



Job Submission



The European DataGrid Project Team

<http://www.eu-datagrid.org>



Contents

- The EDG Workload Management System (WMS)
- Job Submission to the EDG Testbed
 - Job Preparation
 - Job Description Language (JDL)
 - Job Submission & Monitoring
 - A simple program example: the job lifecycle



The EDG WMS

- The user interacts with Grid via a **Workload Management System**
- The Goal of WMS is the **distributed scheduling and resource management in a Grid environment.**
- What does it allow Grid users to do?
 - To submit their jobs
 - To execute them
 - To get information about their status
 - To retrieve their output
- The WMS tries to optimize the usage of resources



WMS Components

- WMS is currently composed of the following parts:
 1. **User Interface (UI)** : access point for the user to the WMS
 2. **Resource Broker (RB)** : the broker of GRID resources, responsible to find the "best" resources where to submit jobs
 3. **Job Submission Service (JSS)** : provides a reliable submission system
 4. **Information Index (II)** : a specialized Globus GIIS (LDAP server) used by the Resource Broker as a filter to the information service (IS) to select resources
 5. **Logging and Bookkeeping services (LB)** : store Job Info available for users to query



Job Preparation: Let's think the way the Grid thinks!

- Information to be specified
 - Job characteristics
 - Requirements and Preferences of the computing system
 - Software dependencies
 - **Job Data requirements**
 - Specified using a Job Description Language (JDL)



Job Description Language (JDL) 1/5

- Based upon Condor's *CLASSified ADvertisement language (ClassAd)*
- ClassAd is a fully extensible language
- ClassAd is constructed with the classad construction operator []
It is a sequence of attributes separated by semi-colons. An attribute is a pair (key, value), where value can be a Boolean, an Integer, a list of strings, ...
<attribute> = <value>;

So, the JDL allows to define a set of attribute, the WMS takes into account when making its scheduling decision



Job Description Language (JDL) 2/5

- The supported attributes are grouped in two categories:
 - *Job (Attributes)*
Define the job itself
 - Resources
 - Taken into account by the RB for carrying out the matchmaking algorithm
 - *Computing Resource (Attributes)*
Used to build expressions of Requirements and/or Rank attributes by the user
Have to be prefixed with "other."
 - *Data and Storage resources (Attributes)*
Input data to process, SE where to store output data, protocols spoken by application when accessing SEs



Job Description Language (JDL): relevant attributes 3/5

- **Executable** (mandatory)
 - The command name
- **Arguments** (optional)
 - Job command line arguments
- **StdInput, StdOutput, StdErr** (optional)
 - Standard input/output/error of the job
- **Environment** (optional)
 - List of environment settings
- **InputSandbox** (optional)
 - List of files on the UI local disk needed by the job for running
 - The listed files are staged from the UI to the remote CE
- **OutputSandbox** (optional)
 - List of files, generated by the job, which have to be retrieved



Job Description Language (JDL): relevant attributes 4/5

➤ Requirements

- Job requirements on computing resources
- Specified using attributes of resources published in the Information Service
- If not specified, default value defined in UI configuration file is considered
 - ◆ Default: `other.Active` (the resource has to be able to accept jobs)

➤ Rank

- Expresses preference (how to rank resources that have already met the Requirements expression)
- Specified using attributes of resources published in the Information Service
- If not specified, default value defined in the UI configuration file is considered
 - ◆ Default: `-other.EstimatedTraversalTime` (the lowest estimated traversal time)



Job Description Language (JDL): "data" attributes 5/5

- **InputData** (optional)
 - Refers to data used as input by the job: these data are published in the Replica Catalog and stored in the SEs)
 - PFNs and/or LFNs
- **ReplicaCatalog** (mandatory if InputData has been specified with at least one LFN)
 - The Replica Catalog Identifier
- **DataAccessProtocol** (mandatory if InputData has been specified)
 - The protocol or the list of protocols which the application is able to speak with for accessing *InputData* on a given SE
- **OutputSE** (optional)
 - The Uniform Resource Identifier of the output SE
 - RB uses it to choose a CE that is compatible with the job and is close to SE



Example JDL File

```
Executable = "gridTest";
StdError = "stderr.log";
StdOutput = "stdout.log";
InputSandbox = {"home/joda/test/gridTest"};
OutputSandbox = {"stderr.log", "stdout.log"};
InputData = "LF:testbed0-00019";
ReplicaCatalog = "ldap://sunlab2g.cnaf.infn.it:2010/ \
                 lc=test, rc=WP2 INFN Test, dc=infn, \
                 dc=it";
DataAccessProtocol = "gridftp";
Requirements = other.Architecture=="INTEL" && \
               other.OpSys=="LINUX" && other.FreeCpus >=4;
Rank = "other.MaxCpuTime";
```



Job Submission

➤ `dg-job-submit [-r <res_id>] [-n <user e-mail address>] [-c <config file>] [-o <output file>] <job.jdl>`

`-r` the job is submitted by the RB directly to the computing element identified by `<res_id>`

`-n` an e-mail message containing basic information regarding the job (status and identification) is sent to the specified `<e-mail address>` when the job enters one of the following status:

DONE or ABORTED

READY

RUNNING

`-c` the configuration file `<config file>` is pointed by the UI instead of the standard configuration file

`-o` the generated `dg_jobId` is written in the `<output file>`

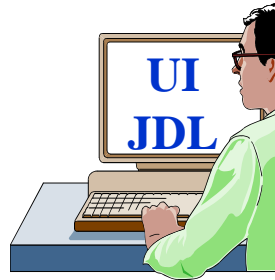
Useful for other commands, e.g.:

`dg-job-status -i <input file>` (or `dg_jobId`)

