Getting Started with Serpens extension for Kepler 2.2.x

Working with gLite module

{marcinp, michalo, tzok}@man.poznan.pl

gLite module description covers:

Listing of services on gLite infrastructure
Data management in grid environment
Job submission and management using gLite

This paper covers Serpens 2.2.1 release

July 20, 2011

Getting Started with Serpens – gLite module

Getting Started with Serpens – gLite module guide is for scientists who would like to use Serpens suite in Kepler. It was created to assist Kepler users to manage grid resources from the level of Kepler 2.2.x workflow.

Table of Contents

1 Introduction

This guide introduces the main components and functionality of Serpens gLite module. It is an add-on module for Kepler, a software application for the analysis and modeling of scientific data and it provides functionality for data and job management on gLite environment from the level of Kepler workflow. This module needs a Roaming Access Server to work.

1.1 What is gLite

From official website http://glite.cern.ch/:

gLite (pronounced "gee-lite") is the next generation middleware for grid computing. Born from the collaborative efforts of more than 80 people in 12 different academic and industrial research centers as part of the EGEE Project, gLite provides a framework for building grid applications tapping into the power of distributed computing and storage resources across the Internet.

The gLite distribution is an integrated set of components designed to enable resource sharing. In other words, this is middleware for building a grid.

The gLite middleware was produced by the EGEE <http://www.eu-egee.org> project and it is currently being developed by the EMI <http://www.eu-emi.eu> project. In addition to code developed within the project, the gLite distribution pulls together contributions from many other projects, including LCG <http://lcg.web.cern.ch/LCG/> and VDT <http://vdt.cs.wisc.edu/components/vdt.html>. The distribution model is to construct different services ('node-types') from these components and then ensure easy installation and configuration on the chosen platforms (currently Scientific Linux <https://www.scientificlinux.org> versions 4 and 5, and also Debian 4 <http://www.debian.org> for the WNs).

gLite middleware is currently deployed on hundreds of sites of different DCIs and enables global science in a number of disciplines, notably serving the LCG <http://lcg.web.cern.ch/LCG/> project.

1.2 User Certificate

In gLite environment, each user is authenticated and authorized with his certificate and private key. These are provided by local Certificate Authorities (CA). Please contact grid CA available in your country in order to obtain your pair of certificate and private key.

Grid resources are assigned to Virtual Organisations (VOs). When you have your credentials ready, you need to apply to VO managers to be added as infrastructure user. Please search for registration page for the specific VO of your choice Once you are confirmed of successful addition, you can continue with Serpens suite for Kepler.

1.3 What is Roaming Access Server (RAS)

Roaming Access Server is a middle layer software between grid infrastructure and client libraries/applications. RAS has its own configuration which lets administrators define options and infrastructure specific features. It has all grid certificates and software installed. RAS communicates with grid services and manages connections to storages and workload managers.

Serpens gLite module communicates with RAS and all its actors are dependent on it.

1.4 What is Serpens gLite module?

Serpens gLite module provides a way for accessing grid resources through the Kepler workflow. It allows users to:

- list available services for given grid virtual organisation,
- upload and download many files in parallel,
- prepare, submit and resubmit jobs,
- manage jobs.

Additionally, gLite module contains error prevention mechanisms that work in the background of actors behaviour. It gathers information about encountered failures of storage and computing elements and later reuses working and avoids broken parts. It is also assured that simple job submission error will be recovered automatically if possible.

1.5 Prerequisites

In order to be authorized to work with specific grid virtual organisation one must have a grid proxy. It can be generated with `grid-proxy-init` or `voms-proxy-init` commands.

2 Using Serpens gLite module

Once Serpens gLite module is installed in Kepler, you will notice new actors available at the Component list.

Actor name	Purpose
RASi2gLogin	Prepares workflow to work with grid services.
RASi2gGetSRMList	Retrieves a list of available storages.
RASi2gGetWMSList	Retrieves a list of available workload managers.
RASi2gUploadFiles	Uploads files to remote locations.
RASi2gDownloadFiles	Downloads files from remote locations.
RASi2gCreateJobDetailsXML	Prepares an internal XML representation of job.
RASi2gSubmit	Submits a job.
RASi2gResubmit	Resubmits a job upon failure.
RASi2gGetJobsStatuses	Retrieves statuses of an array of jobs.
RASi2gGetJobChildren	Retrieves IDs of children subjobs.
RASi2gGetJobOutputsLocation	Retrieves remote locations of job's output files.
RASi2gGloginTerminal	Invokes a terminal to communicate interactively with grid worker node.

