available licenses

The license server collects license feature information from physical servers and merges this data together into a service domain. After merging, the individual license server information is retained and you can view this information together with the physical server information.

The licenses in use have been checked out from FlexNet by your projects. Free licenses and licenses reserved by a project have not yet been checked out from FlexNet.

The total number of licenses could change as licenses expire or are added. As non-LSF users check out licenses, the OTHERS count in `blstat` should increase and the TOTAL_FREE count decreases. The number of licenses for each project changes whenever LSF redistributes license tokens among competing projects.

View license server and license feature information passed to jobs

You can display the license servers used by each service domain allocated to the license features.

1. Run **blstat -S**.

```
blstat -S
FEATURE: feature1
SERVICE_DOMAIN: domain1
SERVERS     INUSE  FREE
 server1      1      0
 server2      0      1
 TOTAL        1      1
SERVICE_DOMAIN: domain2
SERVERS     INUSE  FREE
 server3      1      0
 TOTAL        1      0
```

The license feature `feature1` is assigned to `server1` and `server2` in the `domain1` service domain and `server3` in the `domain2` service domain. A job uses the feature1 license feature when the job is submitted with `"rusage[feature1=1]"` as the `rusage` string.

View license usage

1. Run `blstat -s` to display license usage.

```
blstat -s
FEATURE: p1_f2
SERVICE_DOMAIN: app_1 TOTAL_LICENSE: 10
LSF_USE LSF_DESERVE LSF_FREE   NON_LSF_USE NON_LSF_DESERVE  NON_LSF_FREE
     0       10         10          0           0                 0
FEATURE: p1_f1
SERVICE_DOMAIN: app_1 TOTAL_LICENSE: 5
LSF_USE LSF_DESERVE LSF_FREE   NON_LSF_USE NON_LSF_DESERVE  NON_LSF_FREE
     0       5          5           0           0                 0
```

If there are any distribution policy violations, `blstat` marks these with an asterisk (`*`) at the beginning of the line.

View workload distribution information

1. Run `blinfo -a` to display WORKLOAD_DISTRIBUTION information.

```
blinfo -a
FEATURE      SERVICE_DOMAIN  TOTAL  DISTRIBUTION
g1           LS              0      [p1, 50.0%] [p2, 50.0%]
```

```
                                        WORKLOAD_DISTRIBUTION
                                        [LSF 66.7%, NON_LSF 33.3%]
```

Sort license feature information

You can sort license feature information alphabetically, by total licenses, or by available licenses.

The value of total licenses is calculated using the number of licenses LSF workload deserves from all service domains that supply licenses to the feature, regardless of whether non-LSF workload has borrowed licenses from LSF workload.

- Sort alphabetically:

  blstat -o alpha
- Sort by total licenses:

  blstat -o total

  The feature with the largest number of total licenses displays first.
- Sort by available licenses:

  blstat -o avail

  The feature with the largest number of available licenses displays first.

  You can also run `blstat -o` with options `-Lp`, -t, `-D`, `-G`, `-s`, `-S`.

Note:

The values of "total licenses" and "licenses available" are calculated differently when `blstat -o` is used with different options:

- Options `-Lp`, -t, `-D`, `-G`: Total licenses means the sum of licenses that are allocated to LSF workload from all the service domains configured to supply licenses to the feature. Licenses borrowed by non-LSF workload are subtracted from this sum.
- Options `-s`, `-S`: Total licenses means all the licenses (supplied by the license vendor daemon) from all the service domains configured to supply licenses to that feature.

About error logs

Error logs maintain important information about License Scheduler operations.

Tip:

Log files grow over time. These files should occasionally be cleared, (manually or using automatic scripts).

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

The location of log files is specified with the parameter LSF_LOGDIR in `lsf.conf`.

The error log file names for the LSF License Scheduler system daemons are:

- `bld.log.`*host_name*
- `blcollect.log.`*host_name*

About blcollect log messages

Messages logged by `blcollect` include the following information:

- Time: The message log time.
- `blcollect` name: The service domain name, which is the license server host name , accessed by blcollect as defined in `lsf.licensescheduler`.
- Status report for feature collection: `blcollect` information gathered successfully or not.
- Detailed information: The number of tokens, the name of tokens, the license server name for license tokens collected by `blcollect`.

Manage log files

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.

1. Set LS_LOG_MASK in `lsf.licensescheduler` to the desired logging level.

   Note:

   If LS_LOG_MASK is not defined, 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.

   Log levels (highest to lowest):

   - LOG_WARNING: Default. Essential error messages only.
   - LOG_DEBUG: Fewest number of debug messages, very useful for debugging a problem.
   - LOG_DEBUG1: More debug messages than LOG_DEBUG.
   - LOG_DEBUG2: Most frequently used debug level.
   - LOG_DEBUG3: All debug messages. Use sparingly.

   Messages logged at the specified level and higher are recorded, while lower level messages are discarded.
2. Clean out or back up log files periodically.

Temporarily change the log level

You must submit the commands from the host on which the daemon is running (only applicable to the `bld`).

You can temporarily change the class or message log level for the `bld` and `blcollect` daemons without changing `lsf.licensescheduler`.

The message log level you set is in effect from the time you set it until you turn it off or the daemon stops running, whichever is sooner. If the daemon is restarted, its message log level is reset back to the value of LS_LOG_MASK and the log file is stored in the directory specified by LSF_LOGDIR.

1. Set the log level for the `bld`.

   bladmin blddebug [**-l** *debug_level*] [**-c** *class_name*]

   For example:

   bladmin blddebug -l 1 -c "LC_TRACE LC_FLEX"

   Logs messages for `bld` running on the local host and sets the log message level to LOG_DEBUG1. The log class is LC_TRACE LC_FLEX.
2. Set the log level for `blcollect`.

   bladmin blcdebug [**-l** *debug_level*] *collector_name ...* | **all**

   For example:

   bladmin blcdebug -l 3 all

   The log mask of all collectors is changed to LOG_DEBUG3.
3. Return the debug settings to their configured values (set with LS_LOG_MASK in `lsf.licensescheduler`).

   bladmin blddebug -o

   bladmin blcdebug -o

For a detailed description of these commands and their options, see the Platform LSF Command Reference.

Troubleshooting

Techniques

- Run `blstat` to check the current license usage information.
- Run `blusers` to check the current job and license usage. This information is the set intersection of License Scheduler Jobs and FlexNet information.
- Run `blinfo` command to check the current License Scheduler configuration.
- Run `BLD -C` to run a check to see if configuration is correct. This action in conjunction with LOG_DEBUG writes detailed configuration settings to the debug log.
- Turn on debugging by setting LSF_LOG_MASK=LOG_DEBUG and reconfiguring the daemon with `bladmin reconfig all`.
- Set the log class for mbatchd debug (LSB_DEBUG_MBD) in `lsf.conf`: LC_LICSCHED.
- Timing information can be logged using LSB_TIME_SCH=timelevel (similar to LSB_TIME_MBD) in `lsf.conf`.
- Run `bhosts -s` to check if resources are being reported correctly to LSF.

File locations

- `BLD` logs are in the standard `$LSF_LOGDIR`.
- `BLCOLLECT` logs are in `/tmp` or `$LSF_LOGDIR` on the hosts the daemon is running.
- Core files from `BLD`, `BLCOLLECT`, `mbatchd`, `lim`, and `mbsched` are located in `/tmp` on the daemon local hosts.

Check if lmstat is supported by blcollect

1. Create shell script to output (for example, echo) target `lmstat` output.
2. Point LMSTAT_PATH in `lsf.licensescheduler` to the shell script.
3. If LIC_COLLECTOR is not set, restart the `bld` to restart `blcollect`. If LIC_COLLECTOR is set, kill `blcollect` and restart `blcollect` manually.
4. Observe the `blcollect` log to view if there are any errors to determine whether `blcollect` is able to parse `lmstat` output properly.

CHAPTER

6

Reference

lsf.licensescheduler

The `lsf.licensescheduler` file contains Platform License Scheduler configuration information. All sections except `ProjectGroup` are required. In cluster mode, the `Project` section is also not required.

Changing lsf.licensescheduler configuration

After making any changes to `lsf.licensescheduler`, run the following commands:

- `bladmin reconfig` to reconfigure `bld`
- If you made the following changes to this file, you may need to restart `mbatchd`:
  - Added or deleted any feature.
  - Added or deleted projects in the `DISTRIBUTION` parameter of the `Feature` section.
  - Changed the `ProjectGroup` section.
  - Changed the feature mode (for example, changed from cluster mode to project mode, or vice versa).

  In these cases a message is written to the log file prompting the restart.

  If you have added, changed, or deleted any `Feature` or `Projects` sections, you may need to restart `mbatchd`. In this case a message is written to the log file prompting the restart.

  If required, run **badmin mbdrestart** to restart each LSF cluster.

Parameters section

Description

Required. Defines Platform 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. Mandatory parameters are as follows:

```
Begin Parameters
ADMIN=lsadmin
HOSTS=hostA hostB hostC
LMSTAT_PATH=/etc/flexlm/bin
LM_STAT_INTERVAL=30
PORT=9581
End Parameters
```

Parameters

- ADMIN
- AUTH
- CLUSTER_MODE
- DISTRIBUTION_POLICY_VIOLATION_ACTION
- ENABLE_INTERACTIVE
- HEARTBEAT_INTERVAL

- HEARTBEAT_TIMEOUT
- HIST_HOURS
- HOSTS
- INUSE_FROM_RUSAGE
- LIB_RECVTIMEOUT
- LIB_CONNTIMEOUT
- LICENSE_FILE
- LM_REMOVE_INTERVAL
- LM_STAT_INTERVAL
- LM_STAT_TIMEOUT
- LMSTAT_PATH
- LOG_EVENT
- LOG_INTERVAL
- LS_DEBUG_BLC
- LS_DEBUG_BLD
- LS_ENABLE_MAX_PREEMPT
- LS_LOG_MASK
- LS_MAX_STREAM_FILE_NUMBER
- LS_MAX_STREAM_SIZE
- LS_MAX_TASKMAN_PREEMPT
- LS_MAX_TASKMAN_SESSIONS
- LS_STREAM_FILE
- LS_PREEMPT_PEER
- MBD_HEARTBEAT_INTERVAL
- MBD_REFRESH_INTERVAL
- MERGE_BY_SERVICE_DOMAIN
- PEAK_INUSE_PERIOD
- PORT
- PREEMPT_ACTION
- STANDBY_CONNTIMEOUT
- BLC_HEARTBEAT_FACTOR

ADMIN

Syntax

ADMIN=*user_name ...*

Description

Defines the Platform License Scheduler administrator using a valid UNIX user account. You can specify multiple accounts.