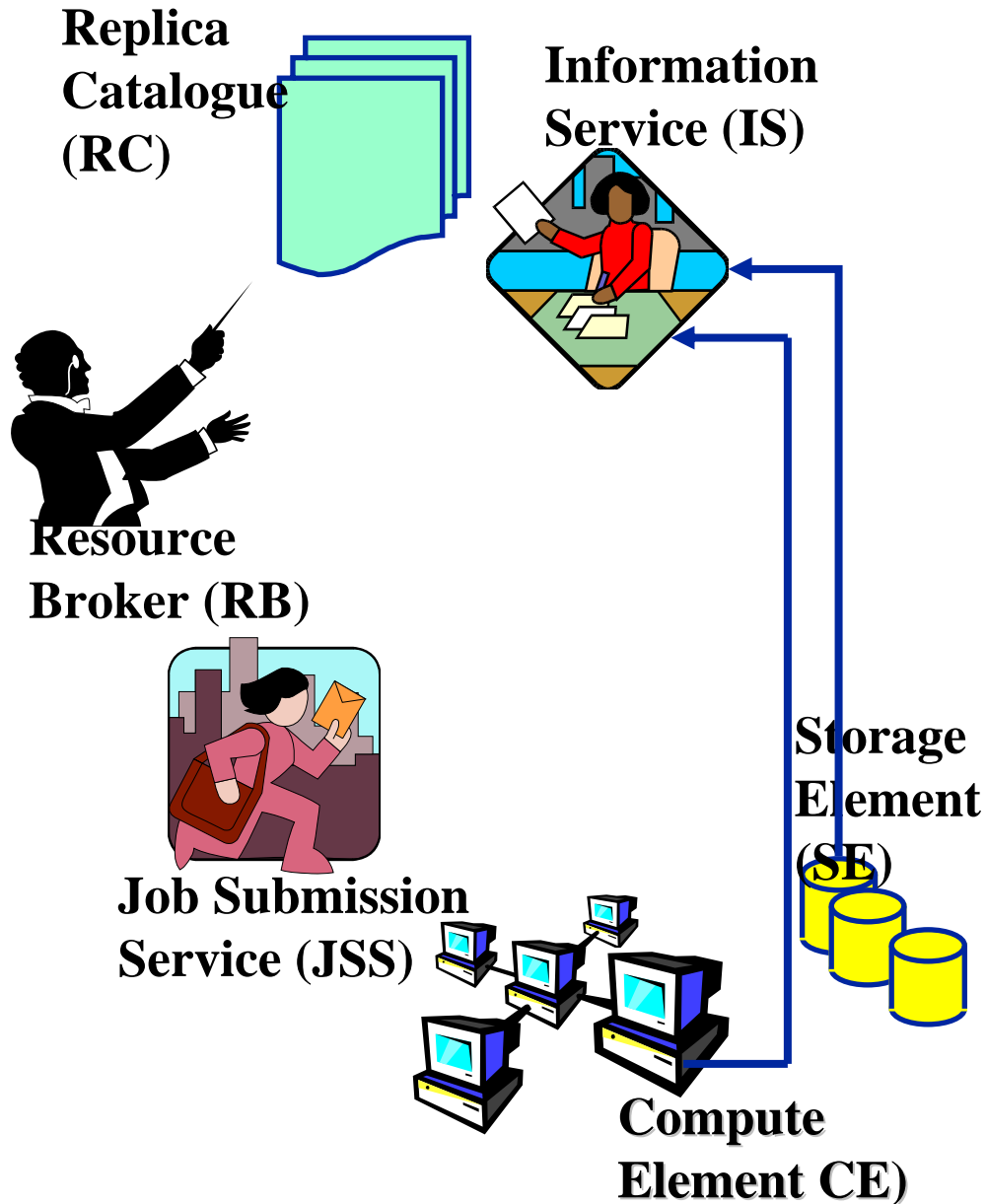
`-i` the status information about `dg_jobId` contained in the `<input file>` are displayed



Job Submission Scenario



Logging &
Bookkeeping
(LB)