2.1 Preparing to work with grid services

Serpens gLite uses RAS as a middle layer to communicate with grid services. This is a very convenient solution, however in some cases a direct approach has more advantages. When data transfer is considered user can decide to depend solely on RAS (acting then like a tunnel) or to actively and directly upload/download files. To make it work, user needs credentials.

2.1.1 RASi2gLogin

To obtain grid services certificates, you can use RASi2gLogin actor and provide data for the following ports:

Input port/parameter	Description
`hostPort`	Port to receive address of Roaming Access Server (RAS), which is a layer between Kepler and gLite middleware. RAS address needs to be specified in form: PROTOCOL://ADDRESS:PORT where PROTOCOL is http or https eg. http://ras.example.com:8080
`proxyPort`	Port to receive proxy data. This must be proxy signed by Virtual Organisation Management Service (VOMS) server.

Actor has one active output port (the remaining ones are obsolete and deprecated):

Output port	Description
`isOkPort`	Port to send if everything was alright.

2.2 Retrieval of information about grid infrastructure

Grid composes of many distributed and generally heterogeneous services. In order to work with them, one needs to retrieve information from centralized index called BDII (Berkeley Database Information Index). Such index exists for each grid Virtual Organisation and contains data about available resources, constraints, possibilities, etc.

2.2.1 RASi2gGetSRMList

Serpens gLite module contains two most necessary actors to retrieve information from BDII. RASi2gGetSRMList retrieves available SRMs (Storage Resource Managers) from which user chooses an appropriate one to use for data storage. To use this actor, please provide data to its ports:

Input port/parameter	Description
`hostPort`	Port to receive address of Roaming Access Server (RAS), which is a layer between Kepler and gLite middleware. RAS address needs to be specified in form: PROTOCOL://ADDRESS:PORT where PROTOCOL is http or https eg. http://ras.example.com:8080

Input port/parameter	Description
`proxyPort`	Port to receive proxy data. This must be proxy signed by Virtual Organisation Management Service (VOMS) server.

Its output ports are:

Output port	Description
`defaultSrmPort`	First Storage Element from the list.
`srmList`	A list of accessible Storage Elements for Virtual Organisation for which your proxy is signed.

2.2.2 RASi2gGetWMSList

Another actor useful in working with grid is RASi2gGetWMSList which will retrieve available WMSes (Workload Management Systems). WMS is responsible for scheduling jobs to specific CEs (Computing Elements) ensuring that all jobs requirements will be met. It is a crucial element in job submission process. To obtain WMS list, please send needed data to RASi2gGetWMSList actor:

Input port/parameter	Description
`hostPort`	Port to receive address of Roaming Access Server (RAS), which is a layer between Kepler and gLite middleware. RAS address needs to be specified in form: PROTOCOL://ADDRESS:PORT where PROTOCOL is http or https eg. http://ras.example.com:8080
`proxyPort`	Port to receive proxy data. This must be proxy signed by Virtual Organisation Management Service (VOMS) server.

Its output ports are:

Output port	Description
`defaultWmsAddressPort`	First WMS from the list.
`wmsAddressList`	A list of accessible Workload Management Systems for Virtual Organisation for which your proxy is signed.

2.3 Data management

Grid infrastructure has very robust and complex data management system. There are many storage services involved, which ensure your data is stored safely. Possibly, this allows huge amounts of data to be produced and easily replicated among different storages.

Serpens gLite module contains a single actor for uploading and one for downloading of remote, grid files. Both actors have some common parameters.

First of all, data transfer may be direct or tunneled. In the first case, user running Kepler is required to have 20000-25000 port range opened for TCP connections and to have grid certificates loaded. The latter is achieved by firing RASi2gLogin actor before any data transfer, but the former is not always possible. Thus, there is the other option – tunneled connections – which will leave all data transfers from storages to RAS, and user will get it from RAS in the next step. This puts RAS in a role of proxy in data transfer process, which means it adds some burden to RAS itself and slows down upload/download. So, it is always advised to use direct connections wherever possible.

Both actors run each of their actions in parallel. User can decide how many transfers should take place at the same time, however the default value is reasonable to be left – more transfer threads actually slow down the process.

Actions taken by actors are fully controlled for validity in each step. Data management on grid has a few layers and problems may occur on each of them. For example, successful transfer may be done, but file registration fails. Using Serpens actors to upload and download files ensures that the whole process is successful.

