Something you may have wanted to know about L&B

Zdeněk Šustr, Jiří Sitera, František Dvořák, Jiří Filipovič, Daniel Kouřil, Aleš Křenek, Luděk Matyska, Miloš Mulač, Jan Pospíšil, Miroslav Ruda, Zdeněk Salvet, Michal Voců
CESNET z. s. p. o., Zikova 4, Prague, Czech Republic
E-mail: zdenek.sustr@cesnet.cz

Abstract. Logging and Bookkeeping (L&B) is a gLite subsystem responsible for tracking jobs on the grid. Normally the user interacts with it via glite-wms-job-status and glite-wms-job-logging-info gLite commands. Here we present other, less generally known but still useful L&B usage patterns which are available with recently developed L&B features.

L&B exposes simple HTML (for humans) and plain text (for scripting) interfaces over HTTPS. Hence, for example, pointing a web browser to a jobid yields human-readable page on the job status details. Similarly, the plain L&B server endpoint URL shows a clickable list of an active user’s jobs and notification handles.

Apart of actively querying L&B server, users can also subscribe for receiving notifications on job state changes. Possible criteria range from simple “whatever happens to this job” to “a job of this VO user gets resubmitted to another CE.” We demonstrate how to use notifications via both API and CLI.

1. The gLite L&B Service

The gLite Logging and Bookkeeping (L&B) service is a distributed grid component intended to provide unified and consistent view of a job state in a grid middleware. Its design and implementation concepts are proven by years of production usage in the EGEE grid [1]. The L&B service consists of three major parts:

- **Data gathering part** – L&B collects data on jobs by receiving events (called L&B events) from all grid components jobs pass through (a push model). The L&B infrastructure provides asynchronous, reliable transfer of all events to an L&B server.

- **Data processing and storing part** – the L&B server is not only a storage of delivered L&B events but it also implements a state machine processing the received events into a current job state (see fig. 1). The processing is rather complex and the L&B service is designed to deal with various conditions like resubmissions, temporary failures or wrong system time reported by grid components.

- **Data access part** – both the L&B events (a detailed information valuable for problem tracking) and the detailed job status (a high level view acting as a primary source of information for users) are accessible through a consumer L&B API.

This paper focuses on less known ways of accessing job status information. Section 2 describes L&B notifications – an interface allowing clients to receive messages on any job status change that might be of interest. Section 3 describes a lightweight L&B interface which is
intended to provide users with basic information about grid jobs using just a general web browser or RSS client. For detailed description of the L&B (architecture, components, interfaces), see the L&B User’s Guide [2] and the report [3].

2. L&B Notifications

2.1. Concepts and Architecture

There are two ways to receive job details and information on job state changes from the L&B service API (fig. 2). Both provide roughly the same information, they differ in the mode in which the information is delivered. The first one, a query/response model, is well known and widely used. On the other hand, less known and used L&B notifications are an implementation of a streaming publish/subscribe model.

Essential design concepts and goals of L&B notifications are the following:

- Notifications are managed through notification registrations. An L&B user creates one or more registrations to receive notification messages. Each registration has its own set of conditions (defining when messages should be sent) and its own life cycle. Registrations can be created, used, changed or dropped.

- Notification registrations are soft-state entities. This means that they need to be periodically refreshed, otherwise they are dropped automatically.

- Users do not need to listen for notification messages constantly. Clients can stop and restart as required. The L&B notification infrastructure guarantees that all undelivered messages will wait in a queue (as long as the registration is being refreshed) and will be delivered without loss.

- Users can move, i.e. stop receiving messages generated under a particular registration on one host and start receiving elsewhere.

The L&B query/response part is straightforward – a direct interaction of the L&B client, implementing the API, with an L&B server. This is the same for managing notification registrations. However, notification messages are queued and delivered by an intermediate notification interlogger component (a slightly modified component delivering standard L&B events) attached to each L&B server.

2.2. Typical usage

The L&B notification functionality is available to the users in the L&B client library (C binding, Java is foreseen). Its API is briefly described in sect. 2.4, full reference is given in L&B Developers Guide [4].

The following description and examples use glite-lb-notify command line client. It is a thin wrapper around the API, and it exposes a subset of the API features only (especially applicable job conditions). Nevertheless, the examples can be reproduced immediately, providing a good starting point.

A typical notification workflow consists of three major steps:
Create a new notification registration with job conditions. This is done by running the `glite-lb-notify new` subcommand.