Used for both project mode and cluster mode.

AUTH

Syntax

AUTH=Y

Description

Enables Platform License Scheduler user authentication for projects for `taskman` jobs.

Used for both project mode and cluster mode.

CLUSTER_MODE

Syntax

CLUSTER_MODE=Y

Description

Enables cluster mode (instead of project mode) in Platform License Scheduler. Setting in individual `Feature` sections overrides the global setting in the `Parameters` section.

Cluster mode emphasizes high utilization of license tokens above other considerations such as ownership. License ownership and sharing can still be configured, but within each cluster instead of across multiple clusters. Preemption of jobs (and licenses) also occurs within each cluster instead of across clusters.

Cluster mode was introduced in Platform License Scheduler 8.0. Before cluster mode was introduced, project mode was the only choice available.

Default

Undefined (N). Platform License Scheduler runs in project mode.

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 Platform License Scheduler checks for distribution policy violations.

reporting_command

Specify the keyword `CMD` with the directory path and command that Platform License Scheduler runs when reporting a violation.

Description

Optional. Defines how Platform License Scheduler handles distribution policy violations. Distribution policy violations are caused by non-LSF workloads; Platform License Scheduler explicitly follows its distribution policies.

Platform 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 Platform 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.

Used for project mode only.

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. Platform 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. Platform License Scheduler allocates licenses equally to each cluster and does not distribute licenses for interactive tasks.

Used for project mode only.

HEARTBEAT_INTERVAL

Syntax

HEARTBEAT_INTERVAL=seconds

Description

The time interval between `bld` heartbeats indicating the `bld` is still running.

Default

60 seconds

HEARTBEAT_TIMEOUT

Syntax

HEARTBEAT_TIMEOUT=seconds

Description

The time a slave bld waits to hear from the master `bld` before assuming it has died.

Default

120 seconds

HIST_HOURS

Syntax

HIST_HOURS=hours

Description

Determines the rate of decay the accumulated use value used in fairshare and preemption decisions. When **HIST_HOURS=0**, accumulated use is not decayed.

Accumulated use is displayed by the `blstat` command under the heading `ACUM_USE`.

Used for project mode only.

Default

5 hours. Accumulated use decays to 1/10 of the original value over 5 hours.

HOSTS

Syntax

HOSTS=*host_name.domain_name ...*

Description

Defines Platform License Scheduler hosts, including Platform License Scheduler candidate hosts.

Specify a fully qualified host name such as `hostX.mycompany.com`. You can omit the domain name if all your Platform License Scheduler clients run in the same DNS domain.

Used for both project mode and cluster mode.

INUSE_FROM_RUSAGE

Syntax

INUSE_FROM_RUSAGE=Y|N

Description

When not defined or set to **N**, the INUSE value uses `rusage` from `bsub` job submissions merged with license checkout data reported by `blcollect` (as reported by `blstat`).

When **INUSE_FROM_RUSAGE=Y**, the INUSE value uses the `rusage` from `bsub` job submissions instead of waiting for the blcollect update. This can result in faster reallocation of tokens when using dynamic allocation (when ALLOC_BUFFER is set).

When for individual license features, the `Feature` section setting overrides the global `Parameters` section setting.

Used for cluster mode only.

Default

N

LIB_CONNTIMEOUT

Syntax

LIB_CONNTIMEOUT=*seconds*

Description

Specifies a timeout value in seconds for communication between Platform License Scheduler and Platform LSF APIs. **LIB_CONNTIMEOUT=0** indicates no timeout.

Used for both project mode and cluster mode.

Default

5 seconds

LIB_RECVTIMEOUT

Syntax

LIB_RECVTIMEOUT=*seconds*

Description

Specifies a timeout value in seconds for communication between Platform License Scheduler and LSF.

Used for both project mode and cluster mode.

Default

5 seconds

LICENSE_FILE

Syntax

LICENSE_FILE=*path*

Description

Sets the path to the Platform License Scheduler license file containing a valid license for this product.

Used for both project mode and cluster mode.

Default

Uses the path set in LSF_LICENSE_FILE in the LSF configuration file `lsf.conf`.

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 (using preemption). `lmremove` causes `lmgrd` and vendor daemons to close the TCP connection with the application, then retries the license checkout.

License Scheduler only considers preepting a job after this interval has elapsed. LM_REMOVE_INTERVAL overrides the LS_WAIT_TO_PREEMPT value if LM_REMOVE_INTERVAL is larger.

Used for project mode only.

Default

180 seconds

LM_STAT_INTERVAL

Syntax

LM_STAT_INTERVAL=*seconds*

Description

Defines a time interval between calls that Platform License Scheduler makes to collect license usage information from FlexNet license management.

Used for both project mode and cluster mode.

Default

60 seconds

LM_STAT_TIMEOUT

Syntax

LM_STAT_TIMEOUT=*seconds*

Description

Sets the timout value passed to the `lmstat` command. The `Parameters` section setting is overwritten by the `ServiceDomain` setting, which is overwritten by the command line setting (`blcollect -t timeout`).

Used for both project mode and cluster mode.

Default

180 seconds

LMSTAT_PATH

Syntax

LMSTAT_PATH=*path*

Description

Defines the full path to the location of the FlexNet command `lmstat`.

Used for both project mode and cluster mode.

LOG_EVENT

Syntax

LOG_EVENT=Y

Description

Enables logging of Platform License Scheduler events in the `bld.stream` file.

Default

Not defined. Information is not logged.

LOG_INTERVAL

Syntax

LOG_INTERVAL=*seconds*

Description

The interval between token allocation data logs in the data directory

Default

60 seconds

LS_DEBUG_BLC

Syntax

LS_DEBUG_BLC=*log_class*

Description

Sets the debugging log class for the Platform License Scheduler `blcollect` daemon.

Used for both project mode and cluster mode.

Specifies the log class filtering to be applied to `blcollect`. Only messages belonging to the specified log class are recorded.

LS_DEBUG_BLC sets the log class and is used in combination with LS_LOG_MASK, which sets the log level. For example:

```
LS_LOG_MASK=LOG_DEBUG LS_DEBUG_BLC="LC_TRACE"
```

To specify multiple log classes, use a space-separated list enclosed in quotation marks. For example:

```
LS_DEBUG_BLC="LC_TRACE"
```

You need to restart the `blcollect` daemons after setting LS_DEBUG_BLC for your changes to take effect.

Valid values

Valid log classes are:

- LC_AUTH and LC2_AUTH: Log authentication messages
- LC_COMM and LC2_COMM: Log communication messages
- LC_FLEX - Log everything related to FLEX_STAT or FLEX_EXEC Flexera APIs
- LC_LICENSE and LC2_LICENSE : Log license management messages (LC_LICENCE is also supported for backward compatibility)
- LC_PERFM and LC2_PERFM: Log performance messages
- LC_PREEMPT - Log license preemption policy messages
- LC_RESREQ and LC2_RESREQ: Log resource requirement messages
- LC_SYS and LC2_SYS: Log system call messages
- LC_TRACE and LC2_TRACE: Log significant program walk steps
- LC_XDR and LC2_XDR: Log everything transferred by XDR

Default

Not defined.

LS_DEBUG_BLD

Syntax

LS_DEBUG_BLD=*log_class*

Description

Sets the debugging log class for the Platform License Scheduler `bld` daemon.

Used for both project mode and cluster mode.

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 and LC2_AUTH: Log authentication messages
- LC_COMM and LC2_COMM: Log communication messages
- LC_FLEX - Log everything related to FLEX_STAT or FLEX_EXEC Flexera APIs
- LC_LICENSE and LC2_LICENSE : Log license management messages (LC_LICENCE is also supported for backward compatibility)
- LC_MEMORY - Log memory use messages
- LC_PREEMPT - Log license preemption policy messages
- LC_RESREQ and LC2_RESREQ: Log resource requirement messages
- LC_TRACE and LC2_TRACE: Log significant program walk steps
- LC_XDR and LC2_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`.

Used for both project mode and cluster mode.

Default

N

LS_LOG_MASK

Syntax

LS_LOG_MASK=*message_log_level*

Description

Specifies the logging level of error messages for Platform 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.

Used for both project mode and cluster mode.

For example:

```
LS_LOG_MASK=LOG_DEBUG
```

The log levels in order from highest to lowest are:

- LOG_ERR
- LOG_WARNING
- LOG_INFO
- LOG_DEBUG
- LOG_DEBUG1
- LOG_DEBUG2
- LOG_DEBUG3

The most important Platform 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`.

Platform 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_STREAM_FILE_NUMBER

Syntax

LS_MAX_STREAM_FILE_NUMBER=*integer*

Description

Sets the number of saved bld.stream.timestamp log files. When **LS_MAX_STREAM_FILE_NUMBER=2**, for example, the two most recent files are kept along with the current bld.stream file.

Used for both project mode and cluster mode.

Default

0 (old `bld.stream` file is not saved)

LS_MAX_STREAM_SIZE

Syntax

LS_MAX_STREAM_SIZE=*integer*

Description

Defines the maximum size of the bld.stream file in MB. once this size is reached an EVENT_END_OF_STREAM is logged, a new `bld.stream` file is created, and the old `bld.stream` file is renamed `bld.stream.`*`timestamp`*.

Used for both project mode and cluster mode.

Default

1024

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.

Used for both project mode and cluster mode.

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 Platform License Scheduler.

The number taskman sessions must be a positive integer.

The actual maximum number of taskman jobs is affected by the operating system file descriptor limit. Make sure the operating system file descriptor limit and the maximum concurrent connections are large enough to support all `taskman` tasks, License Scheduler (`bl*`) commands, and connections between License Scheduler and LSF.

Used for both project mode and cluster mode.

LS_STREAM_FILE

Syntax

LS_STREAM_FILE=*path*

Used for both project mode and cluster mode.

Description

Defines the full path and filename of the bld event log file, `bld.stream` by default.

Note:

In Platform License Scheduler 8.0 the `bld.events` log file was replaced by the `bld.stream` log file.

