EUROPEAN MIDDLEWARE INITIATIVE

LOGGING AND BOOKKEEPING – ADMINISTRATOR'S GUIDE

Document version:	**1.4.11**
EMI Component Version:	**4.x**
Date:	**March 8, 2013**

This work is co-funded by the European Commission as part of the EMI project under Grant Agreement INFSO-RI-261611.

CONTENTS

L&B DOCUMENTATION AND VERSIONS OVERVIEW

The Logging and Bookkeeping service (L&B for short) was initially developed in the EU DataGrid project[1] as a part of the Workload Management System (WMS). The development continued in the EGEE, EGEE-II and EGEE-III projects,[2] where L&B became an independent part of the gLite[3] middleware [1], and then in the EMI Project.[4]

The complete L&B Documentation consists of the following parts:

- **L&B User's Guide** [2]. The User's Guide explains how to use the Logging and Bookkeeping (L&B) service from the user's point of view. The service architecture is described thoroughly. Examples on using L&B's event logging commands to log user tags and change job ACLs are given, as well as L&B query and notification use cases.
- **L&B Administrator's Guide** – this document. The Administrator's Guide explains how to administer the Logging and Bookkeeping (L&B) service. Several deployment scenarios are described together with the installation, configuration, running and troubleshooting steps.
- **L&B Developer's Guide** [3]. The Developer's Guide explains how to use the Logging and Bookkeeping (L&B) service API. Logging (producer), querying (consumer) and notification API as well as the Web Services Interface is described in details together with programing examples.
- **L&B Test Plan** [4]. The Test Plan document explains how to test the Logging and Bookkeeping (L&B) service. Two major categories of tests are described: integration tests (include installation, configuration and basic service ping tests) and system tests (basic functionality tests, performance and stress tests, interoperability tests and security tests).

The following versions of L&B service are covered by these documents:

- *L&B version 4.0*: included in the EMI-3 *Monte Bianco* release
- *L&B version 3.2*: included in the EMI-2 *Matterhorn* release
- *L&B version 3.1*: an update for the EMI-1 *Kebnekaise* release
- *L&B version 3.0*: included in the EMI-1 *Kebnekaise* release
- *L&B version 2.1*: replacement for *L&B version 2.0* in gLite 3.2
- *L&B version 2.0*: included in gLite 3.2 release
- *L&B version 1.x*: included in gLite 3.1 release

L&B packages can be obtained from two distinguished sources:

- **gLite releases**: gLite node-type repositories, offering a specific repository for each node type such as *glite-LB*. Only binary RPM packages are available from that source.

[1] http://eu-datagrid.web.cern.ch/eu-datagrid/
[2] http://www.eu-egee.org/
[3] http://www.glite.org
[4] http://www.eu-emi.eu/

- **emi releases**: EMI repository[5] or EGI's UMD repository,[6] offering all EMI middleware packages from a single repository. There are RPM packages, both source and binary, the latter relying on EPEL for dependencies. There are also DEB packages (starting with EMI-2) and `tar.gz` archives.

Note: Despite offering the same functionality, binary packages obtained from different repositories differ and switching from one to the other for upgrades may not be altogether straightforward.

Updated information about L&B service (including the L&B service roadmap) is available at the L&B homepage: http://egee.cesnet.cz/en/JRA1/LB

[5] http://emisoft.web.cern.ch/emisoft/
[6] http://repository.egi.eu/

1 INTRODUCTION

1.1 SERVICE OVERVIEW

A fairly complete overview of the L&B service is given in L&B User's Guide [2]. This section is a brief excerpt only, providing minimal information necessary for understanding the rest of this document.

The task of L&B is gathering *L&B events* from various grid middleware components (see Figure 1) and delivering the events to L&B servers where users can query for them (Figure 2).

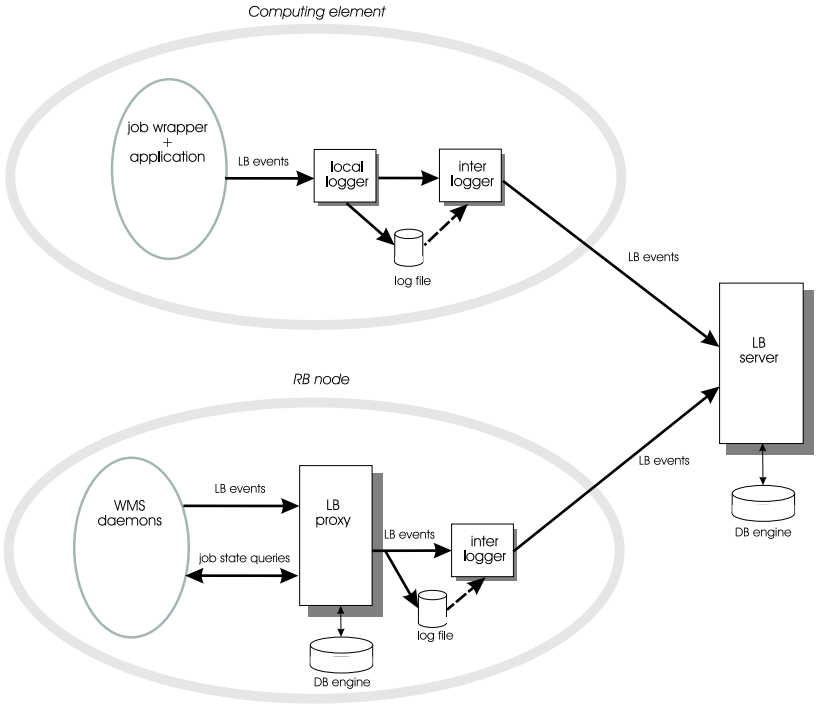

Figure 1: Components involved in gathering and transfering L&B events

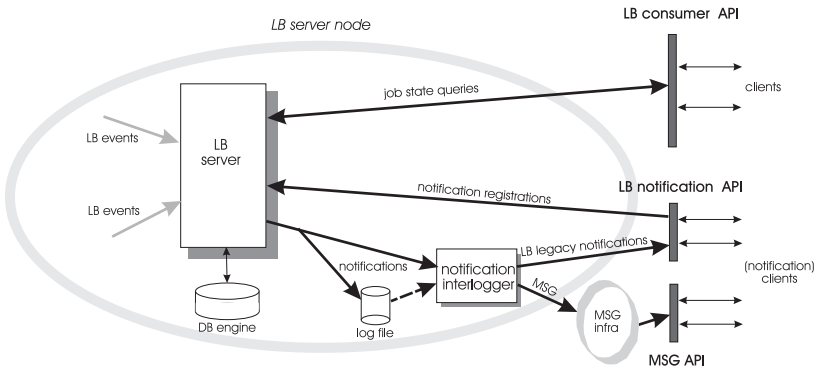

Figure 2: L&B queries and notifications

1.1.1 L&B API AND LIBRARY

Both logging events and querying the service are implemented via calls to a public L&B API. The complete API (both logging and queries) is available in ANSI C binding, most of the querying capabilities also in C++. These APIs are provided as sets of C/C++ header files and shared libraries. The library implements communication protocol with other L&B components (logger and server), including encryption, authentication etc. Since *L&B version 2.0* an experimental Java binding of the logging API is available.

We do not describe the API here in detail; it is documented in L&B Developer's Guide[3], including complete reference and both simple and complex usage examples.

Events can be also logged with a standalone program (using the C API in turn), intended for usage in scripts.

The query interface is also available as a web-service provided by the L&B server (Sect. 1.1.3).

Finally, certain frequent queries (all user's jobs, single job status, ...) are available as HTML pages (by pointing ordinary web browser to the L&B server endpoint), or as simple text queries (since *L&B version 2.0*) intended for scripts. See [2] for details.

1.1.2 LOGGER

The task of the *logger* component is taking over the events from the logging library, storing them reliably, and forwarding to the destination server. The component should be deployed very close to each source of events—on the same machine ideally, or, in the case of computing elements with many worker nodes, on the head node of the cluster[7].

Technically the functionality is realized with two daemons:

- *Local-logger* accepts incoming events, appends them in a plain disk file (one file per Grid job), and forwards to inter-logger. It is kept as simple as possible in order to achieve maximal reliability.
- *Inter-logger* accepts the events from the local-logger, implements the event routing (currently trivial as the destination address is a part of the jobid), and manages delivery queues (one per destination server). It is also responsible for crash recovery—on startup, the queues are populated with undelivered events read from the local-logger files. Finally, the inter-logger purges the files when the events are delivered to their final destination.

1.1.3 SERVER

L&B server is the destination component where the events are delivered, stored and processed to be made available for user queries. The server storage backend is implemented using MySQL database.

Incoming events are parsed, checked for correctness, authorized (only the job owner can store events belonging to a particular job), and stored into the database. In addition, the current state of the job is retrieved from the database, the event is fed into the state machine and the job state updated accordingly.

On the other hand, the server exposes querying interface (Fig. 2). The incoming user queries are transformed into SQL queries on the underlying database engine. The query result is filtered, authorization rules applied, and the result sent back to the user.

[7] In this setup logger also serves as an application proxy, overcoming networking issues like private address space of the worker nodes, blocked outbound connectivity etc.

While using the SQL database, its full query power is not made available to end users. In order to avoid either intentional or unintentional denial-of-service attacks, the queries are restricted in such a way that the transformed SQL query must hit a highly selective index on the database. Otherwise the query is refused, as full database scan would yield unacceptable load. The set of indices is configurable, and it may involve both L&B system attributes (for example job owner, computing element, timestamps of entering particular state, . . .) and user defined ones.

The server also maintains the active notification handles, providing the subscription interface to the user. Whenever an event arrives and the updated job state is computed, it is matched against the active handles[8]. Each match generates a notification message, an extended L&B event containing the job state data, notification handle, and the current user's listener location. The event is passed to the *notification inter-logger* via persistent disk file and directly (see Fig. 2). The daemon delivers events either in a standard way, using the specified listener as destination, or forwards them to a messaging broker for delivery through the messaging infrastructure. When using the standard delivery mechanism, the server generates control messages when the user re-subscribes, changing the listener location. Inter-logger recognizes these messages, and changes the routing of all pending events belonging to this handle accordingly.

1.1.4 PROXY

L&B proxy is the implementation of the concept of local view on job state (see [2]). Since *L&B version 2.0*, L&B proxy is intergrated into L&B server executable. When deployed (on the WMS node in the current gLite middleware) it takes over the role of the local-logger daemon—it accepts the incoming events, stores them in files, and forwards them to the inter-logger.