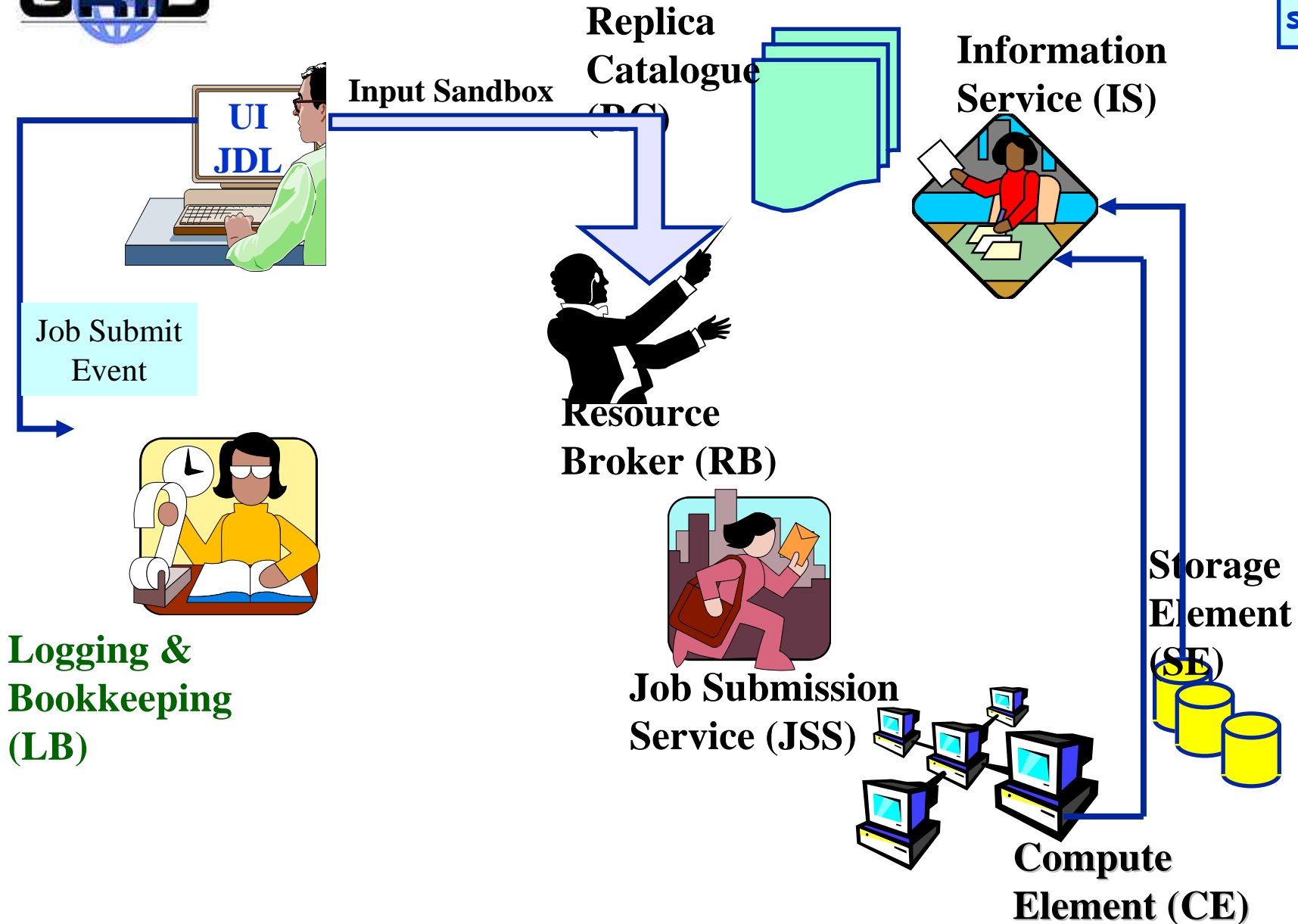




A Job Submission Example

Job Status

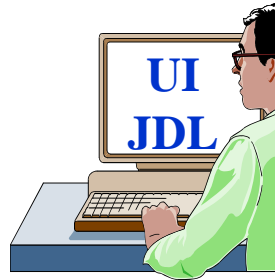
submitted



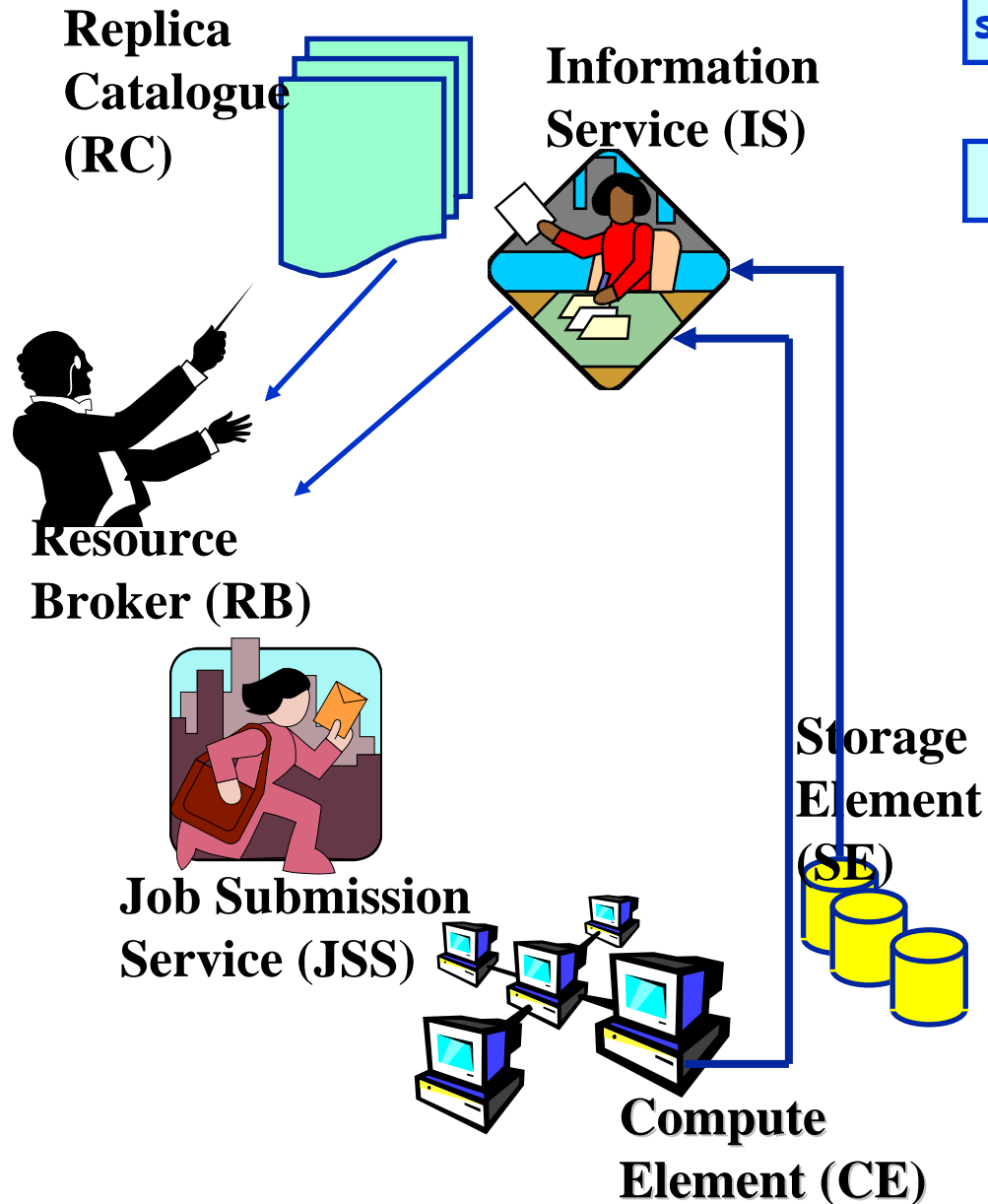


A Job Submission Example

Job Status



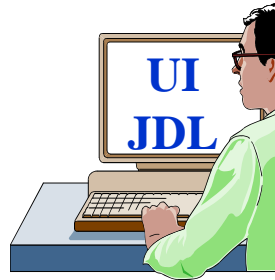
Logging &
Bookkeeping
(LB)



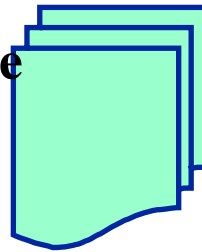


A Job Submission Example

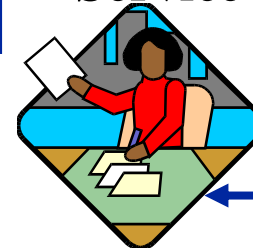
Job Status



Replica Catalogue (RC)