Default

`LSF_TOP/work/db/bld.stream`

LS_PREEMPT_PEER

Syntax

LS_PREEMPT_PEER=Y

Description

Enables bottom-up license token preemption in hierarchical project group configuration. Platform License Scheduler attempts to preempt tokens from the closest projects in the hierarchy first. This balances token ownership from the bottom up.

Used for project mode only.

Default

Not defined. Token preemption in hierarchical project groups is top down.

MBD_HEARTBEAT_INTERVAL

Syntax

MBD_HEARTBEAT_INTERVAL=*seconds*

Description

Sets the length of time the cluster license allocation remains unchanged after a cluster has disconnected from `bld`. After MBD_HEARTBEAT_INTERVAL has passed, the allocation is set to zero and licenses are redistributed to other clusters.

Used for cluster mode only.

Default

900 seconds

MBD_REFRESH_INTERVAL

Syntax

MBD_REFRESH_INTERVAL=*seconds*

Description

MBD_REFRESH_INTERVAL: Cluster mode and project mode. This parameter allows the administrator to independently control the minimum interval between load updates from `bld`, and the minimum interval between load updates from LIM. The parameter controls the frequency of scheduling interactive (taskman) jobs. The parameter is read by `mbatchd` on startup. When MBD_REFRESH_INTERVAL is set or changed, you must restart `bld`, and restart `mbatchd` in each cluster.

Used for both project mode and cluster mode.

Default

15 seconds

AUTH

Syntax

MERGE_BY_SERVICE_DOMAIN=Y

Description

Correlates job license checkout with the `lmstat` output across all service domains first before reserving licenses.

This parameter supports the case where application's checkout license number is less than or equal to the job's rusage. If the checked out licenses are greater than the job's rusage, the ENABLE_DYNAMIC_RUSAGE parameter is still required.

Default

Not defined.

PEAK_INUSE_PERIOD

Syntax

PEAK_INUSE_PERIOD=*seconds*

Description

Defines the interval over which a peak INUSE value is determined for dynamic license allocation in cluster mode for all license features over all service domains.

Used for cluster mode only.

When defined in both the `Parameters` section and the `Features` section, the `Features` section definition is used for that license feature.

Default

300 seconds

PORT

Syntax

PORT=*integer*

Description

Defines the TCP listening port used by Platform License Scheduler hosts, including candidate Platform License Scheduler hosts. Specify any non-privileged port number.

Used for both project mode and cluster mode.

PREEMPT_ACTION

Syntax

PREEMPT_ACTION=*action*

Description

Specifies the action used for `taskman` job preemption.

By default, if PREEMPT_ACTION is not configured, `bld` sends a `TSTP` signal to preempt taskman jobs.

You can specify a script using this parameter. For example, **PREEMPT_ACTION = /home/user1/preempt.s** issues preempt.s when preempting a taskman job.

Used for project mode only.

Default

Not defined. A `TSTP` signal is used to preempt `taskman` jobs.

STANDBY_CONNTIMEOUT

Syntax

STANDBY_CONNTIMEOUT=*seconds*

Description

Sets the connection timeout the standby `bld` waits when trying to contact each host before assuming the host is unavailable.

Used for both project mode and cluster mode.

Default

5 seconds

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.

Used for both project mode and cluster mode.

Default

3

Clusters section

Description

Required. Lists the clusters that can use Platform License Scheduler.

When configuring clusters for a WAN, the Clusters section of the master cluster must define its slave clusters.

The Clusters section is the same for both project mode and cluster mode.

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 Platform License Scheduler service domains as groups of physical license server hosts that serve a specific network.

The ServiceDomain section is the same for both project mode and cluster mode.

ServiceDomain section structure

Define a section for each Platform 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
- LM_STAT_TIMEOUT

NAME

Defines the name of the service domain.

Used for both project mode and cluster mode.

LIC_SERVERS

Syntax

LIC_SERVERS=([(*host_name* | *port_number*@*host_name* |**(***port_number*@*host_name* *port_number*@*host_name port_number*@*host_name***))**] ...**)**

Description

Defines the FlexNet license server hosts that make up the Platform License Scheduler service domain. For each FlexNet license server host, specify the number of the port that FlexNet uses, then the at symbol (@), then the name of the host. If FlexNet uses the default port on a host, you can specify the host name without the port number. Put one set of 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.

Used for both project mode and cluster mode.

Examples

- One FlexNet license server host:
  ```
  LIC_SERVERS=((1700@hostA))
  ```
- Multiple FlexNet license server hosts with unique `license.dat` files:
  ```
  LIC_SERVERS=((1700@hostA)(1700@hostB)(1700@hostC))
  ```
- Redundant FlexNet license server hosts sharing the same `license.dat` file:
  ```
  LIC_SERVERS=((1700@hostD 1700@hostE 1700@hostF))
  ```

LIC_COLLECTOR

Syntax

LIC_COLLECTOR=*license_collector_name*

Description

Optional. Defines a name for the license collector daemon (`blcollect`) to use in each service domain. `blcollect` collects license usage information from FlexNet and passes it to the Platform License Scheduler daemon (`bld`). It improves performance by allowing you to distribute license information queries on multiple hosts.

You can only specify one collector per service domain, but you can specify one collector to serve multiple service domains. Each time you run `blcollect`, you must specify the name of the collector for the service domain. You can use any name you want.

Used for both project mode and cluster mode.

Default

Undefined. The Platform 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 Platform License Scheduler makes to collect license usage information from FlexNet 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.

Used for both project mode and cluster mode.

Default

Platform License Scheduler applies the global value defined in the `Parameters` section.

LM_STAT_TIMEOUT

Syntax

LM_STAT_TIMEOUT=*seconds*

Description

Sets the timout value passed to the `lmstat` command. The `Parameters` section setting is overwritten by the `ServiceDomain` setting, which is overwritten by the command line setting (`blcollect -t timeout`).

Used for both project mode and cluster mode.

Default

180 seconds

Feature section

Description

Required. Defines license distribution policies.

Feature section structure

Define a section for each feature managed by Platform License Scheduler.

```
Begin Feature
NAME=vcs
FLEX_NAME=vcs

...

Distribution policy
Parameters

...

End Feature
```

Parameters

- NAME
- CLUSTER_MODE
- FLEX_NAME
- DISTRIBUTION
- ALLOCATION
- GROUP
- GROUP_DISTRIBUTION
- CLUSTER_DISTRIBUTION
- INUSE_FROM_RUSAGE
- ALLOC_BUFFER
- LOCAL_TO
- LS_ACTIVE_PERCENTAGE
- LS_FEATURE_PERCENTAGE
- NON_SHARED_DISTRIBUTION
- PEAK_INUSE_PERIOD
- PREEMPT_ORDER
- PREEMPT_RESERVE
- RETENTION_FACTOR
- SERVICE_DOMAINS
- WORKLOAD_DISTRIBUTION
- ENABLE_DYNAMIC_RUSAGE
- DYNAMIC
- LM_REMOVE_INTERVAL
- ENABLE_MINJOB_PREEMPTION
- ACCINUSE_INCLUDES_OWNERSHIP
- LS_WAIT_TO_PREEMPT

NAME

Required. Defines the token name—the name used by Platform License Scheduler and LSF to identify the license feature.

Used for both project mode and cluster mode.

Normally, license token names should be the same as the FlexNet Licensing feature names, as they represent the same license. However, LSF does not support names that start with a number, or names containing a dash or hyphen character (-), which may be used in the FlexNet Licensing feature name.

CLUSTER_MODE

Syntax

CLUSTER_MODE=Y

Description

Enables cluster mode (instead of project mode) for the license feature. Setting in the `Feature` section overrides the global setting in the `Parameters` section.

Cluster mode emphasizes high utilization of license tokens above other considerations such as ownership. License ownership and sharing can still be configured, but within each cluster instead of across multiple clusters. Preemption of jobs (and licenses) also occurs within each cluster instead of across clusters.

Cluster mode was introduced in Platform License Scheduler 8.0. Before cluster mode was introduced, project mode was the only choice available.

Default

Undefined (N). Platform License Scheduler runs in project mode.

FLEX_NAME

Optional. Defines the feature name—the name used by FlexNet to identify the type of license. You only need to specify this parameter if the Platform License Scheduler token name is not identical to the FlexNet feature name.

Used for both project mode and cluster mode.

FLEX_NAME allows the NAME parameter to be an alias of the FlexNet feature name. For feature names that start with a number or contain a dash (-), you must set both NAME and FLEX_NAME, where FLEX_NAME is the actual FlexNet Licensing feature name, and NAME is an arbitrary license token name you choose.

For example

```
Begin Feature
FLEX_NAME=201-AppZ
NAME=AppZ201
DISTRIBUTION=LanServer1(Lp1 1 Lp2 1)
End Feature
```

DISTRIBUTION

Syntax

DISTRIBUTION=[*service_domain_name*([*project_name number_shares*[/*number_licenses_owned*]] ... [default])] ...

service_domain_name

Specify a Platform License Scheduler service domain (described in the ServiceDomain section) that distributes the licenses.

project_name

Specify a Platform 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. When configured, preemption is enabled and owned licenses are reclaimed using preemption when there is unmet demand.

`default`

A reserved keyword that represents the default project if the job submission does not specify a project (`bsub -Lp`), or the specified project is not configured in the `Projects` section of `lsf.licensescheduler`. Jobs that belong to projects do not get a share of the tokens when the project is not explicitly defined in `DISTRIBUTION`.

Description

Used for project mode only.

One of DISTRIBUTION or GROUP_DISTRIBUTION must be defined when using project mode. GROUP_DISTRIBUTION and DISTRIBUTION are mutually exclusive. If defined in the same feature, the Platform License Scheduler daemon returns an error and ignores this feature.

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.)

Examples

```
DISTRIBUTION=wanserver (Lp1 1 Lp2 1 Lp3 1 Lp4 1)
```

In this example, the service domain named `wanserver` shares licenses equally among four 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 or interactive tasks that licenses are to be allocated to.

project_name

Specify a Platform License Scheduler project (described in the `Projects` section or as `default`) 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 interactive tasks.

Used for project mode only.

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 tasks in Platform License Scheduler projects, use the LSF Task Manager, `taskman`. The Task Manager utility is supported by Platform License Scheduler.

Default

If ENABLE_INTERACTIVE is not set, each cluster receives equal share, and interactive tasks receive no shares.

Examples

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. This is different from a `ProjectGroup` section; groups of projects are not hierarchical.

project_name

Specify a Platform License Scheduler project (described in the `Projects` section) that is allowed to use the licenses. The project must appear in the DISTRIBUTION and 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. The number of licenses owned by the group is the total number of licenses owned by member projects.