In addition, the proxy provides the basic principal functionality of L&B server, that is processing events into job state and providing a query interface, with the following differences:

- only events coming from sources on this node are considered; hence the job state may be incomplete,
- proxy is accessed through local UNIX-domain socket instead of network interface,
- no authorization checks are performed—proxy is intended for privileged access only (enforced by the file permissions on the socket),
- aggressive purge strategy is applied—whenever a job reaches a known terminal state (which means that no further events are expected), it is purged from the local database immediately,
- no index checks are applied—we both trust the privileged parties and do not expect the database to grow due to the purge strategy.

1.2 DEPLOYMENT SCENARIOS

1.2.1 STANDALONE L&B SERVER

This is a recommended standard production deployment.

[8] The current implementation enforces specifying an actual jobid in the subscription hence the matching has minimal performance impact.

L&B server is installed on a dedicated machine, where no other grid services (gLite WMS in particular) run. Hence user queries and notification processing are offloaded from the WMS, not affecting its performance directly.

In this setup the full reported performance is achieved, currently up to several hundreds thousands jobs per day, with the goal of one million, see [4].

Further performance can be gained with clustering M WMSs and N L&Bs while configuring all WMSs to distribute jobs to the L&Bs uniformly. In this setup bottlenecks emerging from L&B proxy to L&B server serialized communication are likely to be eliminated. The optimal $M : N$ ratio strongly depends on the rate of user queries and number of evaluated notifications, and it must be determined empirically for each specific installation.

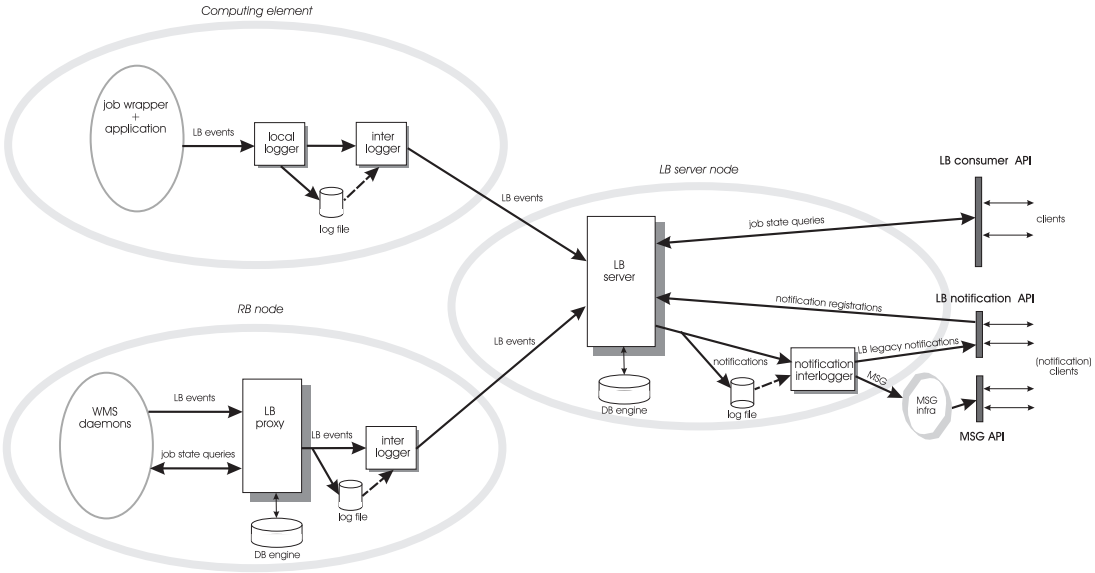

Figure 3: L&B deployment – overall picture

1.2.2 HYBRID L&B SERVER-PROXY

L&B server runs on the WMS node, in combined server-proxy mode, serving both user queries and supporting WMS. Total processing requirements for a single jobs are lower (unlike with separated proxy and server, job state is computed and stored only once).

On the other hand, processing user queries is done on the WMS node, limiting its performance then. This setup is suitable for smaller installations with a single (unclustered) WMS, expected load of upto 30–50 kjobs/day, and not very heavy user-generated load.

The functionality is available since *L&B version 2.0*.

1.2.3 L&B SERVER ON WMS NODE

This setup is obsolete and very inefficient, hence discouraged.

Ancient L&B versions (< 1.2), where L&B proxy was not available yet, used to be frequently installed with L&B server on the WMS machine. With the introduction of L&B proxy it makes little sense anymore but,

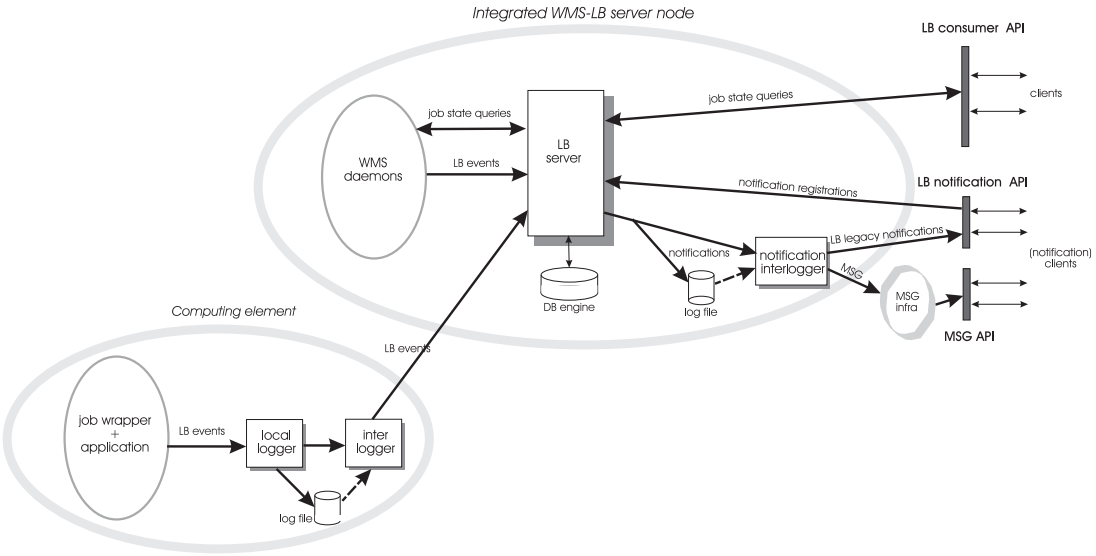

Figure 4: L&B deployment with combined server-proxy

unfortunately, this setup still persists at some sites. Its consequence is doubling both CPU and disk load, yielding observable performance degradation.

Large production sites should consider standalone L&B server (Sect. 1.2.1) instead, while for smaller sites the hybrid setup (Sect. 1.2.2) may be more appropriate.

2 INSTALLATION AND CONFIGURATION

2.1 COMPLETE LIST OF PACKAGES

L&B is currently distributed mainly in RPMs packages and, since *L&B version 3.2* also in *deb* packages. It is available also in binary form packed as `.tar.gz`.

In *L&B version 1.x*, the list of all LB packages was the following:

`glite-lb-common`	common files
`glite-lb-client`	client library and CLI tools
`glite-lb-client-interface`	client library interface (header files)
`glite-lb-harvester`	enhanced L&B notification client (since *L&B version 1.10*)
`glite-lb-logger`	local-logger and inter-logger
`glite-lb-proxy`	proxy (restricted server used by WMS)
`glite-lb-server`	server
`glite-lb-server-bones`	multi-process server building library
`glite-lb-utils`	auxiliary utilities
`glite-lb-ws-interface`	web service interface
`glite-security-gsoap-plugin`	GSS wrapper and GSS plugin for gSoap

Starting with *L&B version 2.0*, the code has been restructured quite a lot, especially the dependencies were lightened, and the new list of packages – applicable also to *L&B version 3.x* is now the following:

`glite-lb-doc`	documentation
`glite-lb-common`	common files
`glite-lb-client`	client library and CLI tools
`glite-lb-client-java`	Java implementation of the client (since *L&B version 2.1*)
`glite-lb-harvester`	enhanced L&B notification client (since *L&B version 2.1*)
`glite-lb-logger`	local-logger and inter-logger
`glite-lb-logger-msg`	plugin for message delivery over ActiveMQ (since *L&B version 3.0*)
`glite-lb-server`	server, including merged proxy functionality
`glite-lb-state-machine`	state machine and LB plugin for Job Provenance
`glite-lb-utils`	auxiliary utilities
`glite-lb-ws-interface`	web service interface
`glite-lb-yaim`	YAIM initialization scripts for L&B (since *L&B version 2.1*)
`glite-lb-nagios-plugins`	Nagios plugin that checks the L&B server (since *L&B version 3.1*)

More detailed description together with the dependencies can be read directly from each package, for example by issuing the command

```
rpm -qiR <package_name>
```

Some of the LB packages depend also on other gLite packages, different due to the restructuring since *L&B version 2.0*. For *L&B version 1.x* they are:

`glite-wms-utils-jobid`	gLite jobId management library
`glite-jp-common`	Job Provenance auxiliary library

And for *L&B version 2.0 and newer*:

`glite-jobid-api-c`	gLite jobId C API library
`glite-lbjp-common-db`	database access layer
`glite-lbjp-common-jp-interface`	interface to the Job Provenance service
`glite-lbjp-common-log`	glite common logging format implementation
`glite-lbjp-common-maildir`	persistent request spool management
`glite-lbjp-common-server-bones`	multi-process server building library
`glite-lbjp-common-trio`	extended printf implementation
`glite-lbjp-common-gss`	GSS wrapper (formerly *glite-security-gss*)
`glite-lbjp-common-gsoap-plugin`	GSS plugin for gSoap (formerly *glite-security-gsoap-plugin*)

where all `glite-lbjp-common-*` packages are common both to L&B and Job Provenance (JP).