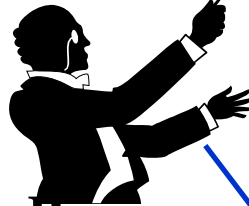
Information Service (IS)



submitted

waiting

ready



Resource Broker (RB)

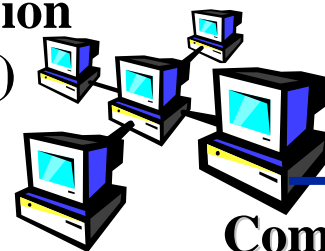
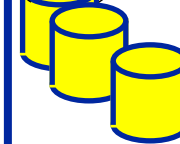


Logging & Bookkeeping (LB)



Job Submission Service (JSS)

Storage Element (SE)

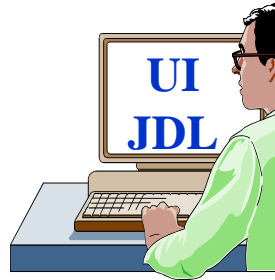


Compute Element (CE)

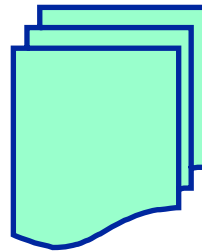


A Job Submission Example

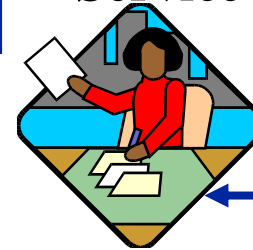
Job Status



Replica Catalogue (RC)



Information Service (IS)



Resource Broker (RB)



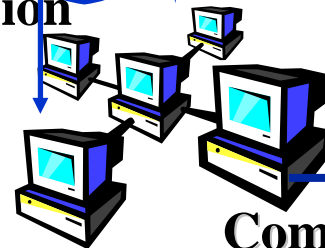
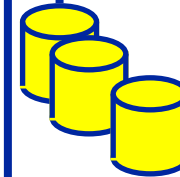
Logging & Bookkeeping (LB)



Job Submission Service (JSS)

BrokerInfo

Storage Element (SE)

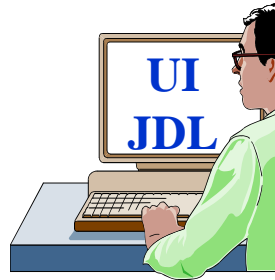


Compute Element (CE)



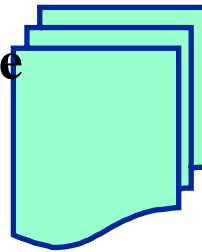


A Job Submission Example

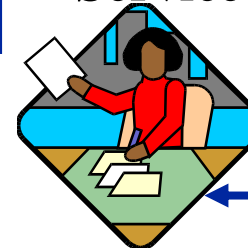


Logging &
Bookkeeping
(LB)

Replica
Catalogue
(RC)

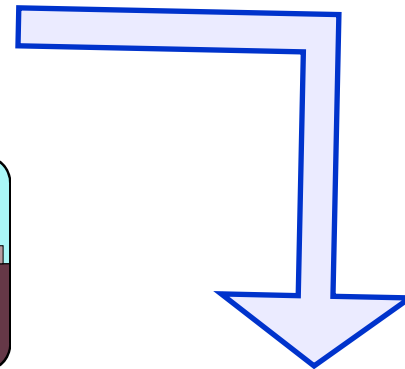


Information
Service (IS)

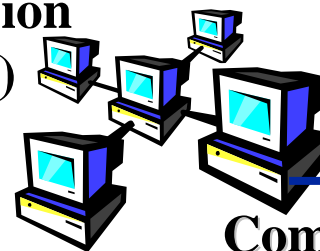


Resource
Broker (RB)

Input Sandbox

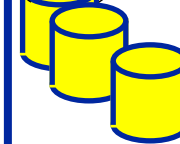


Job Submission
Service (JSS)

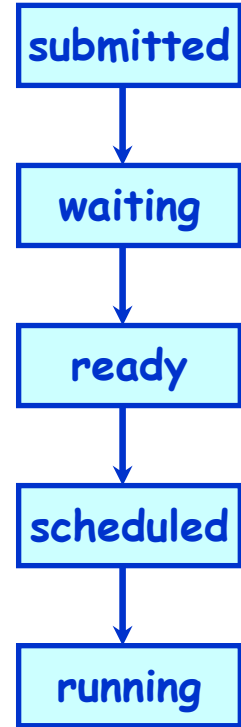


Compute
Element (CE)

Storage
Element
(SE)