Used for project mode only.

This parameter is ignored if GROUP_DISTRIBUTION is also defined.

Example

For example, without the `GROUP` configuration shown, `proj1` owns 4 license tokens and can reclaim them using preemption. After adding the `GROUP` configuration, `proj1` and `proj2` together own 8 license tokens. If `proj2` is idle, `proj1` is able to reclaim all 8 license tokens using preemption.

```
Begin Feature
NAME = AppY
DISTRIBUTION = LanServer1(proj1 1/4 proj2 1/4 proj3 2)
GROUP = GroupA(proj1 proj2)
End Feature
```

GROUP_DISTRIBUTION

Syntax

GROUP_DISTRIBUTION=*top_level_hierarchy_name*

top_level_hierarchy_name

Specify the name of the top level hierarchical group.

Description

Defines the name of the hierarchical group containing the distribution policy attached to this feature, where the hierarchical distribution policy is defined in a `ProjectGroup` section.

One of DISTRIBUTION or GROUP_DISTRIBUTION must be defined when using project mode. GROUP_DISTRIBUTION and DISTRIBUTION are mutually exclusive. If defined in the same feature, the Platform License Scheduler daemon returns an error and ignores this feature.

If GROUP is also defined, it is ignored in favor 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
```

CLUSTER_DISTRIBUTION

Syntax

CLUSTER_DISTRIBUTION=*service_domain*(*cluster shares*/*min*/*max* ...)...

service_domain

Specify a Platform License Scheduler WAN service domain (described in the ServiceDomain section) that distributes licenses to multiple clusters, and the share for each cluster.

Specify a Platform License Scheduler LAN service domain for a single cluster.

cluster

Specify each LSF cluster that accesses licenses from this service domain.

shares

For each cluster specified for a WAN service domain, specify a positive integer representing the number of shares assigned to the cluster. (Not required for a LAN service domain.)

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 assigned by the service domain. The total number of shares is the sum of the shares assigned to each cluster.

min

Optionally, specify a positive integer representing the minimum number of license tokens allocated to the cluster when dynamic allocation is enabled for a WAN service domain (when ALLOC_BUFFER is defined for the feature).

The minimum allocation is allocated exclusively to the cluster, and is similar to the non-shared allocation in project mode.

Cluster shares take precedence over minimum allocations configured. If the minimum allocation exceeds the cluster's share of the total tokens, a cluster's allocation as given by `bld` may be less than the configured minimum allocation.

max

Optionally, specify a positive integer representing the maximum number of license tokens allocated to the cluster when dynamic allocation is enabled for a WAN service domain (when ALLOC_BUFFER is definedfor the feature).

Description

CLUSTER_DISTRIBUTION must be defined when using cluster mode.

Defines the cross-cluster 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 cluster name followed by its share assignment. The share assignment determines what fraction of available licenses is assigned to each cluster, in the event of competition between clusters.

Examples

```
CLUSTER_DISTRIBUTION=wanserver(Cl1 1 Cl2 1 Cl3 1 Cl4 1)
```

```
CLUSTER_DISTRIBUTION = SD(C1 1 C2 1) SD1(C3 1 C4 1) SD2(C1 1) SD3(C2 1)
```

In these examples, `wanserver`, `SD`, and `SD1` are WAN service domains, while `SD2` and `SD3` are LAN service domains serving a single cluster.

INUSE_FROM_RUSAGE

Syntax

INUSE_FROM_RUSAGE=Y|N

Description

When not defined or set to **N**, the INUSE value uses `rusage` from `bsub` job submissions merged with license checkout data reported by `blcollect` (as reported by `blstat`).

When **INUSE_FROM_RUSAGE=Y**, the INUSE value uses the `rusage` from `bsub` job submissions instead of waiting for the blcollect update. This can result in faster reallocation of tokens when using dynamic allocation (when ALLOC_BUFFER is set).

When for individual license features, the `Feature` section setting overrides the global `Parameters` section setting.

Used for cluster mode only.

Default

N

ALLOC_BUFFER

Syntax

ALLOC_BUFFER = *buffer* | *cluster_name buffer* ... **default** *buffer*

Description

Enables dynamic distribution of licenses across clusters in cluster mode.

Cluster names must be the names of clusters defined in the Clusters section of `lsf.licensescheduler`.

Used for cluster mode only.

ALLOC_BUFFER=*buffer* sets one buffer size for all clusters, while **ALLOC_BUFFER=*cluster_name buffer* ...** sets a different buffer size for each cluster.

The buffer size is used during dynamic redistribution of licenses. Increases in allocation are determined by the `PEAK` value, and increased by `DEMAND` for license tokens to a maximum increase of `BUFFER`, the buffer size configured by ALLOC_BUFFER. The licenses allocation can increase in steps as large as the buffer size, but no larger.

Allocation buffers help determine the maximum rate at which tokens can be transferred to a cluster as demand increases in the cluster. The maximum rate of transfer to a cluster is given by the allocation buffer divided by MBD REFRESH INTERVAL. Be careful not to set the allocation buffer too large so that licenses are not wasted because they are be allocated to a cluster that cannot use them.

Decreases in license allocation can be larger than the buffer size, but the allocation must remain at PEAK +BUFFER licenses. The license allocation includes up to the buffer size of extra licenses, in case demand increases.

Increasing the buffer size allows the license allocation to grow faster, but also increases the number of licenses that may go unused at any given time. The buffer value must be tuned for each license feature and cluster to balance these two objectives.

Detailed license distribution information is shown in the `blstat` output.

Use the keyword `default` to apply a buffer size to all remaining clusters. For example:

```
Begin Feature
NAME = f1
CLUSTER_DISTRIBUTION = WanServers(banff 1 berlin 1 boston 1)
ALLOC_BUFFER = banff 10 default 5
End Feature
```

In this example, dynamic distribution is enabled. The cluster `banff` has a buffer size of 10, and all remaining clusters have a buffer size of 5.

To allow a cluster to be able to use licenses only when another cluster does not need them, you can set the cluster distribution for the cluster to 0, and specify an allocation buffer for the number of tokens that the cluster can request.

For example:

```
Begin Feature
CLUSTER_DISTRIBUTION=Wan(CL1 0 CL2 1)
ALLOC_BUFFER=5
End Feature
```

When no jobs are running, the token allocation for CL1 is 5. CL1 can get more than 5 tokens if CL2 does not require them.

Default

Not defined. Static distribution of licenses is used in cluster mode.

LOCAL_TO

Syntax

LOCAL_TO=*cluster_name* | *location_name***(***cluster_name* [*cluster_name* ...]**)**

Description

Used for project mode only.

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, Platform 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 Platform 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.

Platform License Scheduler manages features with different localities as different resources. Use `blinfo` and `blstat` 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 when using project mode.

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 = clusterA
End Feature

Begin Feature
NAME = hspice
GROUP_DISTRIBUTION = group1
SERVICE_DOMAINS = SD2
LOCAL_TO = clusterB
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 taskman jobs, and is not restricted by cluster.

LS_ACTIVE_PERCENTAGE

Syntax

LS_ACTIVE_PERCENTAGE=Y | N

Description

Configures license ownership in percentages instead of absolute numbers and adjusts ownership for inactive projects. Sets **LS_FEATURE_PERCENTAGE=Y** automatically.

Settings **LS_ACTIVE_PERCENTAGE=Y** dynamically adjusts ownership based on project activity, setting ownership to zero for inactive projects and restoring the configured ownership setting when projects become active. If the total ownership for the license feature is greater than 100%, each ownership value is scaled appropriately for a total ownership of 100%.

Used for project mode only.

Default

`N` (Ownership values are not changed based on project activity.)

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 the owned values in `DISTRIBUTION` and the `NON_SHARED_DISTRIBUTION` values only.

When using hierarchical projects, percentage is applied to `OWNERSHIP`, `LIMITS`, and `NON_SHARED` values.

Used for project mode only.

For example:

```
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 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 Platform 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 Platform License Scheduler service domain (described in the ServiceDomain section) that distributes the licenses.

project_name

Specify a Platform License Scheduler project (described in the 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 privately owned, and are not shared with other license projects. They are available only to one project.

Used for project mode only.

Use `blinfo -a` to display NON_SHARED_DISTRIBUTION information.

For projects defined with NON_SHARED_DISTRIBUTION, you must assign the project `OWNERSHIP` an equal or greater number of tokens than the number of non-shared licenses. If the number of owned licenses is less than the number of non-shared licenses, `OWNERSHIP` is set to the number of non-shared licenses.

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.

PEAK_INUSE_PERIOD

Syntax

PEAK_INUSE_PERIOD=*seconds* | *cluster seconds ...*

Description

Defines the interval over which a peak INUSE value is determined for dynamic license allocation in cluster mode for this license features and service domain.

Use keyword `default` to set for all clusters not specified, and the keyword `interactive` (in place of cluster name) to set for `taskman` jobs. For example:

```
PEAK_INUSE_PERIOD = cluster1 1000 cluster2 700 default 300
```

Used for cluster mode only.

When defined in both the `Parameters` section and the `Features` section, the `Features` section definition is used for that license feature.

Default

300 seconds

PREEMPT_ORDER

Syntax

PREEMPT_ORDER=BY_OWNERSHIP

Description

Optional. Sets the preemption order based on configured OWNERSHIP.

Used for project mode only.

Default

Not defined.

PREEMPT_RESERVE

Syntax

PREEMPT_RESERVE=Y

Description

Optional. Enables Platform 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.

Used for project mode only.

Default

Y. Reserved licenses are preemptable.

RETENTION_FACTOR

Syntax

RETENTION_FACTOR=*integer***%**

Description

Ensures that when tokens are reclaimed from an overfed cluster, the overfed cluster still gets to dispatch additional jobs, but at a reduced rate. Specify the retention factor as a percentage of tokens to be retained by the overfed cluster.

For example:

```
Begin Feature
NAME = f1
CLUSTER_MODE = Y
CLUSTER_DISTRIBUTION = LanServer(LAN1 1 LAN2 1)
ALLOC_BUFFER = 20
RETENTION_FACTOR = 25%
End Feature
```

With RETENTION_FACTOR set, as jobs finish in the overfed cluster and free up tokens, at least 25% of the tokens can be reused by the cluster to dispatch additional jobs. Tokens not held by the cluster are redistributed to other clusters. In general, a higher value means that the process of reclaiming tokens from an overfed cluster takes longer, and an overfed cluster gets to dispatch more jobs while tokens are being reclaimed from it.