2.1.1 DEVELOPMENT PACKAGES

Since *L&B version 3.2* development files have been separated from runtime packages, and they are distributed in standalone `-devel` packages. They are not required for L&B's operation. The following list of development RPMs is given form the sake of completeness:

```
glite-jobid-api-c-devel
glite-jobid-api-cpp-devel
glite-lb-client-devel
glite-lb-common-devel
glite-lb-logger-devel
glite-lb-state-machine-devel
glite-lbjp-common-db-devel
glite-lbjp-common-gsoap-plugin-devel
glite-lbjp-common-gss-devel
glite-lbjp-common-jp-interface-devel
glite-lbjp-common-log-devel
glite-lbjp-common-maildir-devel
glite-lbjp-common-server-bones-devel
glite-lbjp-common-trio-devel
```

2.2 COMMON LOGGING FORMAT

Since *L&B version 2.1* L&B service follows the **gLite common logging recommendations v1.1**:

https://twiki.cern.ch/twiki/pub/EGEE/EGEEgLite/logging.html.

The implementation is done in the `glite-lbjp-common-log` package and it uses `log4c` (http://log4c.sourceforge.net) and its configuration file `log4crc`.

There is one configuration file [`/opt/glite`]`/etc/glite-lb/log4crc` that startup scripts use by setting the `LOG4C_RCPATH` environment variable.

A file `log4crc.example-debugging` may be useful to copy to `$HOME/.log4crc` (or by setting the `LOG4C_RCPATH` environment variable to a directory containing the `log4crc` file) to obtain detailed debugging output. One can debug only specific parts of the LB system, for example by uncommenting `LB.SERVER.DB` category in the `log4crc` file, one gets only the debugging info related to the underlying database subsystem calls.

Currently supported log categories are:

`SECURITY`	Security matters
`ACCESS`	Communication issues
`CONTROL`	L&B server control
`LB`	*unused*
`LB.LOGD`	Messages from the L&B local logger
`LB.INTERLOGD`	Messages from the L&B interlogger
`LB.NOTIFINTERLOGD`	Messages from the notification interlogger
`LB.SERVER`	Processing within the L&B server
`LB.SERVER.DB`	DB calls made by the server
`LB.SERVER.REQUEST`	Processing L&B server requests
`LB.HARVESTER`	Messages from the L&B harvester
`LB.HARVESTER.DB`	DB calls made by the harvester
`LB.AUTHZ`	Authentication matters

The following log priorities are recognized, in a descending order of severity: `fatal`, `error`, `warn`, `info`, `debug`. Note that running with any of the logging categories set do `debug` for a prolonged period of time will result in the logfile growing uncontroledly.

In a typical setup, logging messages generated by the L&B service will be found in `/var/log/messages`.

Note: *syslog* is typically configured to drop log messages with priority `debug`. You may want to enable it, for example by using something like this in `/etc/syslog.conf` and then restarting syslog:

```
*.debug;mail.none;authpriv.none;cron.none                -/var/log/messages
```

2.3 L&B SERVER

2.3.1 HARDWARE REQUIREMENTS

Hardware requirements depend on performance and storage time requirements. Disk space used by LB server consists of database space and working space for backup dumps and temporary files used by exports to Job Provenance and R-GMA tables. Necessary database space can be calculated by multiplying job retention length (job purge timeout), job submission rate, and per-job space requirements (120 KB per job is recommended for current common EGEE usage pattern; jobs can consume more than that with use of very long JDL descriptions, user tags, or very high number of resubmissions). For temporary files, approximately 10 GB is sufficient for LB server setups working normally, more can be needed when backlog forms in data export to any external service. For example, typical setup processing 40 000 jobs per day where all jobs are purged after 10 days needs about 58 gigabytes $(10 \cdot 40000 \cdot 120\text{KB (per job)} + 10)$ not accounting for operating system and system logs.

For smooth handling of 40 000 jobs/day, this or better machine configuration is necessary:

- 1GB RAM
- CPU equivalent of single Xeon/Opteron 1.5GHz
- single 7200 rpm SATA disk.

In order to achieve higher performace, following changes are recommended:

- Faster disks. Disk access speed is crucial for LB server, couple of 15k rpm SCSI or SAS disks (one for MySQL database data file, the second for DB logs, LB server's working directories, and operating system files) or RAID with battery backed write-back cache is preferable.
- More memory. Large RAM improves performance through memory caching, relative speed gain is likely to be rougly proportional to memory/database size ratio. To use 3 GB or more efficiently, 64bit OS and MySQL server versions are recommended.
- Faster or more processors. CPU requirements scale approximately linearly with offered load.

2.3.2 STANDARD INSTALLATION

- Installing from **EMI or UMD repository** (applies to *L&B version 3.0* and higher):

  Install and configure a clean system, the **EPEL** repository, and the **EMI** repository as per instructions given in https://twiki.cern.ch/twiki/bin/view/EMI/GenericInstallationConfigurationEMI1. Then install metapackage `emi-lb`.

- Installing from **gLite's node-type repository** (applies to *L&B version up to 2.1*):

  Install and configure OS and basic services (certificates, CAs, time synchronization, software repositories) according to the https://twiki.cern.ch/twiki/bin/view/LCG/GenericInstallGuide320. Then install metapackage `glite-LB` from an appropriate gLite software repository.

YAIM configuration can be done then:

```
/opt/glite/yaim/bin/yaim -c -s site-info.def -n glite-LB
```

Available parameters specific to LB server are (mandatory parameters are **highlighted**):

- **MYSQL_PASSWORD** – root password of MySQL server (mandatory)
- `GLITE_LB_EXPORT_PURGE_ARGS` – purge timeouts (default: `-cleared 2d -aborted 15d -cancelled 15d -other 60d`)

  According to local retention policy you may want to use different purge timeouts (for example WLCG would need `-cleared 90d -aborted 90d -cancelled 90d -other 90d`).
- `GLITE_LB_EXPORT_ENABLED` – set to `true` for export to JP, installed glite-lb-client and glite-jp-client are needed (default: `false`)
- `GLITE_LB_EXPORT_JPPS` – Job Provenance Primary Storage where to export purged jobs, required if export to JP is enabled
- `GLITE_LB_RTM_ENABLED` – enable settings for Real Time Monitor - indexes and additional access (default: false)
- `GLITE_LB_TYPE` – type of the L&B service: server, proxy, both (default: autodetect, L&B node only: 'server', WMS node only: proxy, L&B and WMS: 'both')
- `GLITE_LB_INDEX_OWNER` – when specified, add (`true`) or drop (`false`) 'owner' index (default: 'owner' index not touched)

- `GLITE_LB_MSG_BROKER` – URL of the MSG broker, 'auto' for looking in BDII, 'false' for disabling MSG notifications (default: auto)
- `GLITE_LB_MSG_NETWORK` – required network type when searching in BDII (default: PROD)
- `LCG_GFAL_INFOSYS` – BDII servers (default: lcg-bdii.cern.ch:2170)
- `EMIR_URL` – enable publishing L&B service info to EMIR server (default: empty), ignored when using local BDII to publish into EMIR (`BDII_EMIR_ENABLE=yes`)

Authorization:

- `GLITE_LB_SUPER_USERS` – additional DNs of super-users (default: empty)[9]
- `GLITE_LB_WMS_DN` – DNs of WMS servers (default: empty)[9]
- `GLITE_LB_RTM_DN` – DNs using to get notifications from L&B server (default: `rtmsrv00.hep.ph.ic.ac.uk` machine certificate)
- `GLITE_LB_AUTHZ_<category>` – more detailed tuning of access grants using certificate DNs, see Section 2.3.13 (default: empty, '`.*`' for logging and job registrations)
- `GLITE_LB_AUTHZ_<category>_FQAN` – more detailed tuning of access grants using VOMS FQANs, see Section 2.3.13 (default: empty)

Additional helper or legacy parameters:

- `GLITE_LB_LOCATION` – L&B prefix (default: `/opt/glite` or `/usr`)
- `GLITE_LB_LOCATION_ETC` – system config directory (default: `/opt/glite/etc` or `/etc`)
- `GLITE_LB_LOCATION_VAR` – gLite local state directory (default: `/opt/glite/var` or `/var/glite`)
- `GLITE_JP_LOCATION` – can be used when JP subsystem location differs from LB (default: empty)
- `GLITE_LB_HARVESTER_ENABLED` – set to `true` for sending notifications, used mainly for legacy export to MSG publish system (default: `false`)
- `GLITE_LB_HARVESTER_MSG_OPTIONS` – additional options for MSG publish (default: `-wlcg`)
- `GLITE_WMS_LCGMON_FILE` – pathname of file where job state export data are written for use by lgcmon/R-GMA (default: `/var/glite/logging/status.log`). *Note: This feature is now obsolete and only available in L&B version 1.x.*

In addition to those, YAIM LB module uses following parameters:

- `INSTALL_ROOT` – installation root, shouldn't be changed (default: '/opt' or '/')
- `GLITE_LOCATION_VAR` – directory for local state files (default: '/opt/glite/var' or '/var/glite')
- `GLITE_USER` – unprivileged user name (default: 'glite')
- **SITE_NAME** – site name (mandatory)
- **SITE_EMAIL** – site email, for cron scripts (mandatory)

Lists are separated by comma.

[9] The use of this parameter is a FAQ. See section 4.2.

2.3.3 CONFIGURATION FILES

Using *yaim*, you do not need to edit L&B's configuration files for your installation to work. On the other hand, if you cannot use *yaim*, you may configure your L&B server through config files detailed bellow. Since *L&B version 3.0* configuration files are located in `/etc/glite-lb/`.

`glite-lb-authz.conf`	Authorization policy, in use since *L&B version 2.1*. Extensively explained in Section 2.3.13 (page 22).
`log4crc`	Logging output configuration, in use since *L&B version 2.0*. Explained in Section 2.2 (page 13).
`msg.conf`	Messaging plugin and features configuration, in use since *L&B version 3.0*. Explained in Section 2.3.10 (page 20).
`site-notif.conf`	Permanent notification registrations to be maintained on the server. Available since *L&B version 3.2*. Explained in section 2.3.11 (page 21).
`html-header.html`	Recognized since *L&B version 3.3*; optional HTML header to be included in all L&B server's HTML output. Explained in section 2.3.19 (page 25).