- Each registration (subscription to the notification stream) has its unique identifier called the **Notification ID**. It is returned every time a new registration for notifications is created.
- Notification messages can be potentially generated by the L&B server on every change in the detailed job status information. Several options of the `new` subcommand affect which messages are actually generated, e.g. by limiting the conditions to a specific state (see the command’s man page for details). In particular, the `-c` option instructs the server to send messages only when state transition (an arrow in fig. 1) occurs, i.e. not necessarily on all status detail updates.
- Another important option is `-t timeout`, specifying how long the registration should last (the server may not respect lifetime settings that are too long, though).

Receive notification messages by calling `glite-lb-notify receive` and passing the ID obtained in the previous step as a parameter.

- Even if the client is not listening temporarily, all notification messages generated are being queued, and they are delivered once the client restarts.
- By default, the command line client only displays the Job ID, state and owner for each notification message delivered. However, all job status details\(^1\) are actually included in each message, and it is possible to specify fields to be displayed with the `-f` option.
- The `receive` command waits for messages for limited time only (optionally specified with `-i`). However, this is a timeout for the listening client only, and it should not be confused with validity of the notification registration (i.e. `-t` option of `new` or `refresh` commands); if the client terminates on this timeout, notification messages are typically still queued.

Do not forget to refresh the registration. Each registration is soft-state and it must be refreshed regularly.

- The `glite-lb-notify receive` command has an option to keep refreshing automatically while receiving data (the `-r` option).
- Alternatively, you may use the `glite-lb-notify refresh` command.

\(^1\) with the exception of JDL fields; see the `-f` option of the `new` command to get them
Although `glite-lb-notify` itself provides only limited functionality w.r.t. the API, we did not give its exhaustive description in this section for the sake of readability. For a full description refer to the L&B Users Guide [2] or the command’s man page.

2.3. Specific examples

2.3.1. All my jobs. Any change in any of my jobs’ life cycle will be sent to me as a notification message

(i) `glite-lb-notify new -O -c`
   - The `-O` option denotes all jobs belonging to the current user. It can also be replaced with `-o <DN>`, specifying the user’s DN\(^2\).
   - The `-c` option asks for messages to be sent out only on events that actually trigger state change (sect. 2).

The command outputs e.g.:

```
notification ID: https://skurut68-2.cesnet.cz:9000/NOTIF:6X4btOWUxPsCfSO9KdIW3w
valid: '2009-05-27 13:11:44 UTC' (1243429904)
```

(ii) `glite-lb-notify receive <Notification ID>`
where `<Notification ID>` has been returned in the previous step. The command outputs e.g.:

```
notification is valid until: '2009-05-26 14:12:40 UTC' (1243347160)
https://skurut68-2.cesnet.cz:9000/TPYN912Dr9eEc6gJPVEMEg Submitted
/DC=cz/DC=cesnet-ca/O=Masaryk University/CN=Ales Krenek
```

By default (without `-f`), the `receive` command outputs jobid, state and job owner only.

2.3.2. One job. Subscribe to a notification service for a particular Job Id:

(i) `glite-lb-notify new -j <Job ID>`
   - The `-j` option is used to specify the ID of the job whose life cycle we want to follow.
   - Note that the `-c` option is missing. As a result, notification messages will be generated on all events related to that job, even those not resulting in state change.

Output of the command is similar to sect. 2.3.1.

(ii) `glite-lb-notify receive <Notification ID>`
Output of the command is similar to sect. 2.3.1.

2.3.3. Particular job state only. Send notification messages on events resulting in a particular state. There are two distinct examples:

- Notify the user once a particular job finishes:

  (i) `glite-lb-notify new -j <Job ID> --state done -c`
      - The `-j` option specifies the ID of the job whose life cycle is to be followed.
      - The `--state` option gives a list of states the job needs to match for a notification message to be generated.\(^3\)
      - The `-c` option asks for messages to be sent only on state change (sect. 2).

\(^2\) Getting information on other users’ jobs requires appropriate authorization (privileged user or per-job, see L&B User’s Guide [2]). Without it, requesting `-o /some/other/user` is still valid (L&B server can’t check it at the time of notification registration) but no messages will be delivered.