This goes even further, because upon failure actors will retry with other grid storage service (there is a retry limit, not to regress infinitely). Then, after each successful transfer, Serpens incorporates information about valid infrastructure elements in its internal database. Later, when another data transfer takes place, this information is reused. This all takes place in the background of normal workflow design and run activity, so that user can rely on Serpens gLite module and do not care about some temporary problems with infrastructure.

2.3.1 RASi2gUploadFiles

To upload files to grid storages, please use RASi2gUploadFiles actor:

Input port/parameter	Description
`fileArrayPort`	Port to receive array with names of files to be uploaded.
`isTunnelledParameter`	Parameter saying if connection should be tunneled. For information about tunneled/direct connections, please check description of this actor.
`proxyPort`	Port to receive proxy data. This must be proxy signed by Virtual Organisation Management Service (VOMS) server.
`rasPort`	Port to receive address of Roaming Access Server (RAS), which is a layer between Kepler and gLite middleware. RAS address needs to be specified in form: PROTOCOL://ADDRESS:PORT where PROTOCOL is http or https eg. http://ras.example.com:8080
`retriesParameter`	Parameter saying how many retries should there

Input port/parameter	Description
	be. If grid infrastructure fails, then this actor will try to upload files again, but using different Storage Element. However it may be that upload is impossible whatever SE is chosen. Then after some retries, actor will stop.
`seParameter`	Parameter saying which Storage Element to choose. This parameter is optional - when left blank, actor will randomly choose any working SE. Also please note that upon failure, this actor retries with different SE, so please don't rely on setting this parameter - its value may change. SE needs to be specified in form ADDRESS eg. se.example.com
`threadsParameter`	Parameter saying how many uploading threads should there be. Parallel data transfer is beneficial, however only to some extent. Using too many threads at once will lead to performance decrease. Another thing to notice is that with tunnelled connections, too many parallel uploads will greatly decrease whole RAS performance, so change this parameter sensibly.

Its output port is:

Output port	Description
`lfnArray`	Port to send an array of LFNs (addresses of uploaded files).

2.3.2 RASi2gDownloadFiles

To download files, there is the RASi2gDownloadFiles actor. Before actual downloading, it checks if remote files exists and if they are correct. User can switch downloading off and leave an instance of RASi2gDownloadFiles to only check if files exist.

This actor receives an input token with base output path. However it does not store files directly in this directory. It first creates a unique subdirectory in each firing and in the end it outputs full path to it. This was introduced to allow looping of this actor's action.

Input port/parameter	Description
`isCheckOnlyParameter`	Parameter saying if only file existence check should be done. This actor first checks if files are accessible. It may finish after this step and just inform about the results it got. It may also continue and download files. This parameter defines actor's functionality.

Input port/parameter	Description
`isTunnelledParameter`	Parameter saying if connection should be tunneled. For information about tunneled/direct connections, please check description of this actor.
`lfnArrayPort`	Port to receive an array of LFNs. These are logical names for files in remote storage. They must be specified in form similar to UNIX-like paths eg. /grid/vo/some/file
`outputPathPort`	Port to receive base output path. This is a directory where new ones will be created.
`proxyPort`	Port to receive proxy data. This must be proxy signed by Virtual Organisation Management Service (VOMS) server.
`rasPort`	Port to receive address of Roaming Access Server (RAS), which is a layer between Kepler and gLite middleware. RAS address needs to be specified in form: PROTOCOL://ADDRESS:PORT where PROTOCOL is http or https eg. http://ras.example.com:8080
`retriesParameter`	Parameter saying how many retries should there be. If grid infrastructure fails, then this actor will try to upload files again, but using different Storage Element. However it may be that upload is impossible whatever SE is chosen. Then after some retries, actor will stop.
`seParameter`	Parameter saying which Storage Element to choose. This parameter is optional - when left blank, actor will randomly choose any working SE. Also please note that upon failure, this actor retries with different SE, so please don't rely on setting this parameter - its value may change. SE needs to be specified in form ADDRESS eg. se.example.com
`threadsParameter`	Parameter saying how many uploading threads should there be. Parallel data transfer is beneficial, however only to some extent. Using too many threads at once will lead to performance decrease. Another thing to notice is that with tunnelled connections, too many parallel uploads will greatly decrease whole RAS performance, so change this parameter sensibly.

Its output ports are:

Output port	Description
`isCorrect`	Whether all specified files are downloadable.
`generatedOutputPathPort`	This actor generates a unique name and creates new directory. This port will receive the generated name of directory (absolute path).