Besides configuration files observed directly by L&B's services, there is a score of other files installed by L&B and used by other system services. They are:

`/etc/cron.d/glite-lb-purge.cron`	Nightly execution of the L&B purger (see Section 2.3.16).
`/etc/cron.d/locallogger.cron`	Prevent stale handle on `hostcert.pem` by running a `touch` on it quarter-daily.
`/etc/cron.d/glite-lb-notif-keeper.cron`	Refresh registration of L&B notifications set up by site admins py regular calls to `glite-lb-notif-keeper`. Present since *L&B version 3.2*
`/etc/init.d/glite-lb-bkserverd`	Start up the L&B server. Many command line options are hard-coded in the script.[10]
`/etc/init.d/glite-lb-harvester`	Start up the L&B harvester. Some command line options are hard-coded in the script.[10]
`/etc/init.d/glite-lb-locallogger`	Start up the L&B local logger. Some command line options are hard-coded in the script.[10]

2.3.4 READING L&B SERVER CONFIGURATION OVER HTTPS

As of *L&B version 3.0*, it is also possible to use the HTTPs interface to retrieve essential information on L&B Server configuration. For example:

```
firefox https://pelargir.ics.muni.cz:9000/?configuration
```

Among others, the following fields may be discerned from the URL:

`msg_brokers`	A list of messaging brokers that L&B server uses to deliver messages.
`msg_prefixes`	A list of permissible prefixes that must be used in messaging topics.
`server_version`	Version of the L&B server binary.
`server_identity`	The subject of the L&B server's certificate.
`server_indices`	Indices configured in the the L&B server's database. See more on managing indices in Section 3.2.2 (page 29).

[10] When using *yaim*, these scripts are generated and command line options inserted by *yaim*.

Configuration details shown only to L&B super users:

`database_name`	Name of database storing L&B data.
`database_host`	Database server used by L&B.
`authz_policy_file`	Full copy of the current authz configuration file. See more on defining the authorization policy in Section 2.3.13 (page 22).
`admins`	A list of DNs that are allowed super-user access. The list will include superusers given in the authz file as well as those specified from command line through the `--super-user` option.
`dump_start`	Starting time of the latest server dump.See more on backup dumps in Section 3.2.4 (page 31).
`dump_end`	Finishing time of the latest server dump.

2.3.5 MIGRATION TO A DIFFERENT OS VERSION

Migration of a LB server to different machine is possible using following simple procedure (just file copy of the MySQL database). We tested the migration from SL4 32bit (mysql 4.1.22-2) to SL5 64bit (mysql 5.0.45-7).

Steps:

- *Prepare a new machine.* The new machine must get the same hostname as the old machine had. It is a part of job ids stored in the database.
- *Move data.* Just stop the MySQL server and move `/var/lib/mysql` data directory directly to the target machine.
- *(optional) Restore file contexts.* You may need to restore file contexts in case of enabled SELinux.

For example, commands on the target machine:

```
service mysqld stop
cd /var/lib
tar xf /tmp/lb.tar
restorecond -R mysql
service mysqld start
```

2.3.6 MIGRATION OF DATABASE TO SUPPORT TRANSACTIONS

Started from version 1.4.3 of the `glite-lb-server` package, the L&B server introduced optional use of database transactions for L&B database updates in order to improve their performace. This feature is switched on by default when underlying MySQL database uses transactional InnoDB tables. For new installations, YAIM configuration process will create transactional database automatically. For existing LB server database the migration process is not automatically handled.

Note: If you want to add transactions when migrating to *L&B version 2.0 or higher* skip this section and use *L&B version 2.x* migration procedure. The migration of database to support transactions is included ini that procedure.

Steps:

- *Stop the server.* Stop both a L&B server and a MySQL server. Making a fresh backup copy of database is a good idea.
- *Database conversion.* Use provided SQL script:

```
mysql -u lbserver lbserver20 \
        <[/opt/glite]/etc/glite-lb-dbsetup-migrate2transactions.sql
```

- *Start the servers.* MySQL and L&B. Check logs.

2.3.7 MIGRATION TO L&B 2.X

The migration process of existing L&B 1.x database to the L&B 2.x is not handled automatically. The database schema change is required due to support of merged L&B server and proxy services using single database, pointers to purged jobs ("zombies") and other improvements.

Warning: There are two types of L&B database based on the fact that you can have a L&B server or L&B proxy. For more information about L&B proxy please see 2.4

Steps:

- *Stop the server and upgrade to L&B 2.x.* Stop both a L&B server and a MySQL server. Making a fresh backup copy of database is a good idea. Do the upgrade to L&B 2.x, optionally you can move the database to new OS in this step (see 2.3.5).
- *Before migration some database tuning is required.* Especially parameter `innodb_buffer_pool_size` needs to be increased, to support bigger transactions. For details see Section 2.3.18
- *Database type.* Check if you have a L&B server or a L&B proxy. In the following step you must properly set the switch `-s` (server) or `-p` (proxy).
- *Database conversion.* Use provided shell script (with the proper switch from previous step):

```
[/opt/glite]/etc/glite-lb-migrate_db2version20 {-s|-p}
```

- *(Optional) Drop unnecesary index.* This operation is likely to take a lot of time when applied to large database.

```
mysql -u lbserver lbserver20 -e "alter table events drop index host"
```

- *(Optional) Check the L&B indexes.* You may need to add LastUpdateTime for monitoring services. See 3.2.2
- *Start the servers.* MySQL and L&B. Check logs.

This migration procedure is tested in following environment: LB 1.9.x from RPMs SL4 32bit (mysql 4.1.22-2), LB server node, migration to SL5 64bit (mysql 5.0.45-7) LB2.0 RPM.

2.3.8 MIGRATION TO L&B 3.X

There are no specific configuration changes required when migrating from *L&B version 2.x* to *L&B version 3.x*. In case you wish to migrate from *L&B version 1.x*, follow instructions given in section 2.3.7 but upgrade directly to *L&B version 3.0* in the first step.

Note: Upgrading from *L&B version 2.x* provided by gLite L&B node repository to *L&B version 3.0 or higher* provided by the EMI repository has never been tested.

2.3.9 MIGRATION TO L&B 4.X

There was a change in the database schema between *L&B version 3.x* and *L&B version 4.x*. This procedure applies to upgarding from *L&B version 2.x and 3.x*. To upgrade from *L&B version 1.x*, first follow the instructions for upgrading from *L&B version 1.x* to *L&B version 2.x* (Section 2.3.7, page 19) and then continue here.

The change is rather straightforward. When migrating from previous versions, perform a distribution update. Then stop the L&B server, run the DB migration script, and start the L&B server again:

```
/etc/init.d/glite-lb-bkserverd stop
glite-lb-migrate_db2version40
/etc/init.d/glite-lb-bkserverd start
```

2.3.10 CONNECTING TO THE MESSAGING INFRASTRUCTURE

As of *L&B version 3.0*, the L&B server node becomes a producer of messages, which it delivers into the messaging infrastructure.

Correct broker settings are infered from BDII by YAIM on configuration. By default, messaging-related settings are stored in file:

```
[/opt/glite]/etc/glite-lb/msg.conf
```

Overall, `msg.conf` specifies the following information:

- The messaging plugin to be used by the notification interlogger. Plugin settings should be correct *ab initio* and do not require modification by administrators.
- Broker settings (attribute `broker`). They may require an adaptive change in case the currently configured broker disappears and automatic checks fail to switch the settings to another one on time.
- Permissible topic title prefixes (attribute `prefix`). Registrations for delivery to topics not matching the prefix will be rejected. In case no prefix is specified, L&B applies the default EGI prefix: `grid.emi.`

Note: Current broker and prefix settings can be retrieved from the L&B server by any authenticated user reading the server's configuration page – see Section 2.3.4.

2.3.11 MESSAGING: PERSISTENT REGISTRATION FOR NOTIFICATIONS

Starting with *L&B version 3.2* there is a mechanism for site admins to set up and maintain "permanent" registrations for notifications, which should be always kept active on the server. A maintainer script regularly checks existing registrations, extends their validity, and sets up new registrations.

The process is governed by a separate configuration file `/etc/glite-lb/site-notif.conf`. The format of the file is simple, one line per registration, with each line giving a single-word handle and a list of arguments to use for registration.[11]

For instance the following line in `/etc/glite-lb/site-notif.conf`:

```
testnotif       --state running -c -N -a x-msg://grid.emi.lbexample
```

will set up and maintain registration for messages to be generated whenever a job changes state (`-c`) to *running* (`--state running`). Messages will be delivered to topic `grid.emi.lbexample` and user identification (DN) will be replaced with a hash (`-N`). The word `testnotif` is just a plain-text handle.

In another practical example, the following line in `/etc/glite-lb/site-notif.conf`:

```
voDboard        -T -v testvo -a x-msg://grid.emi.testvo.dashboard
```

will set up and maintain registration for messages to be generated whenever a job belonging to VO *testvo* (`-v testvo`) reaches a terminal state (`-T`),[12] and delivered to topic `grid.emi.testvo.dashboard`. Again, the word `voDboard` is simply a plain-text handle.

The maintainer script `glite-lb-notif-keeper` runs regularly and makes sure that registrations do not expire and that the messaging infrastructure keeps receiving them. In case of a listener (messaging broker) being unavailable for a prolonged period of time, the registration is terminated to prevent build-up of undelivered messages. A new registration will be created next time round, and if the listener comes up before then, normal operation will resume.

Applying changes After changing the `site-notif.conf` file, the simplest option is to do nothing and wait for *cron* to invoke the maintainer script `glite-lb-notif-keeper`. You may also run the script manually from the command line. The script handles newly added registrations as well as registrations whose conditions have changed. It does not deal with registrations (lines) removed from the configuration file, though. Those are simply left to expire, typically 12 hours after the latest renewal. In case immediate removal is required, you need to drop the existing registration using `glite-lb-notify drop`.[13]

2.3.12 INDEX CONFIGURATION

Initial YAIM configuration creates L&B indexes typically, the actual configuration depends on required features (for example RTM job monitoring). See Section 3.2.2 for instructions on changing L&B server index configuration afterwards in order to meet specific needs.

[11] Command `glite-lb-notify` is used to make the registrations. See [2] for applicable arguments.

[12] Terminal states are recognized by the server. As of *L&B version 3.2* they are: *cleared*, *aborted*, *canceled* and *purged*.

[13] Note that the server's identity is used to create the registrations, i.e. that the server is the owner. You may need to supply the server's identity to be able to remove a problematic registration.

2.3.13 AUTHORIZATION POLICY

The L&B server applies a quite strict access control policy on the operations provided to the clients to ensure a sufficient level of data protection. By default, the information about a job is only available to the owner of the job. The job owner can specify an ACL assigned to their jobs specifying permissions granted to other users so that they could access the job records, too. More information about the ACL management can be found in the L&B Users' guide.