\(^3\) Only one state (done) is given in this particular example.
Reaching the `done` state means that the job finished execution, either successfully or unsuccessfully. This is further distinguished with `done_code` field in the job status detail which we request with the `-f` option. Together with the default display of jobid and state we get:

```
notification is valid until: '2009-05-26 16:24:48 UTC' (1243355088)
https://skurut68-2.cesnet.cz:9000/E8uE8gPNS8rRBr1ry8Bx1Q Done 0
```

- Notify the user in case any of their jobs aborts or gets cancelled. Since this may not happen for a very long time, keep refreshing the registration while listening.

```
(i) glite-lb-notify new -O --state aborted,cancelled --c
   - The `-O` option denotes all jobs belonging to the current user. It can also be replaced with `-o <DN>`, specifying the user’s DN.
   - The `--state` option gives a list of states the job needs to match for notification messages to be generated.
   - The `-c` option asks for messages to be sent only on state change. Any change of circumstances not resulting in job state change will be ignored.

(ii) glite-lb-notify receive -r -i 36000 <Notification ID>
   - The `-r` option tells the client to keep refreshing the validity of the registration, extending its lifetime to the default value\(^4\) every time the first half of the current interval elapses.
   - The `-i` option tells the client to keep listening for a very long time\(^5\) even if there are no messages coming in. By default, the client only listens for two minutes before terminating.\(^6\)

```
notification is valid until: '2009-05-28 09:34:18 UTC' (1243503258)
next refresh '2009-05-28 09:04:18 UTC' (1243501458)
next refresh '2009-05-28 10:04:18 UTC' (1243505058)
next refresh '2009-05-28 10:34:18 UTC' (1243506858)
https://skurut68-2.cesnet.cz:9000/IrxJaYzFqWsTLU3pzMGfjw Cancelled /DC=cz/DC=cesnet-ca/O=Masaryk University/CN=Ales Krenek
https://skurut68-2.cesnet.cz:9000/55G_JpyH4e6L_jBOLJFrg Cancelled /DC=cz/DC=cesnet-ca/O=Masaryk University/CN=Ales Krenek
next refresh '2009-05-28 11:04:18 UTC' (1243508658)
```

```
https://skurut68-2.cesnet.cz:9000/gYXz_SSHHcXYoFuQQ7xQ Aborted /DC=cz/DC=cesnet-ca/O=Masaryk University/CN=Ales Krenek
```

\(2.4.\) Notification API

Structure of the notification API is reflected in the workflow described in sect. 2.2. There are New, Refresh, and Receive calls. Those are complemented with Bind (internally called also by glite-lb-notify); it tells the server about the local listening address which can be used by multiple subsequent calls to Receive.

The main difference is the richness of conditions used to select when notification messages

\(^4\) typically 1 hour

\(^5\) 10 hours in this case

\(^6\) Put simply: any time a message is received, the counter resets and the client waits for another two minutes.
are delivered. They are specified as a logical formula

\[ (attr_A \ op \ val_A1 \ \lor \ attr_1 \ op \ val_A2 \ \lor \ldots) \]  \hspace{1cm} (1)

\[ \land (attr_B \ op \ val_B1 \ \lor \ attr_1 \ op \ val_B2 \ \lor \ldots) \] \hspace{1cm} (2)

\[ \land \ldots \] \hspace{1cm} (3)

where \( attr_X \) refer to any of approx. 20 job attributes (owner, state, location, destination, return code etc.), and \( op \) is a comparison operator \((<, >, =, \neq)\) or a special \textit{changed} operator (meaning that the referred job attribute was just updated).

See L&B Developer’s Guide [4] for details.

3. Lightweight L&B Interface

Querying the L&B server typically requires the user to use either a standard client installed on the UI (such as \texttt{glite-wms-job-status}), or a custom middleware client implementing L&B querying calls. To allow users access status information on their jobs more easily, L&B also offers a lightweight interface that allows users to access basic information through standard Web browsers or RSS readers.

3.1. HTTPs Querying Interface for Job Status Information

As a Job ID takes the form of an URL, such as:

\[ https://skurut68-2.cesnet.cz:9000/AV-P427QXGo70ohaGk_NQ \]

it is fairly tempting to paste it to a Web browser. And indeed, it works. The only prerequisite is loading the user’s credentials into the browser first, as the L&B server requires...
authentication. A web page giving certain status details for that job, including extended JDL or RSL descriptions, is generated in response to such requests.

Job detail can be viewed by anyone with appropriate permissions, i.e. not only respective job owners, but also L&B server superusers and other users as indicated by applicable ACLs.

The lightweight interface also allows users to get a list of their active jobs registered with the given L&B server by specifying just the server endpoint, e.g.:

```
https://skurut68-2.cesnet.cz:9000/
```

The output is shown in fig. 3.

However, only jobs owned by the user issuing the query are returned. Therefore even privileged users won’t get list of all jobs in this way.

### 3.1.1. Plain Text Output

Job status information and job lists are also available in a plain text “key = value” format. The format is unambiguous (unlike the HTML one), and it is intended for use in scripts. URLs are similar to those above, created by appending a ?text modifier. The functionality is proven to work with tools such as `wget` or `curl`.

An example shows status of a single job:

```
curl -3 --key /tmp/x509up_u62210 --cert /tmp/x509up_u62210
--capath /etc/grid-security/certificates
https://skurut68-2.cesnet.cz:9000/r9eV8p6Ko7YeS_p9PN0gYQ?text
```

Options `--cert`, `--key` and `--capath` provide locations of a user X509 proxy, and certification authorities certificates, respectively.

The command would output e.g.:

```
Job=https://skurut68-2.cesnet.cz:9000/r9eV8p6Ko7YeS_p9PN0gYQ
Status=Done
owner=/DC=cz/DC=cesnet-ca/O=Masaryk University/CN=Ales Krenek
condorId=new jobId (Condor Globus ...)
reason=reason for the change
expectUpdate=NO
location=LogMonitor/harad.ics.muni.cz/
destination=destination CE/queue
cancelling=NO
done_code=1
jdl=# hello.jdl
Executable = "/bin/echo";
Arguments = "Hello World";
StdOutput = "message.txt";
StdError = "stderr";
OutputSandbox = {"message.txt","stderr"};
# hello.jdl end
```

This kind of output can be easily parsed by any tool, typically a simple user script.

### 3.2. Notification Overview over HTTPs

Similar to Job IDs, notification IDs take the form of URLs as well, e.g.:

```
https://skurut68-2.cesnet.cz:9000/NOTIF:ibzNWsRro9ZNTGl5PeNeoQ
```

It is possible to check the existence and expiration times of notification registrations, and a few other details by opening the related Notification ID in a Web browser. It is also possible to get a list of all registrations created by the current user by appending the /NOTIF suffix to the L&B server’s address and port, and using that as an URL:

```
https://skurut68-2.cesnet.cz:9000/NOTIF
```
Moreover, the $text$ modifier described above can also be applied to notification-related URLs. However, this interface refers to notification registrations only, it does not give actual notification messages. RSS feeds discussed below can be used to achieve this purpose.

### 3.3. RSS Feeds to Convey Job State Information
L&B servers publish RSS feeds offering up-to-date information on jobs reaching certain predefined states. As of this writing, there are three different types of feeds available:

- **RSS:finished** for jobs in terminal states (Done/OK, Aborted and Canceled)
- **RSS:running** for running jobs
- **RSS:aborted** for aborted jobs

Feed addresses consist of the https:// prefix, the L&B server address and port, and the title of the required feed, for instance:

https://skurut68-2.cesnet.cz:9000/RSS:finished

Consistent L&B authorization rules apply here as well, therefore your grid credentials need to be loaded into the RSS reader.

### 4. Related work
The whole L&B architecture complies with the general Grid Monitoring Architecture standard (GMA, [5]). The L&B notification interface provides publish/subscribe interaction between data producers and consumers, which makes it – in the sense of GMA – similar to the RGMA streaming interface [6], where any client can subscribe to receive (monitoring) messages based on given conditions. While the RGMA infrastructure is more general than L&B, there are other works relying on technologies that are even more general and commonly used out of the scope of grid monitoring – MOM, Messaging Oriented Middleware, usually an implementation...
of the Java Message Service [7]. For cases where such a general messaging infrastructure is found applicable to a particular use case, L&B can be easily instrumented to forward stream of messages to such an infrastructure for further handling. This was implemented and used with RGMA in the context of the EGEE project. However, a preferred approach is to generate those messages from L&B notifications; currently this is done for Messaging System for Grids (MSG)[8] used by the Dashboard project[9] (as reported in [10]). Another high-level monitoring system, Real Time Monitor (RTM) historically retrieved data from L&B backend database. Recent development [11] moves this to L&B notifications as well.

Yet another related work in the area of grid job state monitoring is the CEMon [12], implementing a publish/subscribe interface and supporting job status information monitoring through one of its sensors. It is, by design, aimed at providing information coming from Computing Elements, ranging from general grid component monitoring to job status information. Unlike L&B, there is no concept of events being gathered from various grid components and transformed into common job status information.

What makes the L&B lightweight interface so elegant is the fact that the jobid is in fact an URL pointing to an L&B server as a regular HTTP address. A Globus GRAM[13] jobid has a similar structure (which inspired L&B), however, GRAM jobid is not directly usable in web browser due to GSI vs. SSL incompatibilities. Other systems use similar jobid structure but not accessible in a plain web browser, for example UNICORE[14] uses a WS endpoint reference.

The ARC middleware[15] adopts a different approach. The jobid is a URL of the job session directory (gsiftp://...), allowing users to access files in that directory (using appropriate data management tools).

An approach similar to that implemented by the L&B lightweight interface can be found in the data management area. While a typical file id is an URI (for example in the SRM specification, the file id (SURL) looks like srm://...), DPM[16] supports HTTP interface allowing access to files and directories by a plain web browser (https://<DPM-head-node/path).

5. Summary
Logging and Bookkeeping, a grid job tracking service, has been deployed for fairly long time, since the EU DataGrid project. However, although it can provide valuable information to the users, it has been used in a rather limited way so far. Having assessed the observed L&B usage patterns, we assume that this happened because of the lack of certain class of interfaces.

First of all, the official glite UI – being generic by design – provides commands covering only the most typical L&B usage scenarios. Moreover, the UI implementation can be hardly considered as lightweight. Many 3rd party libraries are required and the number of supported platforms is limited. UI is also designed to be used directly by the end users, therefore its human-readable outputs are not very suitable for scripting.

On the other hand, the L&B API exposes the full L&B functionality but its usage requires non-trivial programming skills, which is probably the reason that wide adoption by the user community did not happen. The same holds for the L&B web-service interface.

Work described in this paper tries to overcome these disadvantages. The simple HTML interface (sect. 3) makes L&B data available to users via any web browser, while the plain text interface is easy to use in scripts (shell, python, perl...) thanks to its unambiguous machine-readable output format.

L&B notifications (sect. 2) allow users to avoid repeated queries for job status. Their functionality has been considerably extended in recent L&B release7. In order to complement the complex API, we provide also a CLI notification client. Its functionality is somewhat limited

7 L&B 2.0, being finalized at the time of writing this manuscript
(w.r.t. the API), however, it covers the most typical use cases, and it is suitable for both direct use and for scripting.

All the described features and improvements emerge either from direct requirements expressed by the users, or from our evaluations of L&B usage patterns. We believe that this work will improve general usability of L&B, and we look forward to further user feedback and suggestions.

6. References

[1] Ruda M et al. 2008 CESNET Conference 2008 Proceedings pp 3–12 Prague, Czech Republic
[2] Křenek A et al. L&B Users Guide http://egee.cesnet.cz/cvsweb/LB/LBUG.pdf
[3] Matyska L et al. 2007 Job tracking on a grid—the Logging and Bookkeeping and Job Provenance services Tech. rep. CESNET
[4] Křenek A et al. L&B Developers Guide http://egee.cesnet.cz/cvsweb/LB/LBDG.pdf
[5] Tierney B et al. A Grid Monitoring Architecture http://www.gridforum.org/documents/GFD.7.pdf
[6] Cooke A W et al. 2004 Journal of Grid Computing 2
[7] Java Message Service specification http://java.sun.com/products/jms
[8] WLCG messaging system for grids (MSG) https://twiki.cern.ch/twiki/bin/view/LCG/MessagingSystemforGrid
[9] Andreeva J et al. 2007 GMW07, June 25, 2007, Monterey, California, USA
[10] Andreeva J et al. 2009 Proc. CHEP`09
[11] Martyniak J et al. 2009 Proc. CHEP`09
[12] Garzoglio G et al. 2008 Grid Computing International Symposium on Grid Computing (ISGC 2007) pp 89–98
[13] The Globus toolkit http://www.globus.org/
[14] UNICORE (uniform interface to computing resources) http://www.unicore.eu/
[15] Advanced resource connector, ARC http://www.nordugrid.org/middleware/
[16] Web access to DPM storage https://twiki.cern.ch/twiki/bin/view/LCG/DpmHttpsAccess