GLU Console

Introduction
The GLU console is the orchestrator of the whole system. It offers both a user friendly web interface and a REST api (for the command line).

Configuration and bootstrap
For dev
Check the document called `1-QUICK-DEV-SETUP.txt` for detailed instructions on how to set it up

For production
The console is packaged as a regular webapp (war file) and can simply be dropped in any servlet container (tested with tomcat). In order to run the console requires a configuration file. You can find a sample in the `docs` folder. There are 3 ways to provide this configuration file to the console:

1. store it under `$HOME/.org.linkedin.glu/glu-console-webapp.groovy`
2. store it under `conf/glu-console-webapp.groovy` (relative to wherever the VM user.dir is set to)
3. pass a system property `-Dorg.linkedin.glu.console.config.location=/fullpath/to/file`

First bootstrap
The very first time, the console is started, it will create an admin user. Log in as this user:
```
username: admin
password: admin
```

Then click on the 'admin' tab (not the one called 'Admin') and click 'Manage your credentials'.
It is strongly recommended you immediately change the admin password!

User management
There are 2 ways to manage user.

LDAP
If you define an ldap section in the configuration file (see `console.groovy` example), then the console will automatically allow any user who has the correct ldap credentials to login. If the user has never logged in before, a new account will be automatically created and the user will have the role "`USER`" which gives him read access mostly (and limited access to log files). This allows any user to login to the console without any administrator intervention.

Manual user management
Whether you use LDAP or not you can always use this method. If you don't use LDAP then it is your only choice. On the Admin tab, you can create a new user by giving it a user name and a password. You can also assign roles to a user:

`USER`: mostly read access (limited access to some log files (cannot go anywhere on the filesystem))
`RELEASE`: most of the traditional release actions (like starting and stopping entries…)
`ADMIN`: administrative role (create user, assign roles…)
`RESTRICTED`: if you want to ban a user from the console

Note that no matter what the role the user in, actions taken are logged in the audit log that can be viewed from the Admin tab.

Password management
Even if you use LDAP, a user can assign himself a console password (useful if the password need to be stored in scripts for example).

Console tabs
Documentation coming soon. In the meantime, take a look at `2-QUICK-TUTORIAL.txt`.

REST API
Security / Authorization
The security model is simply implementing http basic authorization:

every request needs to come with a Authorization header with the following format:
```
Authorization: Basic <base64(username:password)>
```

The username and password are only slightly obfuscated (base 64) but it is not an issue because the cli talks to the console over https and as a result the headers are never traveling over an insecure channel.

API
Main URI: `/console/rest/v1/<fabric>` (all the URIs in the following tables starts with the main URI)

Method	URI	Description	Request	Response	Status
`GET`	`/plans`	List all the plans	N/A	TBD	✗
`POST`	`/plans`	Create a plan	view details below for the content of body of the POST	• 201 (CREATED) with `Location` header to access the plan (`/plan/<planId>`) • 204 (NO CONTENT) when no plan created because there is nothing to do	✓
`GET`	`/plan/<planId>`	View the plan (as an xml document)	N/A	• 200 (OK) with an xml representation of the plan • 204 (NOT_FOUND) if no such plan	✓

POST	/plan/<planId>/execution	Executes the plan	N/A	• 201 (CREATED) with Location header to access the plan execution (/plan/<planId>/execution/<executionId>). Note that it is a non blocking call and it returns right away and you can check the progress thus allowing to have a progress bar! • 204 (NOT_FOUND) if no such plan	✅

HEAD	/plan/<planId>/execution/<executionId>	Returns the status of the execution	N/A	• 200 (OK) with X-LinkedIn-GLU-Completion header with value:
				  • if plan non completed, percentage completion (ex: 87)
				  • if completed: 100:<completion status> (ex:100:FAILED or 100:COMPLETED)
				• 204 (NOT_FOUND) if no such execution
					✅