Used for cluster mode only.

Default

Not defined

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.

Only a single service domain can be specified when using cluster mode.

Used for both project mode and cluster mode.

WORKLOAD_DISTRIBUTION

Syntax

WORKLOAD_DISTRIBUTION=[*service_domain_name***(LSF** *lsf_distribution* **NON_LSF** *non_lsf_distribution***)**] ...

service_domain_name

Specify a Platform 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*.

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.

Used for both project mode and cluster mode. When running in cluster mode, WORKLOAD_DISTRIBUTION can only be specified for WAN service domains; if defined for a LAN feature, it is ignored.

Use `blinfo -a` to display WORKLOAD_DISTRIBUTION configuration.

Example

```
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.

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).

Used for project mode only.

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 Platform License Scheduler to treat the license as a dynamic resource and prevents Platform License Scheduler from scheduling tokens for the feature when they are not available, or reserving license tokens when they should actually be free.

Used for project mode only. Cluster mode does not support rusage duration.

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.

Used for both project mode and cluster mode.

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: Platform 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, Platform License Scheduler preempts one job that uses 10 or more licenses rather than 10 jobs that each use one license.

Used for project mode only

Default

Undefined: Platform License Scheduler does not optimize the job list when selecting jobs to preempt.

ACCINUSE_INCLUDES_OWNERSHIP

Syntax

ACCINUSE_INCLUDES_OWNERSHIP=Y

Description

When not defined, accumulated use is incremented each scheduling cycle by (tokens in use) + (tokens reserved) if this exceeds the number of tokens owned.

When defined, accumulated use is incremented each scheduling cycle by (tokens in use) + (tokens reserved) regardless of the number of tokens owned.

This is useful for projects that have a very high ownership set when considered against the total number of tokens available for LSF workload. Projects can be starved for tokens when the ownership is set too high and this parameter is not set.

Accumulated use is displayed by the `blstat` command under the heading `ACUM_USE`.

Used for project mode only. Cluster mode does not track accumulated use.

Default

N, not enabled.

LS_WAIT_TO_PREEMPT

Syntax

LS_WAIT_TO_PREEMPT=*seconds*

Description

Defines the number of seconds that jobs must wait (time since it was dispatched) before it can be preempted. Applies to LSF and taskman jobs.

Used for project mode only.

When LM_REMOVE_INTERVAL is also defined, the LM_REMOVE_INTERVAL value overrides the LS_WAIT_TO_PREEMPT value.

Default

0. The job can be preempted even if it was just dispatched.

FeatureGroup section

Description

Optional. Collects license features into groups. Put FeatureGroup sections after Feature sections in `lsf.licensescheduler`.

The `FeatureGroup` section is supported in both project mode and cluster mode.

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.

Used for project mode only. When running in cluster mode, any `ProjectGroup` sections are ignored.

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 `blstat -G` to view the hierarchical dynamic license information.

Use `blinfo -G` to view the hierarchical configuration.

ProjectGroup section structure

Define a section for each hierarchical group managed by Platform License Scheduler.

The keywords `GROUP`, `SHARES`, `OWNERSHIP`, `LIMIT`, and `NON_SHARED` are required. The keywords `PRIORITY` and `DESCRIPTION` are optional. Empty brackets are allowed only for `OWNERSHIP`, `LIMIT`, and `PRIORITY`. `SHARES` must be specified.

```
Begin          ProjectGroup
GROUP          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
```

If desired, `ProjectGroup` sections can be completely independent, without any overlapping projects.

```
Begin ProjectGroup
GROUP               SHARES  OWNERSHIP LIMITS  NON_SHARED
(digital_sim (sim sim_reg)) (40 60)  (100 0)  ()  ()
End ProjectGroup

Begin ProjectGroup
GROUP               SHARES  OWNERSHIP LIMITS  NON_SHARED
(analog_sim (app1 multitoken app1_reg)) (50 10 40)  (65 25 0) (- 50 -) ()
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 Parameters 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. When the `DESCRIPTION` column is not empty it should contain one entry for each project group member.

For example:

```
GROUP       SHARES OWNERSHIP   LIMITS NON_SHARED  DESCRIPTION
(R (A B))   (1 1)  ()          ()     (10 10)     ()
(A (p1 p2)) (1 1)  (40 60)     ()     ()          ("p1 desc." "")
(B (p3 p4)) (1 1)  ()          ()     ()          ("p3 desc." "p4 desc.")
```

Use `blinfo -G` to view hierarchical project group descriptions.

Projects section

Description

Required for project mode only. Ignored in cluster mode. Lists the Platform 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.

Used for project mode only.

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 Platform 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 Platform License Scheduler in the cluster with the `bladmin reconfig` command.

Used for both project mode and cluster mode.

The expressions are evaluated by Platform License Scheduler every 10 minutes based on the `bld` start time. When an expression evaluates true, Platform License Scheduler dynamically changes the configuration based on the associated configuration statements, restarting `bld` automatically.

When Platform LSF determines a feature has been added, removed, or changed, `mbatchd` no longer restarts automatically. Instead a message indicates that a change has been detected, prompting the user to restart manually with `badmin mbdrestart`.

This affects automatic time-based configuration in the Feature section of `lsf.licensescheduler`. When `mbatchd` detects a change in the Feature configuration, you must restart `mbatchd` for the change to take effect.