2.4 Job management

2.4.1 RASi2gCreateJobDetailsXML

A job is not simply a statement to be executed, but also its input/output files, environment variables and other settings. So before actual submission, one needs to prepare job's description. RAS uses internal, XML-based format for this purpose and user is presented with an actor to create such XML description. In order to use RASi2gCreateJobDetailsXML, one needs to post data to the following ports:

Input port/parameter	Description
`defaultSEPort`	Address of Storage Element (SE) to use. It is responsible for providing a uniform access to data storage resources. To get a list of working Storage Elements, please see RASi2gGetSRMList.
`envSettingsListPort`	Environment variables to be set on Working Node (WN) - a machine which will execute the job. It needs to be specified in form: "name:value" eg. LFC_HOST:lfc.example.com
`executablePort`	Executable name. For preinstalled applications, use full path eg.: /bin/ls. For those uploaded to execution directory, use relative path eg.: ./my-executable
`inputFilesListPort`	List of input files that are stored on the Storage Element. The format of names is: "name:location" or only "location". Location is an LFN – a logical name for grid remote storage. For example – input.in:/grid/vo/user/inputs/input01
`jobTypePort`	Type of the job - please use "normal", "interactive", "openmpi" or "parametric". Normal job is a single-core program. Interactive uses i2glogin application which allows to pass data through input/output/error streams directly between Working Node (WN) and client side (in this context another Kepler actor is a client - please see for example RASi2gGloginTerminal). OpenMPI job reserves given number of cores on Computing

Input port/parameter	Description
	Element (CE). Parametric allows to submit any number of jobs at once - they are created iteratively. You can use numerically generated parameters (eg. starting from 1 to 10 with step 2 will submit 5 jobs with values 1, 3, 5, 7, 9). You can also use a predefined list of parameters (eg. "one", "two", "three"). To set details about parameter generation - please see ports parametricType, parametricLimit, parametricList, parametricStart and parametricStep.
`nodesPort`	Number of cores to use. Only applicable for MPI jobs.
`outputFilesListPort`	List of names of expected output files. This must be done before job is submitted. If you don't know what output files will be produced, then set here "output.tar.gz" and in your grid application pack everything into such archive. Please note: standard output and standard error streams are always returned as StdOutput and StdError. Do not specify them here.
`parametricLimitPort`	If numeric values for parametric job are used, then this port will read the final value.
`parametricListPort`	If list of values for parametric job is used, then this port will read this list.
`parametricStartPort`	If numeric values for parametric job are used, then this port will read the starting value.
`parametricStepPort`	If numeric values for parametric job are used, then this port will read the step value.
`parametricTypePort`	For parametric jobs, type of parameters used. Only applicable values are: "list" or "numeric"
`proxyPort`	Port to receive proxy data. This must be proxy signed by Virtual Organisation Management Service (VOMS) server. To obtain this proxy, please check vomsproxy.xml workflow.
`rankPort`	Stochastic rank policy used by Workload Management System (WMS) to choose the best Computing Element (CE) to run current job. This port is optional - if left blank, actor will default to least_waiting_jobs. If set, it needs to be one of predefined values: best_benchmark, greatest_available_cpu_time, max_available_cpu_time, benchmark_spec_float,

Input port/parameter	Description
	max_number_of_free_cpu, best_data_access_cost, shortest_time_to_traverse_queue, greatest_main_memory, min_number_of_handled_jobs, benchmark_spec_int, rank_least_waiting_jobs, least_waiting_jobs.
rasPort	Port to receive address of Roaming Access Server (RAS), which is a layer between Kepler and gLite middleware. RAS address needs to be specified in form: PROTOCOL://ADDRESS:PORT where PROTOCOL is http or https eg. http://ras.example.com:8080
requirementsPort	Job requirements specified exactly in the same way as in Job Description Language (JDL).
targetHostPort	Address of Computing Element (CE). CE stands for machine where job will be executed. It needs to be specified in form ADDRESS:PORT/QUEUE eg. ce.example.com/2119/jobmanager-pbs-short. You can also leave this port blank and rely on value set on "rank" port. Then the Workload Management System (WMS) will choose the best CE for your job.
wmsAddressPort	Port to receive an address of Workload Management System (WMS) to use. WMS is responsible for distribution and management of tasks across grid resources. It needs to be specified in form ADDRESS:PORT/SERVICE. Please note that most often, WMS will be presented with protocol name in the beginning (eg. http://...), but this actor requires it to be dropped. WMS format readable by this actor does not allow protocol name eg. wms.com:7114/glite_wms_wmproxy_server

It has one output port:

Output port	Description
jobDetailsXML	Port to send job description in XML format readable by RAS.