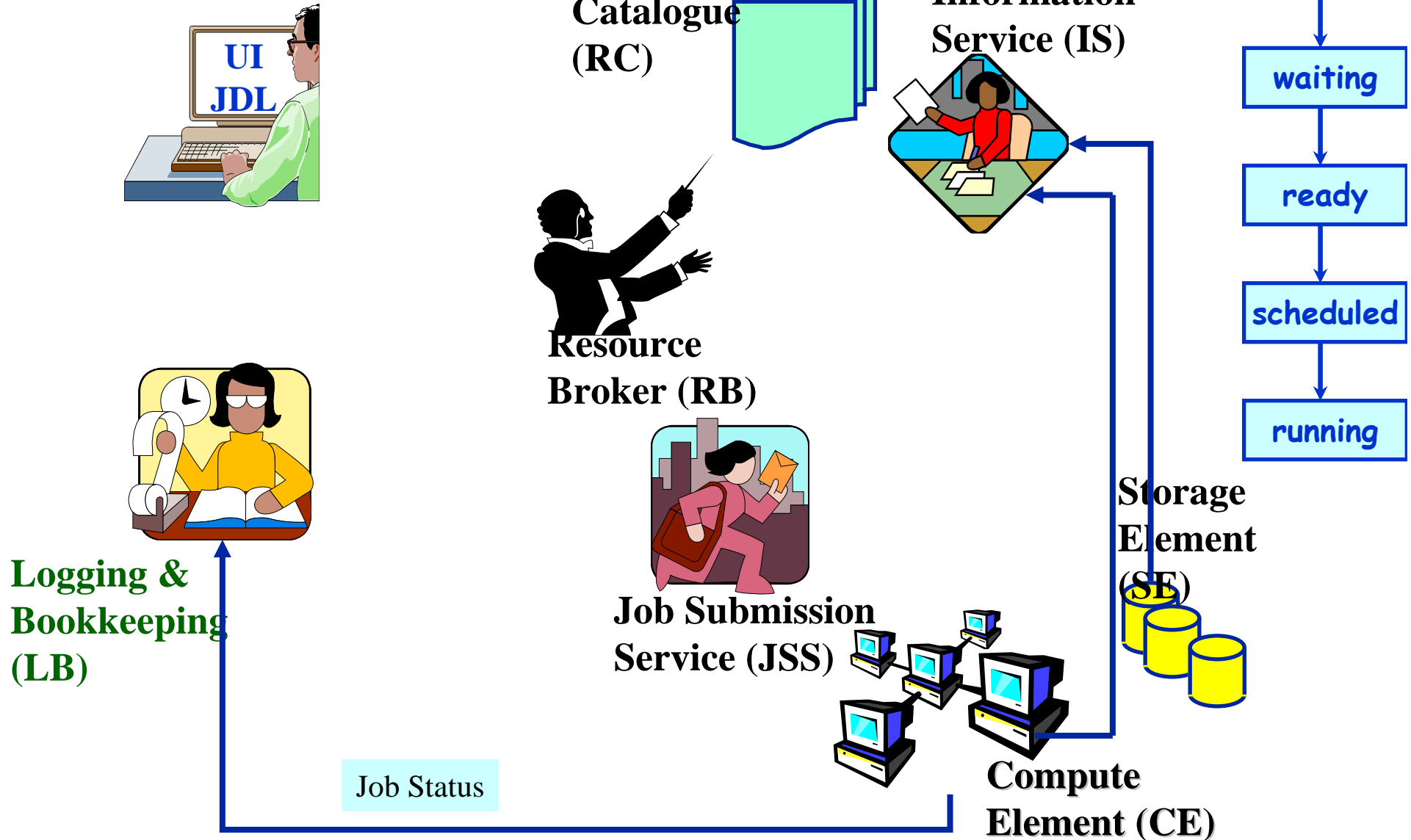


Job Status





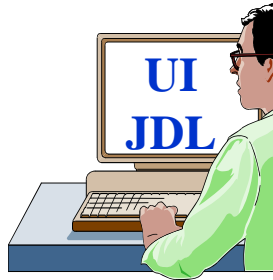
A Job Submission Example



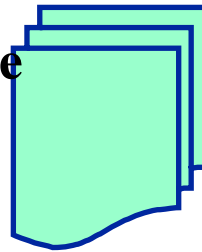


A Job Submission Example

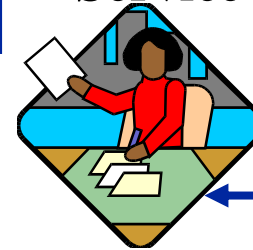
Job Status



Replica Catalogue



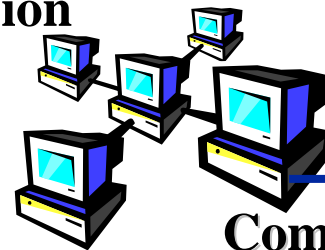
Information Service



Resource Broker

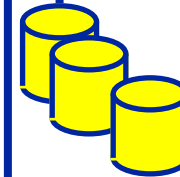


Job Submission Service



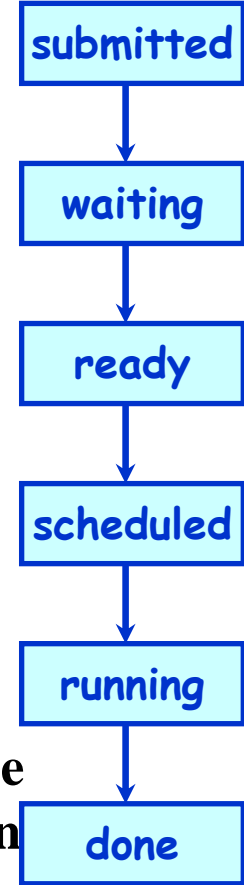