Apart from using the ACLs, the L&B server administrator can also set a server-level policy granting rights to perform particular operation on L&B server that are considered privileged. For example, a privileged user can access data about jobs owned by other users, bypassing the default L&B access control mechanism. *L&B version 2.1* specifies several categories of rights that can be granted to the users:

- `ADMIN_ACCESS`
- `READ_ALL`
- `PURGE`
- `STATUS_FOR_MONITORING`
- `GET_STATISTICS`
- `REGISTER_JOBS`
- `LOG_WMS_EVENTS`
- `LOG_CE_EVENTS`
- `LOG_GENERAL_EVENTS`
- `GRANT_OWNERSHIP` (since *L&B version 3.0*)
- `READ_ANONYMIZED` (since *L&B version 3.2*)

The first action disables all authorization checks. The next four categories concern with acquring data from the L&B server, while the other ones make it possible to define a web of trusted sources passing events to the L&B server.

`ADMIN_ACCESS` is the most powefull privilege allowing to bypass any authorization checks on the server. It replaces the superuser role, which existed in *L&B version 2.0* and older. Note, that the `--super-users` command-line option still exists and translates internally into granting `ADMIN_ACCESS`.

`READ_ALL` enables to access all job information stored on the server. `PURGE` grants the privilege to ask for purging the L&B database. The L&B server's identity is automatically assigned the `READ_ALL` and `PURGE` so that these operations are available for example to a cron script running on L&B node.

When granted to a user, the `STATUS_FOR_MONITORING` right allows the user to query statuses of all jobs maintaned by the server, however only a small subset of the status fields is returned to back. For example, the caller does not obtain the identity of the job owner. The purpose of this right is to provide a way to collect information necessary for overall monitoring while preserving a basic level of privacy.

`GET_STATISTICS` allows its bearers to query for on-line statitistics generated by the L&B server. See 3.2.6 for more information about the purpose of the function.

`READ_ANONYMIZED` allows reading access to job information on the server, but only in an anonymized form (user identification hashed).

L&B version 1.x allowed anyone possessing a trusted digital certificate to send an arbitrary event to the L&B server. While enabling an easy setup, such an arrangement does not comply with some contemporary requirements, for example job traceability, since the data could be distorted by a malicious user. In order to strengthen the trustworthiness of the data provided, the *L&B version 2.1* server has introduced an authorization mechanism to control the originators of events. The authorization model presumes a set of trusted components that are only allowed to send "important" events, while other types can be logged from any source. The `REGISTER_JOBS` authorization category specifies clients allowed to register jobs with the L&B server. The `LOG_CE_EVENTS` category makes it possible to define a set of trusted CEs that are allowed to log events originating from within sites (in particular `RUNNING`, `REALLYRUNNING`, and `DONE`). Similarly, the `LOG_WMS_EVENTS` category defines a web of trusted WMS nodes. The `LOG_GENERAL_EVENTS` category comprises events that can be sent from any place on the grid, namely `CURDESC`, `USERTAG`, and `CHANGEACL`. It is important to understand that these access control rules provides additional level to the existing authorization routines. In particular, being granted the `LOG_GENERAL_EVENTS` right is not sufficient to e.g. change ACLs on jobs of other people and obtain an access to the job information.

As of *L&B version 3.0* it is possible for the job owner to hand over the ownership of the payload to another user, which eases handling for example of pilot jobs. In order to set the payload owner, the job owner must log a `GrantPayloadOwnership` event, where the identity of the new payload owner is introduced. The event can only originate from loggers described by the `GRANT_OWNERSHIP` category of the policy file. More information and an example of ownership setting is given in the L&B Users' guide.

The L&B policy is specified in a policy configuration file that must be given in the server configuration. Specifying the policy file also triggers the enforcement of access policy rights, especially the ones describing the event sources. If the policy is not enabled, the L&B server accepts events from any logger with a trusted certificate. The format of the policy is a subset of the Simplified policy langauge introduced by the Argus gLite authorization service[14]. Unlike the Argus language, the L&B policy supports only certificate subject names and VOMS fully qualified attribute names (FQANs) to describe the users and do not support 'deny'ing of rights. Also, general regular expressions cannot be used in the argument, only the `".*"` wild card is supported. An example of the policy file follows:

```
resource "LB" {
    action "ADMIN_ACCESS" {
        rule permit {
            subject = "/DC=cz/DC=cesnet-ca/O=CESNET/CN=Daniel Kouril"
        }
        rule permit {
            fqan = "/vo/Role=Manager"
        }
    }
    action "STATUS_FOR_MONITORING" {
        rule permit {
            fqan = "/vo/monitoring"
        }
        rule permit {
            fqan = "/TEST/rtm"
        }
```

[14] https://twiki.cern.ch/twiki/bin/view/EGEE/SimplifiedPolicyLanguage

```
    }
    action "LOG_WMS_EVENTS" {
        rule permit {
            subject = "/DC=cz/DC=cesnet-ca/O=CESNET/CN=wms01.cesnet.cz"
        }
    }
    action "LOG_GENERAL_EVENTS" {
        rule permit {
            subject = ".*"
        }
    }
    action "REGISTER_JOBS" {
        rule permit {
            subject = ".*"
        }
    }
    action "GRANT_OWNERSHIP" {
        rule permit {
            fqan = "/VO/Pilot_job_factory"
        }
    }
}
```

After changing the file, the server has to be restarted.

In order to provide yet additional level of authorization, the LCAS schema[5] is still used in the server. If enabled in configuration, it can be used to specify more general settings using the standard LCAS modules and e.g. blacklist particular users. The standard LCAS documentation should be followed to set up the LCAS layer properly.

2.3.14 EXPORT TO R-GMA

Note: This feature is now obsolete and only available in L&B version 1.x.

L&B server can export information on job state changes to R-GMA infrastructure through `lcgmon` in real time. This export is enabled by YAIM by default and uses `GLITE_WMS_LCGMON_FILE` environmental variable to retrieve name of log file which is to be consumed by `lcgmon` (usually `/var/glite/logging/status.log`). The log file has to be rotated regularly.

2.3.15 DATA BACKUP

Data stored L&B server can be backed up using backups of underlaying database or using `glite-lb-dump` utility. The latter has some advantages, see Section 3.2.4 for details.

2.3.16 PURGING OLD DATA

Initial YAIM configuration creates a cron job which runs once a day and purges old data (jobs in Cleared state after two days, Aborted and Cancelled after 15 days, and other jobs after 60 days of inactivity). It is recommended to run the cron jobs more often (in order to purge less jobs during single run) if event queue

backlogs form in client WMS machines when the purging cron jobs is running. For details on setting job purge timeouts, see Sect. 3.2.5

2.3.17 EXPLOITING PARALLELISM

L&B server uses 10 worker processes (threads) to handle active client accesses (inactive connections are killed when necessary). Each worker process uses separate connection to database server. Number of worker processes can be changed by adding `--slaves` parameter with desired number to servers command line using `GLITE_LB_SERVER_OTHER_OPTIONS` variable.

2.3.18 TUNING DATABASE ENGINE

In order to achieve high performance with LB server underlaying MySQL database server has to be configured reasonably well too. Default values of some MySQL settings are likely to be suboptimal and need tuning, especially for larger machines. These are MySQL configuration variables (to be configured in `[mysqld]` section of `/etc/my.cnf`) that need tuning most often:

- `innodb_buffer_pool_size` – size of database memory pool/cache. It is generally recommended to set it to aroung 75% of RAM size (32bit OS/MySQL versions limit this to approx. 2GB due to address space constraints).
- `innodb_flush_log_at_trx_commit` – frequency of flushing to disk. Recommended values include:
  - 1 (default) – flush at each write transaction commit; relatively slow without battery-backed disk cache but offers highest level of data safety
  - 0 – flush once per second; fast, use if loss of latest updates on MySQL or OS crash (e.g. unhandled power outage) is acceptable (database remains consistent)
- `innodb_log_file_size` – size of database log file. Larger values save some I/O activity, but also make database shutdown and crash recovery slower. Recommended value: 50MB. Clean mysqld shutdown and deletion of log files (`/var/lib/mysql/ib_logfile*` by default) is necessary before change.
- `innodb_data_file_path` – path to main database file. File on disk separate from OS and MySQL log files (`innodb_log_group_home_dir` variable, `/var/lib/mysql/` by default) is recommended.

2.3.19 BRANDING L&B'S HTML OUTPUT

Since *L&B version 3.3*, the server allows site admins to specify header files to be used in all HTML output, branding it with their home organization's visual style or logos. The server looks for the file at `/etc/glite-lb/html-header.html` but option `-H` can be used to specify a different location. The contents of the file are inserted verbatim between the `<head>` and `</head>` tags and typically they will give a `<style>` definition.

The include file's existence is checked on server startup, meaning that it must exist at that moment to be used in the session at all. It is not stored in memory, however, but rather read again from disk every time HTML output is being generated. This simplifies debugging/development of the style file.

It is important to note that L&B does in no way parse the file. It simply includes it "as is" and making the resulting output valid is the sole responsibility of the site administrator.

2.4 L&B PROXY

All necessary configuration of L&B proxy is done by YAIM, and described with gLite WMS installation elsewhere. Previous L&B server section applies to merged server+proxy setups (since *L&B version 2.0*).

A special care must be taken when an existing L&B proxy database is migrated to *L&B version 2.x*. In general, this is not a typical scenario – *L&B version 2.x* server in proxy mode on WMS node is introduced with a major WMS upgrade, and it is expeced to be installed from scratch rather than migrated, preserving L&B proxy data.

If the migration is really needed, `glite-lb-migrate_db2version20` script should be run with `-p` (Sect 2.3.7). However, the L&B database name remains `lbproxy` while the *L&B version 2.x* binaries expect unified `lbserver20` by default. Because renaming a MySQL database is a non-trivial, error prone task, the recommended workaround is to add the following variable setting

```
GLITE_LB_SERVER_OTHER_OPTIONS="--mysql lbserver/@localhost:lbproxy"
```

into the gLite startup environment (`[/opt/glite]/etc/profile.d`) instead. This setting makes L&B server use the `lbproxy` database instead of the default.

2.5 L&B LOGGER

All necessary configuration of normal L&B logger is done by YAIM.

2.6 L&B HARVESTER

L&B Harvester gathers information about jobs from L&B servers using efficient L&B notification mechanism. It manages notifications and keeps them in a persistent storage (file or database table) to reuse later on next launch. It takes care of refreshing notifications and queries L&B servers back when some notification expires.