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
```

bladmin

Administrative tool for License Scheduler.

Synopsis

bladmin *subcommand*

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

Description

`bladmin` provides a set of subcommands to control License Scheduler.

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

Subcommand synopsis

ckconfig [**-v**]

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

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

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

blcdebug [**-l** *debug_level*] [**-f** *logfile_name*] [**-o**] *collector_name* ... | all

-h

-V

Usage

ckconfig [-v]

Checks Platform License Scheduler configuration in `LSF_ENVDIR/lsf.licensescheduler` and `lsf.conf`.

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

-v

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

reconfig [*host_name* ... | all]

Reconfigures License Scheduler.

shutdown [*host_name* ... | all]

Shuts down License Scheduler.

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

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

If the `bladmin blddebug` is used without any options, the following default values are used:

- *class_name*=0 (no additional classes are logged)
- *debug_level*=0 (LOG_DEBUG level in parameter LS_LOG_MASK)
- *logfile_name*=current LSF system log file in the LSF system log file directory, in the format *daemon_name*`.log.`*host_name*

-c *class_name* ...

Specifies software classes for which debug messages are to be logged.

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

Valid log classes:

- LC_AUTH: Log authentication messages
- LC_COMM: Log communication messages
- LC_FLEX: Log everything related to FLEX_STAT or FLEX_EXEC Flexera APIs
- LC_LICENCE: Log license management messages
- LC_PREEMPT: Log preemption policy messages
- LC_RESREQ: Log resource requirement messages
- LC_TRACE: Log significant program walk steps
- LC_XDR: Log everything transferred by XDR

Default: 0 (no additional classes are logged)

-l *debug_level*

Specifies level of detail in debug messages. The higher the number, the more detail that is logged. Higher levels include all lower logging levels. For example, LOG_DEBUG3 includes LOG_DEBUG2 LOG_DEBUG1, and LOG_DEBUG levels.

Possible values:

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

`1` LOG_DEBUG1 level for extended logging.

`2` LOG_DEBUG2 level for extended logging.

`3` LOG_DEBUG3 level for extended logging.

Default: 0 (LOG_DEBUG level in parameter LS_LOG_MASK)

-f *logfile_name*

Specifies the name of the file where debugging messages are logged. The file name can be a full path. If a file name without a path is specified, the file is saved in the LSF system log directory.

The name of the file has the following format:

logfile_name.daemon_name.`log.`*host_name*

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

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

Default: current LSF system log file in the LSF system log file directory.

-o

Turns off temporary debug settings and resets them to the daemon starting state. The message log level is reset back to the value of LS_LOG_MASK and classes are reset to the value of LSB_DEBUG_BLD. The log file is also reset back to the default log file.

blcdebug [-l *debug_level*] [-f *logfile_name*] [-o] *collector_name* | all

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

If the `bladmin blcdebug` is used without any options, the following default values are used:

- *debug_level*=0 (LOG_DEBUG level in parameter LS_LOG_MASK)
- *logfile_name*=current LSF system log file in the LSF system log file directory, in the format *daemon_name*`.log.`*host_name*
- *collector_name*=default

-l *debug_level*

Specifies level of detail in debug messages. The higher the number, the more detail that is logged. Higher levels include all lower logging levels. For example, LOG_DEBUG3 includes LOG_DEBUG2 LOG_DEBUG1, and LOG_DEBUG levels.

Possible values:

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

`1` LOG_DEBUG1 level for extended logging.

`2` LOG_DEBUG2 level for extended logging.

`3` LOG_DEBUG3 level for extended logging.

Default: 0 (LOG_DEBUG level in parameter LS_LOG_MASK)

-f *logfile_name*

Specifies the name of the file where debugging messages are logged. The file name can be a full path. If a file name without a path is specified, the file is saved in the LSF system log directory.

The name of the file has the following format:

logfile_name.daemon_name.`log.`*host_name*

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

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

Default: current LSF system log file in the LSF system log file directory.

-o

Turns off temporary debug settings and resets them to the daemon starting state. The message log level is reset back to the value of LS_LOG_MASK and classes are reset to the value of LSB_DEBUG_BLD. The log file is also reset back to the default log file.

If a collector name is not specified, default value is to restore the original log mask and log file directory for the `default` collector.

***collector_name* ... | all**

Specifies the collector names separated by blanks. `all` means all the collectors.

-h

Prints command usage to `stderr` and exits.

-V

Prints release version to `stderr` and exits.

See also

blhosts, `lsf.licensescheduler`, `lsf.conf`

blcollect

license information collection daemon that collects license usage information

Synopsis

blcollect -c *collector_name* **-m** *host_name [...]* **-p** *license_scheduler_port* [**-i** *lmstat_interval* | **-D** *lmstat_path*] [**-t** *timeout*]

blcollect [**-h** | **-V**]

Description

Periodically collects license usage information from Flexera FlexNet. It queries FlexNet for license usage information from the FlexNet `lmstat` command, and passes the information to the License Scheduler daemon (`bld`). The `blcollect` daemon improves performance by allowing you to distribute license information queries on multiple hosts.

By default, license information is collected from FlexNet on one host. Use `blcollect` to distribute the license collection on multiple hosts.

For each service domain configuration in `lsf.licensescheduler`, specify one name for `blcollect` to use. You can only specify one collector per service domain, but you can specify one collector to serve multiple service domains. You can choose any collector name you want, but must use that exact name when you run `blcollect`.

Options

-c

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

-m

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

-p

Required. You must specify the License Scheduler listening port, which is set in `lsf.licensescheduler` and has a default value of 9581.

-i *lmstat_interval*

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

The default interval is 60 seconds.

-D *lmstat_path*

Optional. Location of the FlexNet command `lmstat`.

-t *timeout*

Optional. Timout value passed to the FlexNet command `lmstat`, overwriting the value defined by LM_STAT_TIMEOUT in the `Parameters` or `ServiceDomain` section of the `lsf.licensescheduler` file.

-h

Prints command usage to `stderr` and exits.

-V

Prints release version to `stderr` and exits.

See also

lsf.licensescheduler

blcstat

displays dynamic `blcollect` update information for Platform License Scheduler.

Synopsis

blcstat [**-l**] [*collector_name* ...]

blcstat [**-h** | **-V**]

Description

Displays the time each license collector daemon (`bcollect`) last sent an update to `bld`, along with the current status of each `blcollect`.

Options

-l

Long format. Displays detailed information for each `blcollect` in a multiline format.

collector_name

Displays information only for the specified `blcollect` daemons.

-h

Prints command usage to `stderr` and exits.

-V

Prints the release version to `stderr` and exits.

Output

COLLECTOR_NAME

The name of the license collector daemon as defined by **LIC_COLLECTOR=*license_collector_name*** in the ServiceDomain sections of the `lsf.licensescheduler` file. By default, the name is `_default_`.

STATUS

The current status of the collector.

- `ok`: The collector is working and all license servers can be reached.
- `-ok`: The collector is working, however, not all licenses servers can be reached
- `unavail`: The collector cannot be reached.

LAST_UPD_TIME

The time the last update was received by `bld` for this collector.

-l Output

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

HOST_NAME

The name of the host running this collector.

LICENSE_SERVER

The license server configured in the ServiceDomain section `lsf.licensescheduler` for this collector.

Multiple lines indicate multiple license servers.

Multiple entries in one line separated by '|' indicate configured redundant license servers (sharing the same license file).

License server state is one of:

- `reachable`: The license server is running and providing information to `lmstat`.
- `unreachable`: The license server is not running, or some other problem has blocked the flow of information to `lmstat`.
- `unknown`: `blcollect` is down.

FEATURES

The names of features running on license servers for this collector.

LMSTAT_INTERVAL

The interval between updates from this collector as set by the LM_STAT_INTERVAL parameter in the `Parameters` or `ServiceDomain` section of the `lsf.licensescheduler` file, or by `blcollect` at collector startup.

See also

`blcollect`

blhosts

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

Synopsis

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

Description

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

Options

-h

Prints command usage to `stderr` and exits.

-V

Prints release version to `stderr` and exits.

Output

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

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

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

See also

`blinfo`, `blstat`, `bladmin`

blinfo

displays static Platform License Scheduler configuration information

Synopsis

blinfo -Lp | **-p** | **-D** | **-G** | **-P**

blinfo [**-a** [**-t** *token_name* | **"***token_name ...***"**]] [**-o alpha** | **total**] [**-g "***feature_group ...***"**]

blinfo -A [**-t** *token_name* | **"***token_name ...***"**] [**-o alpha** | **total**] [**-g "***feature_group ...***"**]

blinfo -C [**-t** *token_name* | **"***token_name ...***"**] [**-o alpha** | **total**] [**-g "***feature_group ...***"**]

blinfo [**-t** *token_name* | **"***token_name ...***"**] [**-o alpha** | **total**] [**-g "***feature_group ...***"**]

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

Description

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

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

Options (cluster mode and project mode)

-a

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

You can optionally provide license token names.

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

-C

Shows the cluster locality information for the features.

You can optionally provide license token names.

-D

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

-g *feature_group ...*

When FEATURE_GROUP is configured for a group of license features in `lsf.licensescheduler`, shows only information about the features configured in the FEATURE_LIST of specified feature groups. You can specify more than one feature group at one time.

When you specify feature names with -t, features in the feature list defined by -t and feature groups are both displayed.

Feature groups listed with -g but not defined in `lsf.licensescheduler` are ignored.

-o alpha | total

Sorts license feature information by total tokens.

- alpha: Features are listed in descending alphabetical order.
- total: Features are sorted by the descending order of the sum of licenses that are allocated to LSF workload from all the service domains configured to supply licenses to the feature. Licenses borrowed by non-LSF workload are included in this amount.

-p

Displays values of `lsf.licensescheduler` configuration parameters and `lsf.conf` parameters related to Platform License Scheduler. This is useful for troubleshooting.

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

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

-P

When **LS_FEATURE_PERCENTAGE=Y** or **LS_ACTIVE_PERCENTAGE=Y**, lists the license ownership (if applicable) in percentage.

-h

Prints command usage to `stderr` and exits.

-V

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

Options (project mode only)

-A

When LOCAL_TO is configured for a feature in `lsf.licensescheduler`, shows the feature allocation by cluster locality.

You can optionally provide license token names.

-G

Lists the hierarchical configuration information.

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

-Lp

Lists the active projects managed by Platform License Scheduler.

-Lp only displays projects associated with configured features.

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

Default output

Displays the following fields:

FEATURE

The license name. This becomes the license token name.

When LOCAL_TO is configured for a feature in `lsf.licensescheduler`, `blinfo` shows the cluster locality information for the license features.

SERVICE_DOMAIN

The name of the service domain that provided the license.

TOTAL

The total number of licenses managed by FlexNet. This number comes from FlexNet.

DISTRIBUTION

The distribution of the licenses among license projects in the format [*project_name*, *percentage*[`/`*number_licenses_owned*]]. This determines how many licenses a project is entitled to use when there is competition for licenses. The percentage is calculated from the share specified in the configuration file.

All output (-a)

As default output, plus all other feature-level parameters defined for each feature.

Cluster locality output (-C)

NAME

The license feature token name.

When LOCAL_TO is configured for a feature in `lsf.licensescheduler`, `blinfo` shows the cluster locality information for the license features.

FLEX_NAME

The actual FlexNet feature name—the name used by FlexNet to identify the type of license. May be different from the Platform License Scheduler token name if a different FLEX_NAME is specified in `lsf.licensescheduler`.

CLUSTER_NAME

The name of the cluster the feature is assigned to.

FEATURE

The license feature name. This becomes the license token name.

When LOCAL_TO is configured for a feature in `lsf.licensescheduler`, `blinfo` shows the cluster locality information for the license features.

SERVICE_DOMAIN

The service domain name.

Service Domain Output (-D)

SERVICE_DOMAIN

The service domain name.

LIC_SERVERS

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

(*port_number@host_name*)

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

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

Parameters Output (-p)

Displays values set in the `Parameters` section of `lsf.licensescheduler`.

Displays the following parameter values from `lsf.conf`:

LS_LOG_MASK or LOG_MASK

Specifies the logging level of error messages for Platform 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.

LSF_LIC_SCHED_HOSTS

List of hosts that are candidate Platform License Scheduler hosts. Defined in `lsf.conf`.

LSF_LIC_SCHED_PREEMPT_REQUEUE

Specifies whether to requeue or suspend a job whose license is preempted by Platform License Scheduler. Defined in `lsf.conf`.

LSF_LIC_SCHED_PREEMPT_SLOT_RELEASE

Specifies whether to release the slot of a job that is suspended when its license is preempted by Platform License Scheduler. Defined in `lsf.conf`.

LSF_LIC_SCHED_PREEMPT_STOP

Specifies whether to use job controls to stop a job that is preempted. Defined in `lsf.conf`.

LSF_LICENSE_FILE or LICENSE_FILE

Location of the LSF license file, which includes Platform License Scheduler keys. Defined in `lsf.conf`. If the Platform Scheduler License files are in a different location set by lsf.licensescheduler parameter LICENSE_FILE, this is displayed instead.

Allocation output (-A, project mode)

FEATURE

The license name. This becomes the license token name.

When LOCAL_TO is configured for a feature in `lsf.licensescheduler`, `blinfo` shows the cluster locality information for the license features.

PROJECT

The Platform License Scheduler project name.

ALLOCATION

The percentage of shares assigned to each cluster for a feature and a project.

Hierarchical Output (-G, project mode)

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

GROUP

The project names in the hierarchical grouping and its relationships. Each entry specifies the name of the hierarchical group and its members. The entry is enclosed in parentheses as shown:

(*group* (*member* ...))

SHARES

The shares assigned to the hierarchical group member projects.

OWNERSHIP

The number of licenses that each project owns.

LIMITS

The maximum number of licenses that the hierarchical group member project can use at any one time.

NON_SHARED

The number of licenses that the hierarchical group member projects use exclusively.

PRIORITY

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

DESCRIPTION

The description of the project group.

Project Output (-Lp, project mode)

List of active Platform License Scheduler projects.

-Lp only displays projects associated with configured features.

PROJECT

The project name.

PRIORITY

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

DESCRIPTION

The description of the project.

Examples

`blinfo -a` (project mode) displays both NON_SHARED_DISTRIBUTION and WORKLOAD_DISTRIBUTION information when they are defined:

blinfo -a