2.4.2 RASi2gSubmit

When you have JobDetails in XML format, you can submit a job using RAS. This process is however not always error-free, because grid environment is by definition heterogeneous and not equally loaded so you may find a specific WMS rejecting your job. RASi2gSubmit actor can take care of such situations and automatically retry with different resources. To use this actor, please provide the following ports with data:

Input port/parameter	Description
hostPort	Port to receive address of Roaming Access Server (RAS), which is a layer between Kepler and gLite middleware. RAS address needs to be specified in form: PROTOCOL://ADDRESS:PORT where PROTOCOL is http or https eg. http://ras.example.com:8080
jobDetailsXMLPort	Port to receive an XML specification of job to be submitted. To get such XML specification, please see RASi2gCreateJobDetailsXML actor.
proxyPort	Port to receive proxy data. This must be proxy signed by Virtual Organisation Management Service (VOMS) server. To obtain this proxy, please check vomsproxy.xml workflow.
retryCount	Parameter saying how many retries should there be. It may happen that Workload Management System (WMS) refuses to process new job, either because of external problems (infrastructure failures, etc.) or internal ones (too many connections, etc.). In such case, this actor will choose different WMS and try again. However it may be that job submission is impossible whatever WMS is chosen. Then after some retries, actor will stop.

Its output ports are:

Output port	Description
jobIdPort	Port to send an ID of submitted job.
output	Not used by this actor.
errorCodePort	Not used by this actor.
errorMessagePort	Not used by this actor.

2.4.3 RASi2gGetJobChildren

In gLite grid infrastructure, one can submit parametric jobs. In their specification, every occurrence of string _PARAM_ is replaced by some value. User has two options to follow, numeric – which provides _PARAM_ in iterative manner – or list – which uses given array of values for this purpose. Submission of such job means, that under one job ID, there are numerous subjobs. They as well have their own IDs and independent statuses. In order to get this information, you can use RASi2gGetJobChildren actor.

Parametric job is a very efficient and elegant way of submitting many jobs and running them in parallel. However it is also constraining to infrastructure and it may happen that workload managers refuse to process such job. In such case, RASi2gGetJobChildren actor will attempt to resubmit it using different resources.

To use this actor, please provide data to its ports:

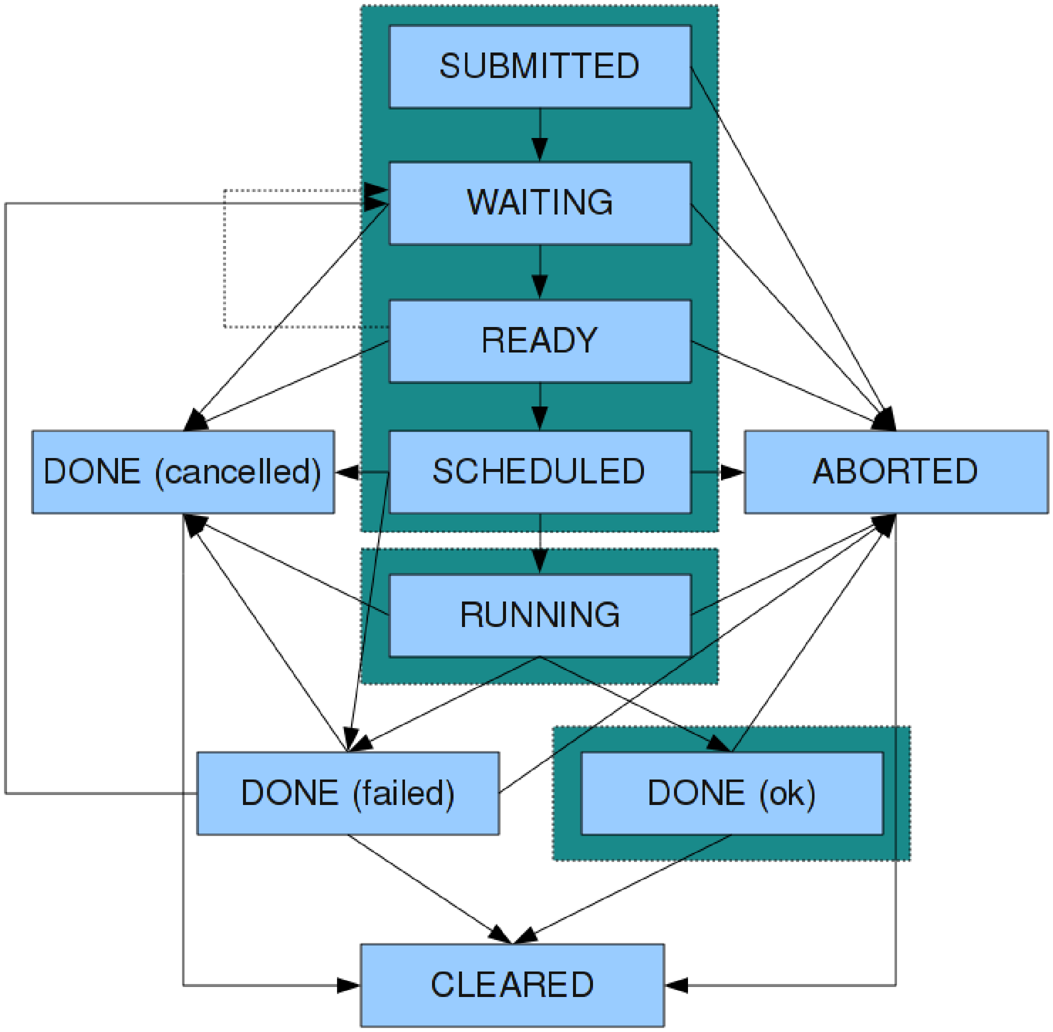
Input port/parameter	Description
`jobIdPort`	ID of a gLite grid job. It will be processed, and its subjobs returned.
`proxyPort`	Port to receive proxy data. This must be proxy signed by Virtual Organisation Management Service (VOMS) server. To obtain this proxy, please check vomsproxy.xml workflow.
`rasPort`	Port to receive address of Roaming Access Server (RAS), which is a layer between Kepler and gLite middleware. RAS address needs to be specified in form: PROTOCOL://ADDRESS:PORT where PROTOCOL is http or https eg. http://ras.example.com:8080
`retriesCount`	Parameter saying how many retries should there be. It may happen that subjobs are not yet ready. In such case, this actor will wait a specific amount of time and try again. However it may be that job is broken or subjobs could not be submitted at all. Then after some retries, actor will stop.

It has one output port:

Output port	Description
`children`	An array of children subjobs' ID (if job is not parametric, then 1-element array with its own ID).

2.4.4 RASi2gGetJobsStatuses

Once the job is submitted, you get its ID, which lets you retrieve its status. You can see gLite job status cycle on the following figure:



Every gLite job is in one of these states. To get current status of a job, one needs to poll the Logging&Bookkeeping service. This is done automatically by RAS, so that it always has recent data about managed jobs. To request job status is to use RASi2gGetJobsStatuses.

In order to increase performance and user experience, this actor checks statuses of a whole array of jobs. This is crucial in workflows with multiple jobs management. To use RASi2gGetJobsStatuses, please provide data to the following ports:

Input port/parameter	Description
`jobIdsPort`	IDs of gLite grid jobs to check. If you want to check parametric's subjob please provide its ID in form sub:PARENT@CHILD, where PARENT and CHILD are both gLite job IDs.
`proxyPort`	Port to receive proxy data. This must be proxy signed

Input port/parameter	Description
	by Virtual Organisation Management Service (VOMS) server. To obtain this proxy, please check vomsproxy.xml workflow.
`rasPort`	Port to receive address of Roaming Access Server (RAS), which is a layer between Kepler and gLite middleware. RAS address needs to be specified in form: PROTOCOL://ADDRESS:PORT where PROTOCOL is http or https eg. http://ras.example.com:8080
`retryCntPort`	Parameter saying how many retries should there be. It may happen that retrieving statuses of jobs fails. In such case, this actor will wait for some time try again. However it may be that status checking is impossible. Then after some retries, actor will stop.

Its output ports are:

Output port	Description
`jobIdsOutPort`	An array of IDs of gLite grid jobs. This is returned, because there is no guarantee that the order of IDs in input array will preserve. After sending input array to RAS, the order changes. However this actor has two output ports - this one and "statuses". They are synchronised, meaning that each i-th element from array "jobIdsOut" will be associated with i-th element from array "statuses".
`statuses`	An array of statuses of gLite grid jobs. Please see "jobIdsOut" documentation above to find out how these two ports are related.