It is not intended for normal usage. You will need Harvester for sending notifications to MSG publish infrastructure, but only for older L&B server releases (gLite 3.1.x, support since gLite 3.1.19). Example of YAIM configuration:

```
GLITE_LB_HARVESTER_ENABLED=true
GLITE_LB_HARVESTER_MSG_OPTIONS="--wlcg-topic=org.wlcg.usage.jobStatus
 --wlcg-config=/etc/msg-publish/msg-publish.conf
 --wlcg-binary=/usr/bin/msg-publish"
```

Since gLite 3.2.1, L&B Harvester is not needed for MSG publish. Notifications are reliably delivered by interlogger instead. Delivery can be switched on in this case by `GLITE_LB_MSG_ENABLED` and `GLITE_LB_MSG_BROKER` YAIM options.

2.7 SMOKE TESTS

Thorough tests of L&B, including performance measurement, are covered in the L&B Test Plan document [4]. This section describes only elementary tests that verify basic functionality of the services.

The following test description assumes the L&B services installed and started as described above.

2.7.1 JOB REGISTRATION

Register a new job with the L&B server and check that its status is reported correctly.

Prerequisities: Installed glite-lb-client package, valid user's X509 credentials, known destination (address:port) of running L&B server. Can be invoked from any machine.[15]

How to run:

```
glite-lb-job_reg -m server_name:port -s Application
```

A new jobid is generated and printed. Run

```
glite-lb-job_status jobid
```

Expected result: The command should report "Submitted" job status.

2.7.2 LOGGING EVENTS VIA LB-LOGGER

Send several L&B events, simulating normal job life cycle. Checks delivery of the events via L&B logger.

Prerequisities: Installed package `glite-lb-client`, valid X509 credentials, known destination (address:port) of running L&B server. Must be run on a machine where `glite-lb-logger` package is set up and running.[15]

How to run:

```
glite-lb-running.sh -m server_name:port
```

The command prints a new jobid, followed by diagnostic messages as the events are logged. Check the status of the new job with

```
glite-lb-job_status jobid
```

Expected result: Due to asynchronous event delivery various job states can be reported for limited time (several seconds). Finally the "Running" status should be reached.

2.7.3 LOGGING EVENTS VIA LB-PROXY

Send events via L&B proxy. Checks the proxy functionality.

Prerequisities: Running L&B proxy, in standalone package for *L&B version 1.x*, or L&B server running with `-P` (proxy only) or `-B` (both server and proxy) options.[15]

How to run: Similar to Sect. 2.7.2:

```
glite-lb-running.sh -x -m  server_name:port
```

And check with:

```
glite-lb-job_status -x /tmp/lb_proxy_server.sock jobid
```

[15] Example scripts or binaries used here can be found either in `/opt/glite/examples` (if installed from a LB node repository) or `/usr/lib64/glite-lb/examples` (if installed from the EMI repository). Please note that these examples are primarily intended for developers, and their use (e.g. command line options) is not covered by documentation execpt for these few specific use cases.

2.7.4 NOTIFICATION DELIVERY

Register for receiving notifications, and log events which trigger notification delivery. This checks the whole notification mechanism.

Prerequisities: Package `glite-lb-client` installed, valid user credentials, environment variables (namely `GLITE_WMS_NOTIF_SERVER`) set to point to the L&B server.

How to run: First register for notifications. Option `-O`[16] allows you to register for notifications on all your jobs:

```
glite-lb-notify new -O
```

This prints out a notification ID (*NotifID*) and validity information. Continue by listening for notifications, using that ID:

`glite-lb-notify receive` *NotifID*

Then, using a different console, register at least one job and generate events. Make sure you use the same Identity used to register for notifications in the first place.

`glite-lb-running.sh -m` *server_name:port*

Expected result: The listener (`glite-lb-notify`) should print out notifications (JobID, state and owner), showing progressive changes of job state as events arrive to the server. There will be multiple notifications showing the same state. That is normal since not all events result in a job state change. If you want to see notifications only for job state changes, use option `-c` on registration.

For more test scenarios, see notification-related sections in [2, 4]. Especially [2] has a comprehensive set of examples using the `glite-lb-notify` command.

[16] Supported since *L&B version 2.0*. For older versions, you need to use option `-j` and give at least one JobID of a job that already exists!

3 MAINTENANCE

3.1 CHANGING DEFAULT SETTINGS

All configurable settings of the L&B daemons (network and local sockets, file paths, modified behaviour etc.) can be set with specific command line options. In the following only relevant options are discussed whenever appropriate. See specific manual pages for complete reference.

3.2 L&B SERVER AND PROXY

This section deals with several typical but more peculiar tasks that need more verbose description. It is complemented with the full commands reference that is provided as standard manual pages installed with the L&B packages.

3.2.1 STANDARD AND DEBUG LOGS

In normal operation L&B server sends error messages to syslog. Informational messages are generally avoided in order to prevent syslog congestion.

Since *L&B version 2.1*, the service implements a common logging format[17] (see section 2.2, page 13).

For *L&B version 2.0 and lower*, the following instructions apply:

When tracing problems, GLITE_LB_SERVER_DEBUG environment variable can be set to non-empty value when starting the service. Then verbose log $GLITE_LOCATION_VAR/lb.log (as well as $GLITE_LOCATION_VAR/notif-il.log eventually when notifications are enabled). Beware that these can grow huge easily.

***L&B version 1.x* only:** not available for L&B proxy, `-d` and output redirection have to be added manually if necessary.

3.2.2 CHANGING INDEX CONFIGURATION

L&B server only (L&B proxy database is neither so huge nor accessed directly by users).

Inefficient queries, yielding full scan of L&B database tables (up to millions of tuples) would degrade server performance. Therefore L&B does not allow arbitrary queries in general (server option `--no-index` can change this behaviour). On the contrary, a query has to hit a *job index*, build on one or more job attributes. It is left up to the specific L&B server administrator to decide which job attributes are selective enough to be indexed and allow queries (for example for many-users communities job owner can be a sufficient criterion; for others, where only a few users submit thousands of jobs, it is not).

Technically, job indices are implemented via dedicated columns in a database table. These columns and their indices are scanned by the L&B server on startup, therefore there is no specific configuration file. Changing the index configuration is rather heavyweight operation (depending on the number of jobs in the database), it performs updates of all tuples in general, and it should be done when the server is not running.

Indices are manipulated with a standalone utility `glite-lb-bkindex` (see its man page for complete usage reference). A general sequence of changing the indices is:

[17] http://en.wikipedia.org/wiki/Common_Log_Format

IndexType	*IndexName*	description
system	owner	job owner
	destination	where the job is heading to (computing element name)
	location	where is the job being processed
	network_server	endpoint of WMS
	stateEnterTime	time when current status was entered
	lastUpdateTime	last time when the job status was updated
time	*state name*	when the job entered given state (Waiting, Ready, . . .)
user	*arbitrary*	arbitrary user tag

Table 1: Available index column types and names

1. stop the running server
2. retrieve current index configuration

   ```
   glite-lb-bkindex -d >index_file
   ```

3. edit `index_file` appropriately
4. re-index the database (it may take long time)

   ```
   glite-lb-bkindex -r -v index_file
   ```

   `-r` stands for "really do it", `-v` is "be verbose"
5. start the server again

The index description file follows the classad format, having the following grammar:

IndexFile ::= [JobIndices = { *IndexList* }]
IndexList ::= *IndexDef* | *IndexDef*, *IndexList*
IndexDef ::= *IndexColumn* | *ComplexIndex*
IndexColumn ::= [type = "*IndexType*"; name = "*IndexName*"]
ComplexIndex ::= { *ColumnList* }
ColumnList ::= *IndexColumn* | *IndexColumn*, *ColumnList*

where eligible *IndexType*, *IndexName* combinations are given in Tab. 1. A template index configuration, containing indices on the most frequently used attributes, can be found in `[/opt/glite]/etc/glite-lb-index.conf.template`.[18]

3.2.3 MULTIPLE INSTANCES

Specific conditions (for example debugging, different authorization setup, . . .) may require running multiple L&B server instances on the same machine. Such setup is available, however, there is no specific support in automated configuration, the additional non-default server instances must be run manually.

The other server instance must use different ports (changed with `-p` and `-w` options), as well as use different pid file (`-i` option).

[18] The `/opt/glite` prefix applies for L&B installed from gLite's L&B node repository.

The servers may or may not share the database (non-default is specified with `-m`)[19].

Though it may have little sense to run multiple L&B proxy instances, it is possible too. Non-default listening socket have to be specified via `-p` option then.

3.2.4 BACKUP DUMPS

L&B server only, not supported by proxy.

(This functionality should not be confused with per-job dumps, Sect. 2.3.16 and 3.2.5)

Besides setting up L&B server database on a reliable storage or backing it up directly (Sect. 2.3.15) L&B server supports backing up only incremental changes in the data. Advantages of this approach are lower volume of data to be backed up, and possibility to load them to another instance (for example for heavyweight queries which should not disturb normal operation), disadvantage is a more complex and more fragile setup.

Using an external utility `glite-lb-dump` (typical invocation is with a single option `-m` *server_name:port*, see man page for details) the server is triggered to dump events, which arrived in a specified time interval, into a text file. (Default interval is from last dump till the current time.)

`glite-lb-dump` is a standalone client program, however, the events are stored at server side (that is not transferred to the client, due to performance reasons), in a uniquely named text file prefixed with the value of `-D` server option. This kind of dump contains events according to their arrival time, regardless of jobs they belong to.

It is sufficient to run the dump regularly (from a cron job), with a frequency matching an acceptable risk of loosing data (several hours typically), and back up the resulting dump files.

In the event of server crash, its database should be recreated empty, and the server started up. Then the dump files can be loaded back with complementary `glite-lb-load` utility.

Server privileges granting `ADMIN_ACCESS` (see section 2.3.13) are required to run `glite-lb-dump` and `glite-lb-load`. Dumping the events does not interfere with normal server operation.

This backup strategy can interfere with too aggressive setting of old data purging (Sect. 3.2.5), If the purging grace period is shorter than the dump interval, events may get purged before they are captured by the backup dump. However, this interference is unlikely (reasonable purge grace period is several times longer than dump period), and it is not fatal in general (data were purged on purpose either).