GET	/plan/<planId>/execution/<executionId>	Returns the execution as an xml document	N/A	• 200 (OK) with an xml representation of the execution (equivalent to the view in the console) • 204 (NOT_FOUND) if no such execution	✅
DELETE	/plan/<planId>/execution/<executionId>	Aborts the execution	N/A	TBD	❌
PUT	/system/model	Loads the manifest in the console	Body contains a query string rootUrl=file:/xxx&release=yyy	• 201 (CREATED) when loaded successfully • 204 (NO_CONTENT) if model was loaded successfully and is equal to the previous one • 400 (BAD_REQUEST) if rootUrl is not a file (note that it is temporary as well) • 404 (NOT_FOUND) when error (note error handling needs to be revisited)	✅
GET	/system/model	Retrieves the current loaded model (aka 'desired' state)	optional request parameters: • prettyPrint=true for human readable output • systemFilter=... for filtering (see below for the syntax) • legacy=true for keeping the legacy section in the output	• 200 (OK) with a json representation of the model	✅
GET	/system/live	Retrieves the current live model coming from ZooKeeper (aka current state)	optional request parameters: • prettyPrint=true for human readable output • systemFilter=... for filtering (see below for the syntax) • legacy=true for keeping the legacy section in the output	• 200 (OK) with a json representation of the live model (note that the metadata contains information like currentState	✅

Filter syntax

The filter syntax is a dsl and allows you to filter by any field of the entry using a dotted notation. An entry looks like this (you get this when doing /system/live or /system/model):

```
{
  "agent": "ei2-app3-zone5.qa",
  "initParameters": {
    "skeleton": "ivy:/com.linkedin.network.container/container-jetty/0.0.007-RC1.1",
    "wars": "ivy:/com.linkedin.jobs/jobs-war/0.0.504-RC3.2612|jobs"
  },
  "metadata": {
    "container": {
      "kind": "servlet",
      "name": "jobs-server"
    },
    "currentState": "running",
    "modifiedTime": 1284583501275,
    "product": "network",
```

```
      "version": "R950"
    },
    "mountPoint": "/jobs-server/i001",
    "script": "ivy:/com.linkedin.glu.glu-scripts/glu-scripts-jetty/3.0.0/script"
  }
```

The dsl has the following syntax:

- and / or / not => to do logic
- <dotted notation>='<value>' => to express the matching criteria

Examples:

1. Only container 'jobs-server'

```
metadata.container.name='jobs-server'
```

2. Container is 'jobs-server' or 'activemq'

```
or {
  metadata.container.name='jobs-server'
  metadata.container.name='activemq'
}

// can be compacted on 1 line as:
or{metadata.container.name='jobs-server';metadata.container.name='activemq'}
```

3. All containers that are not running (on live system only of course)

```
not {
  metadata.currentState='running'
}

// can be compacted on 1 line as:
not{metadata.currentState='running'}
```

4. All containers not running on agent ei2-app3-zone5.qa (on live system only of course)

```
not {
  metadata.currentState='running'
}
agent='ei2-app3-zone5.qa'

// is 100% equivalent to:
and {
  not {
    metadata.currentState='running'
  }
  agent='ei2-app3-zone5.qa'
}

// can be compacted on 1 line as:
not{metadata.currentState='running'};agent='ei2-app3-zone5.qa'
```

The REST api is expecting the filter as a query parameter (systemFilter) and it needs to be properly url encoded. For example it should be systemFilter=not%7bmetadata.currentState%3d'running'%7d

Representing a plan

Query String based

The POST you issue must be of `Content-Type application/x-www-form-urlencoded`

The body then contains a query string with the following parameters:

name	value	required	example
`planAction`	`start`, `stop`, `bounce`, `deploy`, `undeploy`, `redeploy`	Yes	`planAction=start`
`systemFilter`	a system filter as described in the previous section (remember that it **must** be properly url encoded.	No. It simply means don't filter at all.	`systemFilter=and%7bagent%3d'ei2-app3-zone5.qa'%7d`
`order`	`parallel` or `sequential`	No. Default to `sequential`	`order=parallel`

Json/DSL Based

Coming soon