```
FEATURE      SERVICE_DOMAIN  TOTAL  DISTRIBUTION
g1           LS              3      [p1, 50.0%] [p2, 50.0% / 2]
                                    NON_SHARED_DISTRIBUTION
                                    [p2, 2]
                                    WORKLOAD_DISTRIBUTION
                                    [LSF 66.7%, NON_LSF 33.3%]
```

Files

Reads `lsf.licensescheduler`

See also

`blstat`, `blusers`, `lsf.licensescheduler`, `lsf.conf`

blkill

terminates an interactive (`taskman`) Platform License Scheduler task

Synopsis

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

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

Description

Terminates a running or waiting interactive task in License Scheduler.

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

By default, `blkill` notifies the user and waits 60 seconds before killing the task.

Options

task_ID

Task ID of the task you want to kill.

-t *seconds*

Specify how many seconds to delay before killing the task. A value of `0` means to kill the task immediately (do not give the user any time to save work).

-h

Prints command usage to `stderr` and exits.

-V

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

blparams

displays information about configurable Platform License Scheduler parameters defined in the files `lsf.licensescheduler` and `lsf.conf`

Synopsis

blparams [**-h** | **-V**]

Description

Displays values set in the `Parameters` section of `lsf.licensescheduler`.

Displays the following parameter values from `lsf.conf`:

LS_LOG_MASK or LOG_MASK

Specifies the logging level of error messages for Platform 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.

LSF_LIC_SCHED_HOSTS

List of hosts that are candidate Platform License Scheduler hosts. Defined in `lsf.conf`.

LSF_LIC_SCHED_PREEMPT_REQUEUE

Specifies whether to requeue or suspend a job whose license is preempted by Platform License Scheduler. Defined in `lsf.conf`.

LSF_LIC_SCHED_PREEMPT_SLOT_RELEASE

Specifies whether to release the slot of a job that is suspended when its license is preempted by Platform License Scheduler. Defined in `lsf.conf`.

LSF_LIC_SCHED_PREEMPT_STOP

Specifies whether to use job controls to stop a job that is preempted. Defined in `lsf.conf`.

LSF_LICENSE_FILE or LICENSE_FILE

Location of the LSF license file, which includes Platform License Scheduler keys. Defined in `lsf.conf`. If the Platform Scheduler License files are in a different location set by lsf.licensescheduler parameter LICENSE_FILE, this is displayed instead.

Options

-h

Prints command usage to `stderr` and exits.

-V

Prints LSF release version to `stderr` and exits.

See also

`lsf.licensescheduler`, `lsf.conf`

blstat

displays dynamic license information

Synopsis

blstat [**-s**] [**-S**] [**-D** *service_domain_name* | **"***service_domain_name ...***"**] [**-P**][**-t** *token_name* | **"***token_name ...***"**] [**-o alpha** | **total** | **avail**] [**-g "***feature_group ...***"**]

blstat [**-a**] [**-c** *token_name*] [**-G**] [**-lslic**] [**-Lp** *ls_project_name* | **"***ls_project_name ...***"**]

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

Description

Displays license usage statistics for Platform License Scheduler.

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

Options (cluster mode and project mode)

-S

Displays information on the license servers associated with license features.

-s

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

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

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

-g *feature_group ...*

When FEATURE_GROUP is configured for a group of license features in `lsf.licensescheduler`, shows information about features configured in the FEATURE_LIST of specified feature groups. You can specify more than one feature group.

When you specify feature names with -t, features in the FEATURE_LIST defined by -t and feature groups are both displayed.

Feature groups listed but not defined in `lsf.licensescheduler` are ignored.

-lslic

Displays how many lsf_license_scheduler licenses have been checked out (TOTAL_CHECKOUT), the current total in use (PEAK_INUSE) and the Platform License Scheduler status (STATUS).

-o alpha | total | avail

Sorts license feature information alphabetically, by total licenses, or by available licenses.

- `alpha`: Features are listed in descending alphabetical order.
- `total`: Features are sorted by the descending order of the sum of licenses that are allocated to LSF workload from all the service domains configured to supply licenses to the feature. Licenses borrowed by non-LSF workload are not included in this amount.
- `avail`: Features are sorted by descending order of licenses available, including free tokens.

-P

Displays percentage values for INUSE and RESERVE. The percentage value represents the number of tokens this project has used and reserved compared to total number of licenses.

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

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

-h

Prints command usage to `stderr` and exits.

-V

Prints the release version to `stderr` and exits.

Options (project mode only)

-a

Displays each project group's accumulated value of licenses. The license token dispatching order is based on the sort order, which is based on the scaled accumulate value of each project. The lower the value, the sooner the license token is dispatched to that project.

-c *token_name*

Displays cross cluster information for tokens, sorted by the value of SCALED_ACUM. The first cluster listed receives tokens first.

Information displayed includes token usage, reserved tokens, free tokens, demand for tokens, accumulated value of tokens, and scaled accumulate value of tokens in each cluster.

- `FREE`: Allocated to the cluster but not used.
- `AVAIL`: If the feature is configured as dynamic, `AVAIL`=reserve + free – preempted. If it is not dynamic, `AVAIL`= in use + reserve + free – preempted.
- `NEED`: Total number of tokens required by pending jobs (`rusage`).

-G

Displays dynamic hierarchical license information.

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

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

Shows project description for specified projects (non-hierarchical). Use spaces to separate multiple names and enclose them in quotation marks.

Output

Information is organized first by license feature, then by service domain. For each combination of license and service domain, Platform License Scheduler displays a line of summary information followed by rows of license project or cluster information.

In each group of statistics, numbers and percentages refer only to licenses of the specified license feature that can be checked out from FlexNet license server hosts in the specified service domain.

Cluster mode summary output

FEATURE

The license name. (This appears only once for each feature.)

SERVICE_DOMAIN

The name of the service domain that provided the license.

TOTAL_TOKENS

The number of licenses from this service domain reserved for Platform License Scheduler jobs.

TOTAL_ALLOC

The number of licenses from this service domain allocated to clusters by Platform License Scheduler.

In most cases TOTAL_ALLOC is equal to TOTAL_USE, however, when there are licenses counted under OTHERS or when tokens are reclaimed, TOTAL_ALLOC may be less than TOTAL_TOKENS.

TOTAL_USE

The number of licenses in use by Platform License Scheduler projects, determined by totalling all INUSE, RESERVE, and OVER values.

OTHERS

The number of licenses checked out by applications outside of Platform License Scheduler.

Cluster output (cluster mode)

For each cluster that is configured to use the license, `blstat` displays the following information.

CLUSTER

The cluster name.

SHARE

The percentage of licenses assigned to the license project by the Platform License Scheduler administrator. This determines how many licenses the project is entitled to

when there is competition for licenses. This information is static, and for a LAN service domain is always 100%.

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

ALLOC

The number of licenses currently allocated to the cluster by the `bld`.

INUSE

The number of licenses checked out by jobs in the cluster.

RESERVE

The number of licenses reserved in the service domain for jobs running in the cluster. This is determined as the difference between the job `rusage` and the number of checked out licenses attributed to the job by License Scheduler.

If the same license is available from both LAN and WAN service domains in cluster mode, Platform License Scheduler expects jobs to try to obtain the license from the LAN first. It is the responsibility of the administrator to ensure that applications bahave in this manner, using the FlexNet environment variable LM_LICENSE_FILE.

OVER

The amount of license checkouts exceeding `rusage`, summed over all jobs.

PEAK

The maximum of INUSE+RESERVE+OVER observed over the past 5 minutes (by default), The observation period is set by PEAK_INUSE_PERIOD in either the Parameters or Feature section.

PEAK is used in scheduling to estimate the cluster's capacity to use licenses in this service domain.

BUFFER

The optional allocation buffer configured in the `Feature` section ALLOC_BUFFER parameter for WAN service domains. When defined, dynamic license token allocation is enabled.

FREE

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

DEMAND

Numeric value indicating the number of tokens required by each cluster.

Project mode summary output

FEATURE

The license name. (This appears only once for each feature.)

SERVICE_DOMAIN

The name of the service domain that provided the license.

TOTAL_INUSE

The number of licenses in use by Platform License Scheduler projects. (Licenses in use have been checked out from the FlexNet license manager.)

TOTAL_RESERVE

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

TOTAL_FREE

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

OTHERS

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

By default, in project mode these licenses are not being managed by Platform License Scheduler policies.

To enforce license distribution policies for these license features, configure **ENABLE_DYNAMIC_RUSAGE=Y** in the `Feature` section for those features in `lsf.licensescheduler`. (Project mode only.)

Workload output (both modes)

LSF_USE

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

LSF_DESERVE

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

LSF_FREE

The total number of free licenses available to Platform License Scheduler projects in the LSF workload.

NON_LSF_USE

The total number of licenses in use by projects in the non-LSF workload.

NON_LSF_DESERVE

The total number of licenses assigned to projects in the non-LSF workload.

NON_LSF_FREE

The total number of free licenses available to projects in the non-LSF workload.

Project output (project mode)

For each project that is configured to use the license, `blstat` displays the following information.

PROJECT

The Platform License Scheduler project name.

SHARE

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

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

LIMITS

The maximum number of licenses that the hierarchical project group member project can use at any one time.

OWN

Numeric value indicating the number of tokens owned by each project.

INUSE

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

RESERVE

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

FREE

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

DEMAND

Numeric value indicating the number of tokens required by each project.

NON_SHARED

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

DESCRIPTION

Description of the project.

ACUM_USE

The number of tokens accumulated by each consumer at runtime. It is the number of licenses assigned to a given consumer for a specific feature.

SCALED_ACUM

The number of tokens accumulated by each consumer at runtime divided by the SHARE value. Platform License Scheduler uses this value to schedule the tokens for each project.

Project group output (project mode)

SHARE_INFO_FOR

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

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

PROJECT/GROUP

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

-lslic output

TOTAL_CHECKOUT

The total number of Platform License Scheduler licenses checked out.

PEAK_INUSE

The number of Platform License Scheduler licenses currently in use.

STATUS

Platform License Scheduler status.

Viewing license feature locality

In project mode, when LOCAL_TO is configured for a feature in `lsf.licensescheduler`, `blstat` shows the cluster locality information for the license features.

Sample output

For example, for a cluster mode feature:

blstat -t f1000

```
FEATURE: f1000           q
 SERVICE_DOMAIN: Lan12
 TOTAL_TOKENS: 1000 TOTAL_ALLOC: 967  TOTAL_USE: 655  OTHERS: 25
  CLUSTER     SHARE   ALLOC INUSE RESERVE OVER  PEAK  BUFFER FREE  DEMAND
  cluster1    66.7 %  647   0     655     0     658   100    0     7452
  interactive 33.3 %  320   0     0       0     0     -      320   0
 SERVICE_DOMAIN: Lan99
 TOTAL_TOKENS: 2000 TOTAL_ALLOC: 2000 TOTAL_USE: 0    OTHERS: 0
  CLUSTER     SHARE   ALLOC INUSE RESERVE OVER  PEAK  BUFFER FREE  DEMAND
  cluster_linux 25.0 %  500   0     0       0     0     100    500   0
  cluster_sparc 25.0 %  500   0     0       0     0     100    500   0
  cluster_aix 25.0 %  500   0     0       0     0     -      500   0
  cluster2 25.0 %  500   0     0       0     0     -      500   0
```

For example, for a project mode feature with a group distribution configuration `blstat` shows the locality of the `hspice` feature configured for various sites:

blstat