Compute Element

Storage Element



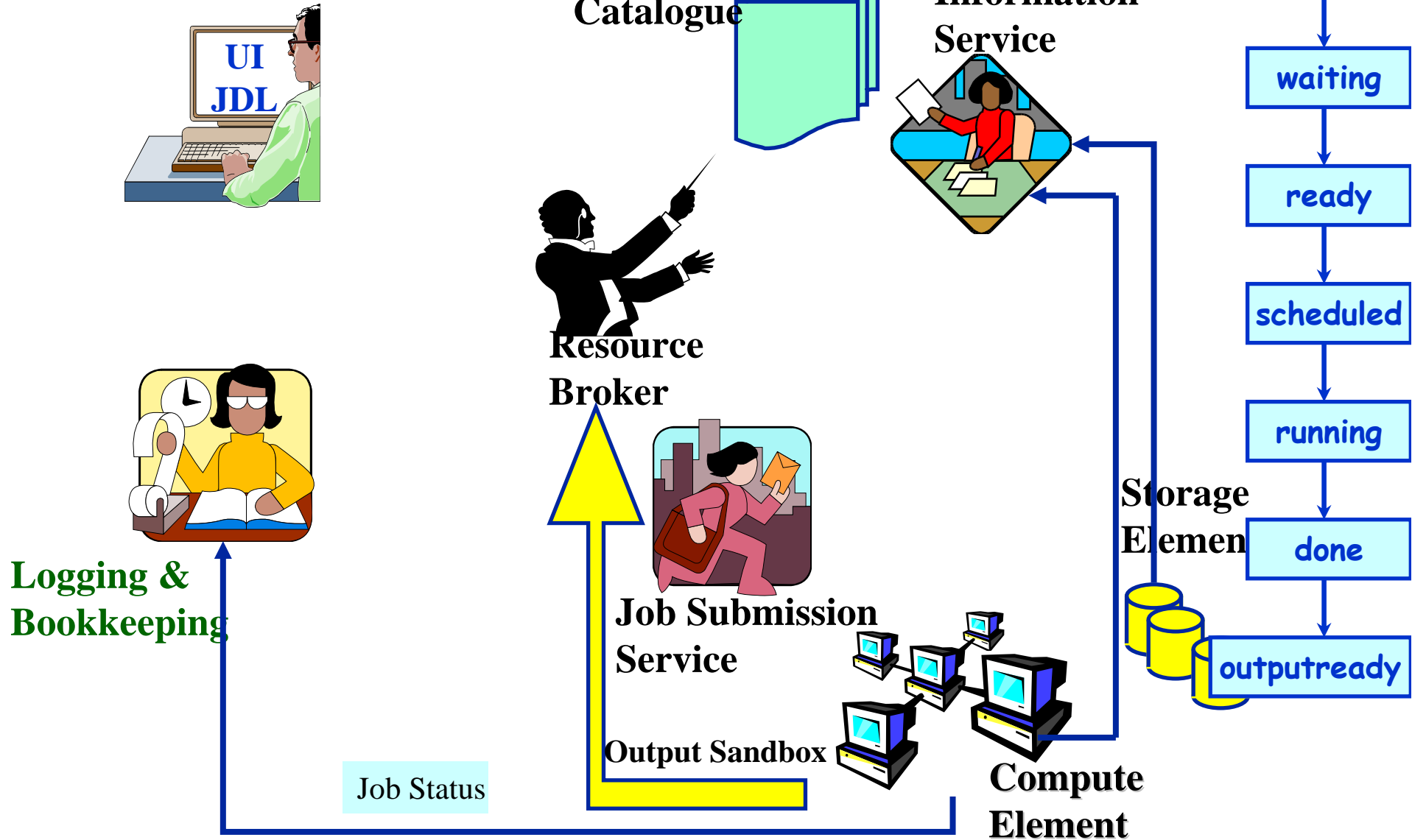
Logging & Bookkeeping

Job Status



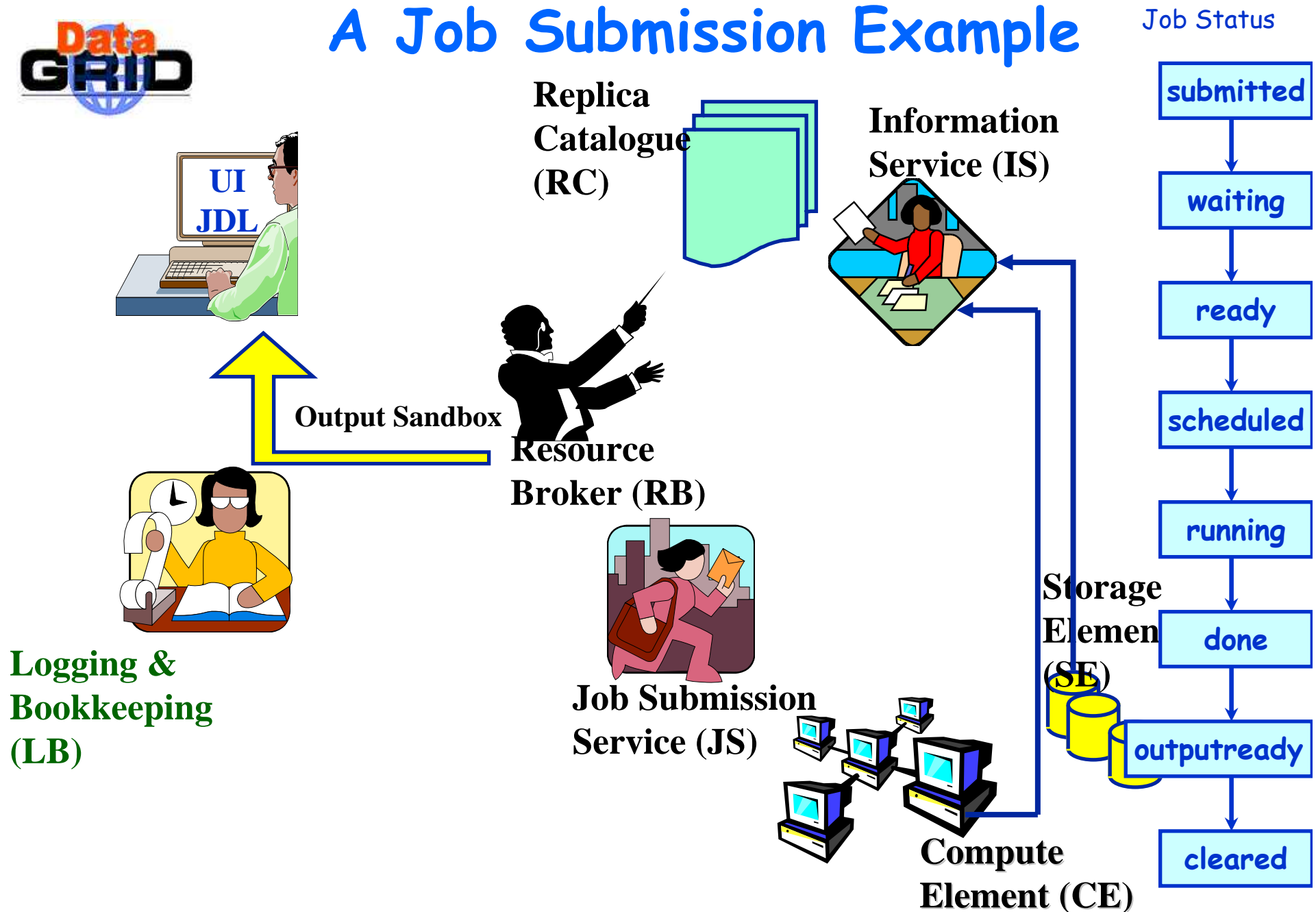


A Job Submission Example



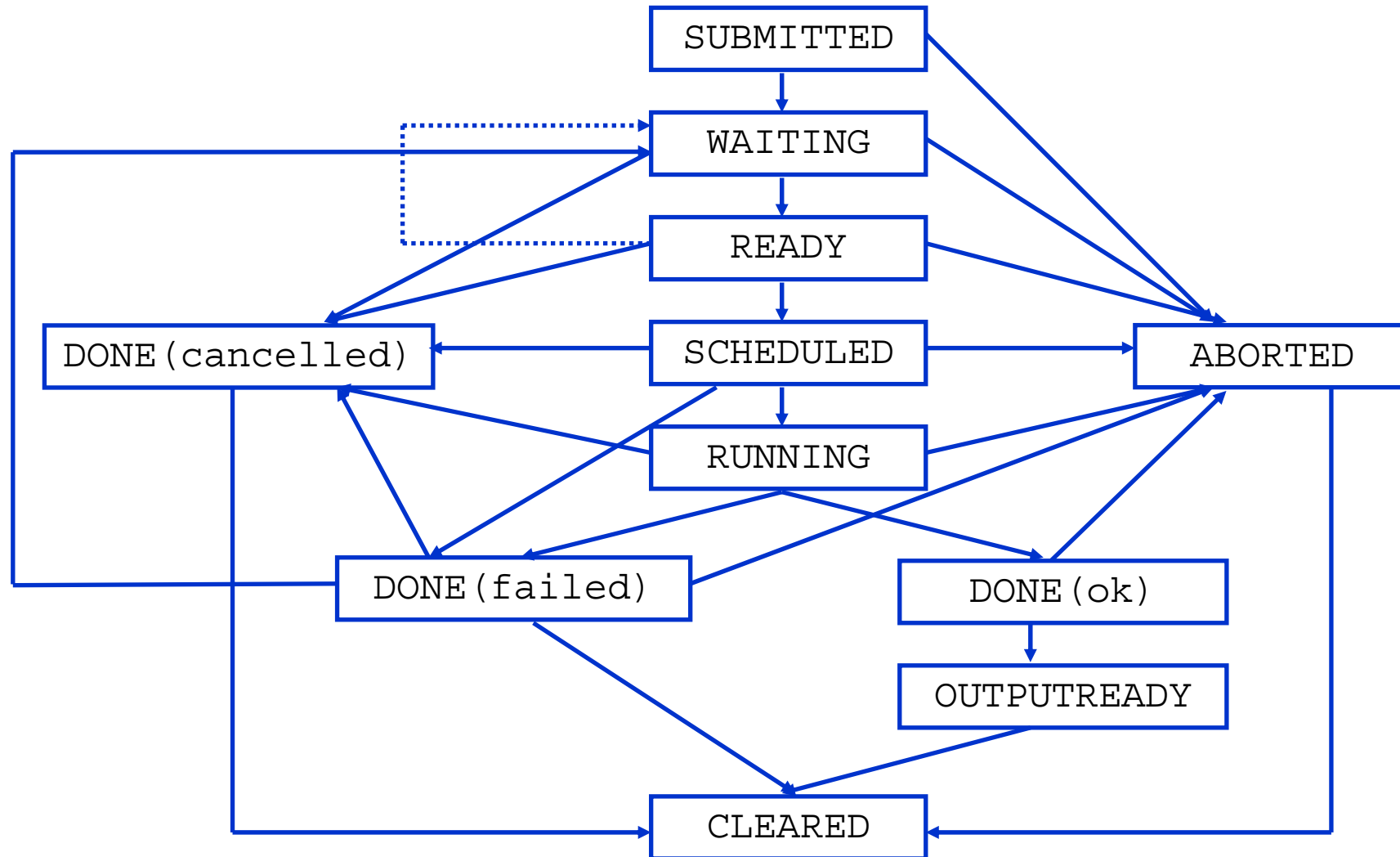


A Job Submission Example





Possible Job States





Other WMS UI Commands

- **dg-job-list-match**
 - Lists resources matching a job description
 - Performs the matchmaking without submitting the job
- **dg-job-cancel**
 - Cancels a given job
- **dg-job-status**
 - Displays the status of the job
- **dg-job-get-output**
 - Returns the job-output (the OutputSandbox files) to the user
- **dg-job-get-logging-info**
 - Displays logging information about submitted jobs (all the events "pushed" by the various components of the WMS)
 - Very useful for debug purposes
- **dg-job-id-info**
 - A utility for the user to display job info in a formatted style