3.2.5 PURGING AND PROCESSING OLD DATA

Primary purpose of the LB purge operation is removal of aged data from LB database. This is necessary in production in order to prevent ever-increasing database and sustain reasonable performance of the server. Therefore the purge should be invoked periodically.

The purge operation has additional important "side effect" – dumping the purged data into a plain text file. These dumps can be archived "as is" or uploaded to Job Provenance.

Purge setup The purge operation itself is performed by a running L&B server (there is no need to shut it down, then). However, it is triggered with `glite-lb-purge` client command (complete usage reference

[19] Even when sharing the database, the servers are still partially isolated from one another, for example a job https://my.machine:9000/xyz cannot be queried as https://my.machine:8000/xyz. However, due to implementation internals, the second job cannot be registered.

is given in its man page). A typical invocation specifies L&B server to purge (`-m` option), and purge timeouts (grace periods) for several job states – options `-a` (aborted), `-n` (canceled), `-c` (cleared), and `-o` (other). L&B versions 2.0 supports also `-e` (done) option. A job falling in one of the four categories is purged when it has not been touched (that is an event arrived) for time longer than the specified category timeout. Suggested values are several days for aborted and canceled jobs, and one day for cleared jobs, however, the values may strongly vary with L&B server policy.

Optionally, `-s` purge command option instructs the server to dump the purged data into a file at the server side. It's location (prefix) is given by `-S` server option, the purge command reports a specific file name on its output.

It is recommended (and the default YAIM setup does so, via the `glite-lb-export.sh` wrapper) to run the purge command periodically from cron.

Server privileges granting `ADMIN_ACCESS` (see section 2.3.13) are required to run `glite-lb-purge`.

If the server database has already grown huge, the purge operation can take rather long and hit the L&B server operation timeout. At client side, that is the glite-lb-purge command, it can be increased by setting GLITE_WMS_QUERY_TIMEOUT environment variable. Sometimes hardcoded server-side timeout can be still reached. In either case the server fails to return a correct response to the client but the purge is done anyway.

L&B proxy purges jobs automatically when they reach a state ensuring that WMS will neither query nor log events to them anymore. Therefore routine purging is not required theoretically. However, frozen jobs which never reach such a state may occur in an unstable environment, and they may cumulate in L&B proxy database for ever. Therefore occasional purging is recommended too. *L&B version 2.x* supports `-x` option of `glite-lb-purge`, allowing to purge L&B proxy database too. With *L&B version 1.x* the emergency purge procedure described bellow is the only option.

Emergency purge When regular purge was not invoked for some time, it may happen that the database grows huge and the regular (on-line) purge fails. In order to work around such situation we provide an off-line emergency purge script `glite-lb-bkpurge-offline.sh`

The script accepts the same `-acno` options, and adds `-d` for "done" jobs. Via `-p` also L&B proxy database can be purged (all L&B versions).

On startup, a warning message is printed and interactive confirmation requested. Re-check that L&B server (proxy) is not running, and carry on only when you know what you are doing.

Post-mortem statistics Once a job is purged from the database, all important data about the job can be processed offline from the corresponding dump file. The idea of post-mortem statistics is the following:

- LB server produces dump files (during each purge on regular basis), see LB server startup script; option `-D / --dump-prefix` of `glite-lb-bkserverd`,
- these dumps are exported for the purposes of JP also on regular basis, see LB/JP deplyment module; option `-s/ --store` of `glite-lb-lb_dump_exporter`,
- it depends on the LB server policy if dumps in this directory are used for the statistics purposes or all files are hardlinked for example to a different directory
- general idea is such that data are available for statistics server that downloads and removes dumps after download! Dump files are then processed on the statistics server.

What needs to be done on the LB server:

- `glite-lb-bkserverd` and `glite-lb-lb_dump_exporter` running
- `gridftp` running (allowing statistics server to download and remove files from a given directory

What needs to be done on the statistics server:

- `glite-lb-utils` package installed
- download and remove files from the LB server see `glite-lb-statistics-gsi.sh` (shell script in the examples directory)
- process dump files using the `glite-lb-statistics` tool see `glite-lb-statistics.sh` (shell script in the examples directory)

all scripts are supposed to be run from a crontab.

Export to Job Provenance An important, though currently optional, processing of L&B dumps is their upload to the Job Provenance service for permanent preservation of the data.

When enabled (via configuration environment variables, see bellow), the export is done in two steps:

- `glite-lb-export.sh` wrapper script, after calling `glite-lb-purge`, breaks up the resulting dump file on a per-job basis. The individual job dump files are stored in a dedicated spool directory.
- `glite-jp-importer` daemon (installed optionally in glite-jp-client.rpm) checks the spool directory periodically, and tries to upload the files into JP.

Details, including the configuration variables, are covered at the following wiki page:

http://egee.cesnet.cz/mediawiki/index.php/LB_purge_and_export_to_JP.

Persistent Information on Purged Jobs ("Zombies") Since L&B version 2.0, the JobID of a purged job is not fully discarded but stored in a separate table. A query for the status of such job returns *Identifier removed*.[20] There are two purposes to this arrangement:

1. Confirming that the job existed and that its details have been exported to Job Provenance (if deployed and configured),
2. Preventing reuse of the JobID in case of flag `EDG_WLL_LOGLFLAG_EXCL` set on registration.

3.2.6 ON-LINE MONITORING AND STATISTICS

CE reputability rank Rather frequent problem in the grid production are "black hole" sites (Computing Elements). Such a site declares itself to have an empty queue, therefore schedulers usually prefer sending jobs there. The site accepts the job but it fails there immediately. In this way large number of jobs can be swallowed, affecting the overall success rate (namely for non-resubmittable jobs).

[20] As of L&B 3.0. In 2.0 and 2.1 releases an empty job status structure was returned with job state set to *purged*.

L&B data as a whole contain enough information to detect such sites. However, due to the primary per-job structure certain reorganization is required.

A job is always assigned to a *group* according to the CE where it is executed (cf. "destination" job state attribute). Similarly to RRDtool[21] for each recently active group (CE), and for each job state (Ready, Scheduled, Running, Done/OK, Done/Failed), a fixed sized series of counters is maintained. At time t, the counters cover intervals $[t-T,t]$, $[t-2T,t-T]$, ... where T a fixed interval size. Whenever a job state changes, the series matching the group and new state is shifted eventually (dropping its expired tail), and the current counter is incremented. In addition, multiple series for different T values (that is covering different total times) are available.

The data are available via statistics calls of the client API, see `statistics.h` for details (coming with glite-lb-client in *L&B version 2.x*, glite-lb-client-interface in *L&B version 1.x*). The call specifies the group and job state of interest, as well as queried time interval. The interval is fitted to the running counter series as accurately as possible, and the average number of jobs per second which entered the specific state for the given group is computed. The resolution (T) of the used counters is also returned.

In gLite 3.1 WMS the calls can be accessed from inside of the matchmaking process via `successFraction(CEId)` JDL function. The function computes the ratio of successful vs. all jobs for a given CE, and it can be directly used to penalize detected black hole CEs in the ranking JDL expression.

The functionality is enabled with `--count-statistics` L&B server option (disabled by default).

The gathered information is currently not persistent, it is lost when the server is stopped. Despite the statistics call API is defined in a general way, the implementation is restricted to a hardcoded configuration of a single grouping criterion (the destination), and a fixed set of counter series (60 counters of $T = 10s$, 30 of 1 minute, and 12 of 15 minutes). The functionality has not been very thoroughly tested yet.

glite-lb-mon is a program for monitoring the number of jobs on the LB server and their several statistics. It is part of the `glite-lb-utils` package, so the monitoring can be done from remote machine where this package is installed and the enironment variable `GLITE_WMS_QUERY_SERVER` properly set. Values like minimum, average and maximum time spent in the system are calulated for jobs that entered the final state (Aborted, Cleared, Cancelled) in specific time (default last hour). Also number of jobs that entered the system during this time is calculated.

A special bkindex configuration is needed. The following time indices must be defined:

```
[ type = "time"; name = "submitted" ],
[ type = "time"; name = "cleared" ],
[ type = "time"; name = "aborted" ],
[ type = "time"; name = "cancelled" ],
```

For more details se man page glite-lb-mon(1).

glite-lb-mon-db is a low-level program for monitoring the the number of jobs in the LB system. Using the LB internals, it connects directly to the underlying MySQL database and reads the number of jobs in each state. The tool is distributed itogether with the server in the `glite-lb-server` package. It can be used to read data also from the database of LB Proxy. For more details se man page glite-lb-mon-db(1).

[21] http://oss.oetiker.ch/rrdtool/

Subjob states in a collection can be calculated on demand on the server and returned as a histogram using standard job status query. There are two ways how to obtain the histogram:

- fast histograms, the last known states are returned, see e.g.

```
glite-lb-job_status -fasthist <collectionJobId>
```

- full histograms. the states of all collection subjobs are recalculated, see o.g.

```
glite-lb-job_status -fullhist <collectionJobId>
```

The command `glite-lb-job_status` is a low level query program that can be found in the package `glite-lb-client` among examples.

3.3 L&B LOGGER

The logger component (implemented by `glite-lb-interlogd` daemon fed by either `glite-lb-logd` or L&B proxy) is responsible for the store-and-forward event delivery in L&B (Sect 1.1.2). Therefore eventual operational problems are related mostly to cumulating undelivered events.

3.3.1 EVENT FILES

L&B logger stores events in one file per job, named `$GLITE_LOCATION_VAR/log/dglogd.log.JOBID` by default (JOBID is only the part after the L&B server address prefix). The format is text (ULM [6]), one event per line. In addition, control information on delivery status is stored in additional file with `.ctl` suffix.

In case of emergency (for example corrupted file) the files can be examined with `glite-lb-parse_eventsfile`.[22] It is possible to hand-edit the event files in emergency (remove corrupted lines). However, glite-lb-interlogd must not be running, and the corresponding .ctl file must be removed.

3.3.2 BACKLOG REASONS

Undeliverable jobid. In normal gLite job processing, jobids are verified on job submission (via synchronous job registration, see [2]), hence occurrence of undeliverable jobid (that is its prefix does not point to a working L&B server) is unlikely. On the other hand, if it happens, and an event with such a jobid is logged, for example due to a third-party job processing software bug, glite-lb-interlogd keeps trying to deliver it indefinitely.[23] The unsuccessful attempts are reported via syslog. The only solution is manual removal of the corresponding files and restart of the service.