2.4.5 RASi2gResubmit

When there are many jobs and one of them crashes, it may be very hard to automatically prepare job's specification and submit it once again. But such situations occur, and to deal with this problem, there is a special actor called RASi2gResubmit. It takes an existing job and clones it into a new one. Its main advantage is that it can resubmit not only as gLite does, but using deep-copy mechanism. This way, even subjobs can be resubmitted individually.

Before actual resubmission, actor checks if job's input files are accessible. It may happen that the primary job failed, because the storage element holding its inputs went down. In such situations, resubmission is pointless, as each next job will fail as well and RASi2gResubmit will find out if that's the case.

When input files are in place and accessible, then actor ensures that the new job have its resources changed, in order to avoid repetition of failures. It will have different WMS and go on to different CE. To use RASi2gResubmit, please provide data to its ports:

Input port/parameter	Description
`cePort`	Port to receive an address of Computing Element (CE) to use. CE stands for machine where job will be executed. It needs to be specified in form ADDRESS:PORT/QUEUE eg. ce.example.com/2119/jobmanager-pbs-short
`isTunneled`	Parameter stating if connection should be tunnelled or not. Tunnelled connections routes all the traffic through Roaming Access Server, thus demanding only open port to RAS. Untunnelled connections are faster because they work directly on remote storage. However, they demand open range of ports.
`jobIdPort`	Port to receive an ID of a job to be resubmitted.
`proxyPort`	Port to receive proxy data. This must be proxy signed by Virtual Organisation Management Service (VOMS) server. To obtain this proxy, please check vomsproxy.xml workflow.
`rasPort`	Port to receive address of Roaming Access Server (RAS), which is a layer between Kepler and gLite middleware. RAS address needs to be specified in form: PROTOCOL://ADDRESS:PORT where PROTOCOL is http or https eg. http://ras.example.com:8080
`retryCount`	Parameter saying how many retries should there be. It may happen that Workload Management System (WMS) refuses to process new job, either because of external problems (infrastructure failures, etc.) or internal ones (too many connections, etc.). In such case, this actor will choose different WMS and try again. However it may be that job submission is impossible whatever WMS is chosen. Then after some retries, actor will stop.
`sePort`	Port to receive an address of Storage Element (SE) to use. This parameter is optional - when left blank, actor will randomly choose any working SE. Also please note that upon failure, this actor retries with different SE, so please don't rely on setting this parameter - its value may change. SE needs to be specified in form ADDRESS eg. se.example.com
`wmsPort`	Port to receive an address of Workload Management System (WMS) to use. WMS is responsible for distribution and management of tasks across grid resources. It needs to be specified in form ADDRESS:PORT/SERVICE. Please note that most often, WMS will be presented with protocol name in

Input port/parameter	Description
	the beginning (eg. http://...), but this actor requires it to be dropped. WMS format readable by this actor does not allow protocol name eg. wms.example.com:7114/glite_wms_wmproxy_server

It has one output port:

Output port	Description
`newJobId`	New job's ID.

2.4.6 RASi2gGetJobOutputsLocation

When job has finished successfully and its output is ready to be downloaded, before using RASi2gDownloadFiles, we first need to retrieve remote locations of those files. For this purpose, we can use RASi2gGetJobOutputsLocation:

Input port/parameter	Description
`hostPort`	Port to receive address of Roaming Access Server (RAS), which is a layer between Kepler and gLite middleware. RAS address needs to be specified in form: PROTOCOL://ADDRESS:PORT where PROTOCOL is http or https eg. http://ras.example.com:8080
`jobIdPort`	Port to receive an ID of gLite grid job.
`proxyPort`	Port to receive proxy data. This must be proxy signed by Virtual Organisation Management Service (VOMS) server. To obtain this proxy, please check vomsproxy.xml workflow.

It has one output port:

Output port	Description
`outputFilesArray`	An array of LFNs specifying job's output files. LFN is a logical address of remote file in form of UNIX-like path eg. /grid/vo/some/file

2.5 Additional

2.5.1 RASi2gGloginTerminal

On gLite infrastructure, it is possible to submit interactive jobs. This process creates communication channels between WN (Worker Node) and client running Kepler. Potentially these channels can be used to any remote management of job execution. As a simple demonstration of its capabilities, there is an actor RASi2gGloginTerminal which allows to establish a secured terminal-like connection. To use it, please provide data to the actor's ports:

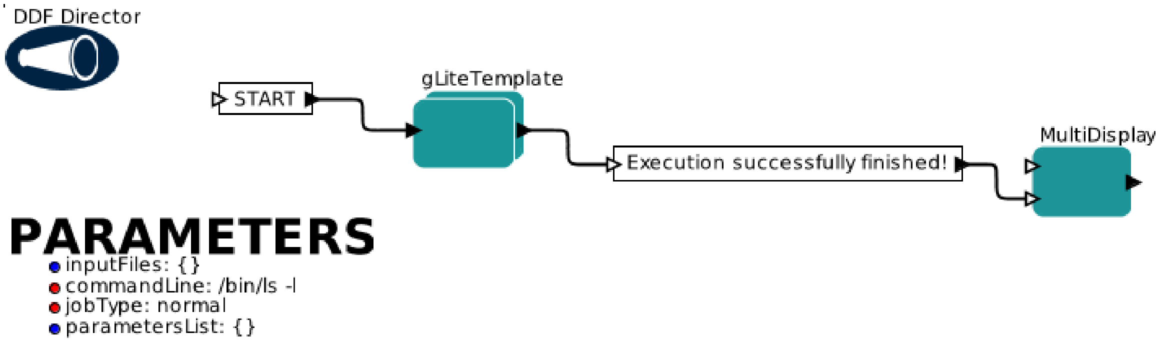
Input port/parameter	Description
`jobIdPort`	Port to receive an ID of gLite grid job. This job needs to be of "interactive" type.
`proxyPort`	Port to receive proxy data. This must be proxy signed by Virtual Organisation Management Service (VOMS) server. To obtain this proxy, please check vomsproxy.xml workflow.
`rasPort`	Port to receive address of Roaming Access Server (RAS), which is a layer between Kepler and gLite middleware. RAS address needs to be specified in form: PROTOCOL://ADDRESS:PORT where PROTOCOL is http or https eg. http://ras.example.com:8080

2.6 Example

Serpens gLite module contains an example workflow in `demos/` directory. It is a customizable **template** solution for job submission and management. Among its features are:

- input file upload,
- handling of single or parametric jobs (up to thousands of jobs),
- automatic job resubmission upon failure,
- output file download.

Note
Before executing grid jobs, you need to have a valid proxy created. In the `demos/` directory there is a **vomsproxy** workflow. Please run it once prior to any further actions in grid environment.



PARAMETERS

- inputFiles: {}
- commandLine: /bin/ls -l
- jobType: normal
- parametersList: {}

For tutorial and example usage, four parameters are presented:

1. `inputFiles`: an array of strings – paths to local files that should be uploaded (ie. made available to grid job).
2. `commandLine`: full command line to be executed (program with its parameters).

3. `jobType`: either string “normal” or “parametric.”
4. `parametersList`: if `jobType` is set to “parametric” then this is an array of strings where each entry will correspond to parameter value of one subjob. In gLite, parametric job definition may contain a special keyword `_PARAM_` which will be substituted by parameter value of each subjob. So you can prepare a generic job which processes an input named `_PARAM_`, submit it as parametric and then multiple jobs will run in parallel, each working on different data.

Using these four parameters, you can test gLite jobs in two ways:

1. Add some local paths in `inputFiles` and set `commandLine` to “`/bin/ls -l`”. Once the job is done and you have its output downloaded, you can check grid node directory listing to make sure the specified input files were uploaded.
2. Set `jobType` to “parametric”, `commandLine` to “`/bin/echo _PARAM_`” and `parametersList` to `{“a”, “b”, “c”}` to check that three jobs are submitted and upon finishing, their outputs are successfully downloaded.

We encourage you to check these two example situations now and observe workflow dynamics.

3 Glossary

JDL – Job Description Language, a standard format of job description.

Serpens – a Kepler 2.x suite containing modules to work with grid and HPC resources from the level of Kepler workflow. Currently available modules provide actors for gLite, UNICORE and Vine Toolkit.

Proxy – short for Proxy Certificate is a short-term (typically 12 hours) Digital Certificate designed to act remotely on behalf of a user.

Trust store – contains trusted CA certificates.

Key store – contains user's private key.

UNICORE registry – A UNICORE registry is an URL published for clients to use UNICORE services. Each UNICORE server should have at least one, publicly available registry as long as it is exposed in public. You should refer to your UNICORE instance manual, or contact your local UNICORE administrator in order to get registry address.

4 References

1. Serpens – http://scilla.man.poznan.pl/serpens
2. UNICORE – http://www.unicore.eu/
3. Kepler-2.0 – https://kepler-project.org/

5 Acknowledgments

This research has also received funding from the European Community's Seventh Framework Program (FP7/2007-2013) under grant agreement n°211804 (EUFORIA).