```
FEATURE: hspice
 SERVICE_DOMAIN: SD3 SD4
 TOTAL_INUSE: 0    TOTAL_RESERVE: 0    TOTAL_FREE: 22   OTHERS: 0
  PROJECT                 SHARE   OWN  INUSE RESERVE FREE   DEMAND
  Lp1                     50.0 %   3     1    0        0     11
  Lp2                     50.0 %   1     3    0        0     11
FEATURE: hspice@clusterA
 SERVICE_DOMAIN: SD1
 TOTAL_INUSE: 0    TOTAL_RESERVE: 0    TOTAL_FREE: 25   OTHERS: 0
  PROJECT                 SHARE   OWN  INUSE RESERVE FREE   DEMAND
  Lp1                     50.0 %   4     0    0      12       3
  Lp2                     50.0 %   5     0    0      13       1
FEATURE: hspice@siteB
 SERVICE_DOMAIN: SD2
 TOTAL_INUSE: 0    TOTAL_RESERVE: 0    TOTAL_FREE: 65   OTHERS: 0
  PROJECT                 SHARE   OWN  INUSE RESERVE FREE   DEMAND
  Lp1                     50.0 %   4     0    0      32       2
  Lp2                     50.0 %   5     0    0      33       6
```

See also

`blhosts`, `blinfo`

bltasks

displays Platform License Scheduler interactive task information

Synopsis

bltasks [**-l**] [*task_ID*]

bltasks [**-l**] [**-p** | **-r** | **-w**] [**-Lp** “*ls_project_name...*”] [**-m** “*host_name...*”] [**-t** “*terminal_name...*”] [**-u** “*user_name...*”]

bltasks [| **-h** | **-V**]

Description

Displays current information about interactive tasks managed by License Scheduler (submitted using `taskman`).

By default, displays information about all tasks.

Options

task_ID

Only displays information about the specified task.

-l

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

-p

Only displays information about tasks with `PREEMPTED` status.

Cannot be used with -r or -w.

-r

Only displays information about tasks with `RUN` status.

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

-w

Only displays information about tasks with `WAIT` status.

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

-Lp “*ls_project_name*...”

Only displays information about tasks associated with the specified projects.

-m "*host_name*..."

Only displays information about tasks submitted from the specified hosts.

-t “*terminal_name*...”

Only displays information about tasks submitted from the specified terminals.

-u “*user_name*...”

Only displays information about tasks submitted by the specified users.

-h

Prints command usage to `stderr` and exits.

-V

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

Default Output

Displays the short format with the following information:

TID

Task ID that License Scheduler assigned to the task.

USER

The user who submitted the task.

STAT

The current status of the task.

- RUN: Task is running.
- WAIT: Task has not yet started.
- PREEMPT: Task has been preempted and currently has no license token.

HOST

The name of host from which the task was submitted.

PROJECT

The name of the project to which the task belongs.

FEATURES

Name of the License Scheduler token.

CONNECT TIME

The submission time of the task.

Output for -l Option

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

TERMINAL

The terminal the task is using.

PGID

UNIX process group ID.

CPU

The total accumulated CPU time of all processes in a task, in seconds.

MEM

Total resident memory usage of all processes in a task, in KB.

SWAP

Total virtual memory usage of all processes in a task, in KB.

Keyboard idle since

Time at which the task became idle.

RES_REQ

The resource requirement of the task.

Command line

The command the License Scheduler task manager is executing.

blusers

displays license usage information for Platform License Scheduler

Synopsis

blusers [**-J** [**-u** *user_name*]] [**-t** *token_name...*] [**-l**]

blusers -P -j *job_ID* **-u** *user_name* **-m** *host_name* [**-c** *cluster_name*]

blusers [**-h** | **-V**]

Description

By default, displays summarized information about usage of licenses.

Options

-J

Displays detailed license resource request information about each job.

In cluster mode, `blusers -J` does not display additional tokens checked out by the job or features not originally requested by the job.

-u *user_name*

Displays detailed license resource request information about each job belonging to the single user specified.

-t

Displays detailed license resource request information about each job using the token names specified.

-l

Long format. Displays additional license usage information.

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

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

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

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

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

port_number@host_name token_name user_name host_name display

-h

Prints command usage to `stderr` and exits.

-V

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

Default Output

FEATURE

The license name. This becomes the license token name.

SERVICE_DOMAIN

The name of the service domain that provided the license.

USER

The name of the user who submitted the jobs.

HOST

The name of the host where jobs have started.

NLICS

The number of licenses checked out from FlexNet.

NTASKS

The number of running tasks using these licenses.

-J Output

Displays the following summary information for each job:

JOBID

The job ID assigned by LSF.

USER

The name of the user who submitted the job.

HOST

The name of the host where the job has been started.

PROJECT

The name of the license project that the job is associated with.

CLUSTER

The name of the LSF cluster that the job is associated with. Displays "-" for an interactive job.

START_TIME

The job start time.

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

RESOURCE

The name of the license requested by the job.

RUSAGE

The number of licenses requested by the job.

SERVICE_DOMAIN

The name of the service domain that provided the license.

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

INUSE

The number of checked out licenses. Displays '-' when SERVICE_DOMAIN is `UNKNOWN`.

Long Output (-l)

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

OTHERS

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

DISPLAYS

Terminal display associated with the license feature.

Viewing license feature locality

When LOCAL_TO is configured for a feature in `lsf.licensescheduler`, `blusers` shows the cluster locality information for the license features. For example:

blusers

```
FEATURE          SERVICE_DOMAIN  USER     HOST       NLICS    NTASKS
hspice@clusterA  SD1             user1    host1      1        1
hspice@siteB     SD2             user2    host2      1        1
```

Examples

blusers -l

```
FEATURE  SERVICE_DOMAIN  USER   HOST    NLICS   NTASKS OTHERS  DISPLAYS
feat1    LanServer       user1  hostA   1       1      0     (/dev/tty)
```

blusers -J

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

See also

`blhosts`, `blinfo`, `blstat`

fod.conf

The `fod.conf` file contains FOD configuration information. All sections are required.

The command `fodinfo` displays configuration information from this file.

Parameters section

Defines FOD configuration.

Structure

The first and last lines are:

```
Begin Parameters

End Parameters
```

Each subsequent line describes one configuration parameter. All parameters are required.

FOD_ADMIN

Syntax

FOD_ADMIN=*user_name*

Description

The FOD administrator. Specify a valid UNIX user account.

FOD_CLUSTERNAME

Syntax

FOD_CLUSTERNAME=*cluster_name*

Description

The FOD cluster name.

FOD_LICENSE_FILE

Syntax

FOD_LICENSE_FILE=*dir*

Description

Location of the FOD license file.

FOD_LOG_DIR

Syntax

FOD_LOG_DIR=*dir*

Description

Location of the FOD log files.

FOD_PORT

Syntax

FOD_PORT=*integer*

Description

UDP port used by FOD. Specify any port number from 512 to 65536.

FOD_WORK_DIR

Syntax

FOD_LOG_DIR=*dir*

Description

Location of the FOD working files.

Hosts section

Lists the FOD master host candidates.

Structure

The Hosts section begins and ends with the lines Begin Hosts and End Hosts. The second line is column heading, HOSTNAME. Subsequent lines list candidate master hosts, one name per line:

```
Begin Hosts
HOSTNAME
host_name1
host_name2
End Hosts
```

HOSTNAME

Specify a fully qualified host name such as hostX.mycompany.com. The first host listed is the master..

The domain name may be omitted if all the hosts are in the same DNS domain.

Applications section

The application controlled by FOD. Specify only one application.

Structure

```
Begin Applications
NAME                    Path              PARAMS        FATAL_EXIT_VALUE
application_name       dir              parameters     (integer...)
End Applications
```

NAME

The name of the application managed by FOD.

PATH

The path to the location of the application.

PARAMS

The application parameters. Specify a dash (-) to indicate that the application has no parameters.

FATAL_EXIT_VALUE

Optional. Exit values for which FOD does not automatically restart the application. Specify a space-separated list of one or more exit values, within parentheses.

fodadmin

Starts applications under FOD or shuts down FOD.

Synopsis

fodadmin shutdown [*host_name*... | all] **fodadmin** [-h | -V]

You must be License Scheduler administrator to use this command.

This command starts applications under FOD or shuts down FOD.

By default, shuts down FOD on the local host.

Options

shutdown [*host_name*... | all]

Shuts down FOD on the specified hosts. This may shut down applications on the hosts that are managed by FOD. If you shut down the master host, FOD starts up on another host, if possible. Specify all to shut down FOD for the cluster.

-h

Prints command usage to `stderr` and exits.

-V

Prints FOD release version to `stderr` and exits.

fodapps

Displays status of applications managed by FOD.

Synopsis

fodapps [-l | | -h | -V]

Description

Lists all applications managed by FOD and displays information about them.

By default, displays status, PID, and host for each application.

Options

-l

Long format. Also displays path and parameters for each application.

-h

Prints command usage to `stderr` and exits.

-V

Prints FOD release version to `stderr` and exits.

Default output

NAME

Name of the application managed by FOD.

STATUS

The status of the application:

running

The application has started and is running properly.

initial

FOD has not yet attempted to start the application. This state is only seen at startup time.

exit

The application failed to start properly. FOD automatically restarts the application.

PID

The application process ID.

HOST

The name of the FOD master host. All applications managed by FOD run on the FOD master host.

-l output

PATH

The full path of the application.

PARAMETERS

The application parameters.

fodhosts

Displays the status of FOD hosts.

Synopsis

fodhosts [-h | -V]

Description

Lists all FOD hosts and displays status.

The first host listed with `ok` status is the master host..

Options

-h

Prints command usage to `stderr` and exits.

-V

Prints FOD release version to `stderr` and exits.

Output

HOST_NAME

Name of FOD host.

STATUS

Status of FOD host.

ok

FOD is running properly on the host.

unavail

Unavailable. The host may be down or FOD may not be started on the host.

fodid

Displays FOD master host and version information.

Synopsis

fodid [-h | -V]

Description

Displays name of current master host and current version of FOD. Confirms that FOD is started and running.

Options

-h

Prints command usage to `stderr` and exits.

-V

Prints FOD release version to `stderr` and exits.

Index

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

W