Corrupted event file. For various reasons the files may get corrupted. In general, corrupted file is detected by glite-lb-interlogd, and it is moved to *quarantine* (by renaming the file to contain "quarantine" in its name). The action is reported in syslog. The renamed files can be removed or repaired by hand and renamed back for glite-lb-interlogd to pick them up again (in this case, the service needn't be stopped).

[22] Not fully supported tool, installed by `glite-lb-client` package among examples.
[23] Unless event expiration is set, though it is not done for normal events.

Slow delivery. Either glite-lb-interlogd or the target L&B server(s) may not keep pace with the incoming stream of events. Amount of such backlog can be quickly assessed by looking at timestamps of the oldest event files (with `ls -lt $GLITE_LOCATION_VAR/log/`, for example). Fully processed event files are deleted in approx. one minute intervals, so files last modified one minute ago or newer do not constitute backlog at all. Unless the backlog situation is permanent, no specific action is required—the event backlog decreases once the source is drained. Otherwise hardware bottlenecks (CPU, disk, network) have to be identified (with standard OS monitoring) and removed, see sections 2.3.1 and 2.3.18 for L&B server tuning tips. In order to maximize performance of `glite-lb-interlogd`, it is recommended to distribute jobs to multiple L&B servers that can be shared by multiple WMS setups in turn.

3.3.3 NOTIFICATION DELIVERY

When `glite-lb-logger` package is optionally installed with L&B server, a modified `glite-lb-notif-interlogd` is run by the server startup script. This version of the daemon is specialized for L&B notifications delivery; it uses the same mechanism, however, the events (notifications) are routed by *notification id* rather than jobid, and targeted to user's listeners, not L&B servers. See [2, 3] for details.

On the contrary to normal events, it is more likely that the event destination disappears permanently. Therefore the notification events have their expiration time set, and glite-lb-interlogd purges expired undelivered notifications by default. Therefore the need for manual purge is even less likely.

The event files have different prefix (`/var/tmp/glite-lb-notif` by default).

3.3.4 DEBUG MODE

All the logger daemons, that is glite-lb-logd, glite-lb-interlogd, and glite-lb-notif-interlogd, can be started with `-d` to avoid detaching control terminal, and `-v` to increase debug message verbosity. See manual pages for details.

3.4 USED RESOURCES

3.4.1 SERVER AND PROXY

Processes and threads. By default L&B server runs as one master process, and 10 slave processes. Threads are not used.

Master pid is stored in the file `$HOME/edg-bkserverd.pid` (*L&B version 1.x*) or in the file `$HOME/glite-lb-bkserv` (*L&B version 2.x*) respectively. Number of slaves can be set with `-s, --slaves` option, pid file location with `-i, --pidfile`.

Slave server processes are restarted regularly in order to prevent memory leakage.

Network and UNIX ports. L&B server listens on port 9000 for incoming queries, 9001 for incoming events, and 9003 for WS interface queries. The former two can be changed with `-p, --port` option (incoming events are always one port higher than queries), the latter with `-w, --wsport`.

When L&B notifications are enabled (automatically in server startup script when glite-lb-logger is also installed with the server), `/tmp/glite-lb-notif.sock` is used for communication with notification interlogger. It can be changed with `--notif-il-sock`.

L&B proxy communicates on two UNIX sockets: `/tmp/lb_proxy_server.sock` (queries) and `/tmp/lb_proxy_store.sock` (incoming events).

L&B version 2.x: as proxy and server are merged, the mode of operation determines the actual server ports (network or UNIX). Notifications are not delivered from proxy-only mode.

IPC. *L&B version 1.x* only: mutual exclusion of server slaves is done via SYSV semaphores. The semset key is obtained by calling ftok(3) on the pid file.

L&B version 2.x does not use semaphores anymore.

Database. Server data are stored in MySQL database. Normal setup assumes "server scenario", that is the database IS NOT protected with password and it is not accessible over network. Option `-m user/password@machine:database` can be used to change the default connect string `lbserver/@localhost:lbserver20`.

Dump files. Backup dump (Sect. 3.2.4) files are stored in `/tmp/dump` by default, however, gLite startup script uses `--dump-prefix` option to relocate them into `$GLITE_LOCATION_VAR/dump`. Similarly dumps resulting from purge operation (Sect. 3.2.5) go to `/tmp/purge` by default, and startup script uses `--purge-prefix` to set `$GLITE_LOCATION_VAR/purge`.

When export to Job Provenance is enabled (Sect. 3.2.5) job registrations are exported to directory `/tmp/jpreg` overridden in startup script with `--jpreg-dir` to `$GLITE_LOCATION_VAR/jpreg`. In addition, dump files are further processed in `GLITE_LOCATION_VAR/jpdump`.

Normal operation of JP export should not leave any files behind, however, abnormal situations (JP unavailable for longer time etc.) may start filling the disk space. Therefore periodic checks are recommended.

Also when purge and dump operations are invoked resulting files are left in their directories and it is the responsibility of the operation caller to clean them up.

Notifications (see above with Network ports) are stored in files with `/var/tmp/glite-lb-notif` prefix. It can be changed with `--notif-il-fprefix`.

Normal operation of notification interlogger purges these files when they are either delivered or expire.

3.4.2 LOGGER

Processes and threads. glite-lb-logd uses one permanent process and forks a child to serve each incoming connection.

glite-lb-interlogd (and derived glite-lb-notif-interlogd) is multithreaded: one thread for handling input, another for recovery, and a dynamic pool of others to deliver events to their destinations.

Network and UNIX ports. glite-lb-logd listens on 9002 (can be changed with `-p` option). The daemons communicate over UNIX socket `/tmp/interlogger.sock`, can be changed with `-s` (for both daemons simultaneously).

Event files are stored with prefix `/var/spool/glite/lb-locallogger/dglogd.log` (changed with `-f`).

4 FAQ—FREQUENTLY ASKED QUESTIONS

4.1 JOB IN STATE 'RUNNING' DESPITE HAVING RECEIVED THE 'DONE' EVENT FROM LRMS

Jobs stay in state *Running* until a *Done* event is received from the workload management system. *Done* events from local resource managers are not enough since the job in question may have been resubmitted in the meantime.

4.2 WMS CANNOT PURGE JOBS OR PERFORM OTHER PRIVILEGED TASKS

In short, WMS has not been given adequate permissions when configuring the L&B server. You need to modify your configuration and restart the server:

4.2.1 FOR *L&B version 3.0.11 or higher*, USING YAIM

Modify your `siteinfo.def`, specifying the DN of your WMS server in YAIM parameter `GLITE_LB_WMS_DN`; for instance:

```
GLITE_LB_WMS_DN=/DC=cz/DC=cesnet-ca/O=CESNET/CN=wms01.cesnet.cz
```

Then rerun YAIM: `/opt/glite/yaim/bin/yaim -c -s site-info.def -n glite-LB`

This will give your WMS exactly the right permissions to carry out all required operations.

4.2.2 FOR ALL VERSIONS OF L&B, USING YAIM

Modify your `siteinfo.def`, specifying the DN of your WMS server in YAIM parameter `GLITE_LB_SUPER_USERS`; for instance:

```
GLITE_LB_SUPER_USERS=/DC=cz/DC=cesnet-ca/O=CESNET/CN=wms01.cesnet.cz
```

Then rerun YAIM: `/opt/glite/yaim/bin/yaim -c -s site-info.def -n glite-LB`

This will give your WMS adequate rights to perform its operations and requests (running purge, querying for statistics, etc.) but it will also grant it additional administrator rights (such as granting job ownership). On newer installations, the method explained in section 4.2.1 is preferable.

4.2.3 FOR *L&B version 2.1 or higher*, WITHOUT YAIM

L&B's authorization settings can be found in file `[/opt/glite]/etc/glite-lb/glite-lb-authz.conf`

Permit actions `PURGE`, `READ_ALL` and `GET_STATISTICS` for your WMS and restart the L&B server. This will lead to results equivalent to 4.2.1. For instance, change the adequate sections in `glite-lb-authz.conf` to:

```
action "READ_ALL" {
        rule permit {
                subject = "/DC=cz/DC=cesnet-ca/O=CESNET/CN=wms01.cesnet.cz"
```

```
        }
}

action "PURGE" {
        rule permit {
                subject = "/DC=cz/DC=cesnet-ca/O=CESNET/CN=wms01.cesnet.cz"
        }
}

action "GET_STATISTICS" {
        rule permit {
                subject = "/DC=cz/DC=cesnet-ca/O=CESNET/CN=wms01.cesnet.cz"
        }
}
```

4.3 L&B SERVER THROWS "DUPLICATE ENTRY ... FOR KEY 1" ERRORS

The L&B server will occasionally report errors through *syslog* saying,

```
ERROR CONTROL - ... : File exists (Duplicate entry '...' for key 1)
```

These error messages are caused by certain portions of code that take care of storing database records for keys, which may or may not already exist in the database, and do so by trying to insert the record first (hence the key violation) and modify the record if the insert fails. This has the unfortunate side effect of the unsuccessful insert being reported as an `ERROR` in the logging output.

Unless you are experiencing trouble with the specific data entity[24] referenced in the error message, it is safe to disregard.

[24] Usually a *Job ID*

REFERENCES

[1] E. Laure, F. Hemmer, F. Prelz, S. Beco, S. Fisher, M. Livny, L. Guy, M. Barroso, P. Buncic, P. Kunszt, A. Di Meglio, A. Aimar, A. Edlund, D. Groep, F. Pacini, M. Sgaravatto, and O. Mulmo. Middleware for the next generation grid infrastructure. In *Computing in High Energy Physics and Nuclear Physics (CHEP 2004)*, 2004.

[2] A. Křenek et al. L&B User's Guide. http://egee.cesnet.cz/en/JRA1/LB/.

[3] A. Křenek et al. L&B Developer's Guide. http://egee.cesnet.cz/en/JRA1/LB/.

[4] A. Křenek et al. L&B Test Plan. http://egee.cesnet.cz/en/JRA1/LB/.

[5] Site Authorisation and Enforcement Services: LCAS, LCMAPS, and gLExec. https://www.nikhef.nl/pub/projects/grid/gridwiki/index.php/LCAS.

[6] J. Abela and T. Debeaupuis. Universal format for logger messages. IETF Internet Draft, 1997.

[7] G. Avellino et al. The DataGrid Workload Management System: Challenges and Results. *Journal of Grid Computing*, 2(4):353–367, Dec 2004.