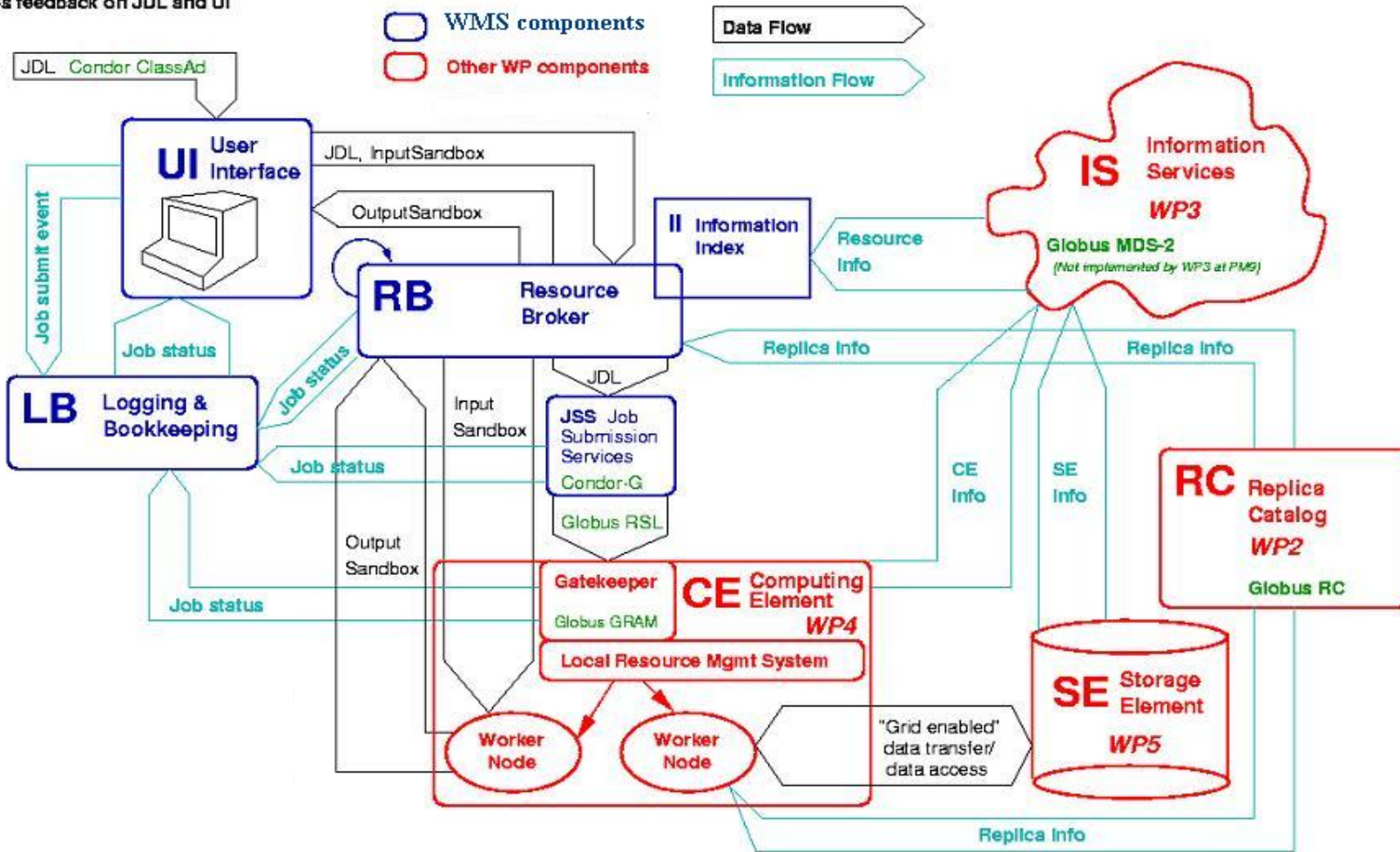


UI configuration file

- Can be set if user is not happy with default one
- Most relevant attributes:
 - RB(s)
 - When submitting a job, the first specified RB is tried, if the operation fails the second one is considered, etc.
 - LBserver(s)
 - The LB to be used for a job is chosen by the RB
 - So when a `dg-job-status <dg-jobid>` is issued, the LB to contact is specified in the `dg-jobid`
 - This list specifies the LB(s) that must be contacted when issuing a `dg-job-status -all / dg-job-get-logging-info -all` (to have information for all the jobs belonging to that user)
 - Default JDL Requirements
 - `other.active`
 - Default JDL Rank
 - `- other.EstimatedTraversalTime`

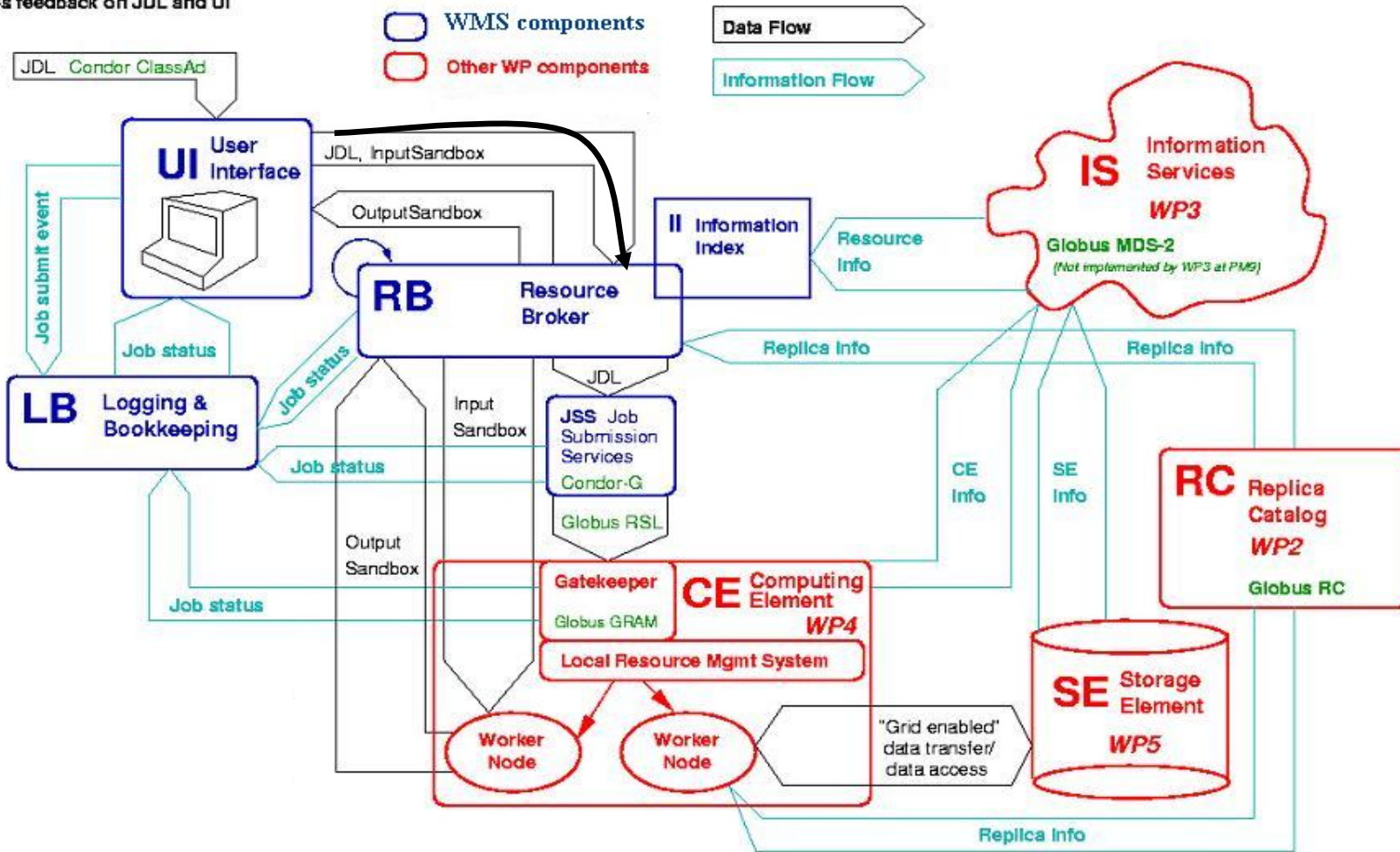
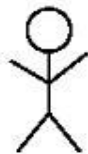
End User WP8,9,10

- Specifies job using JDL
- Submits job using UI
- Controls and monitors job(s)
- Provides feedback on JDL and UI



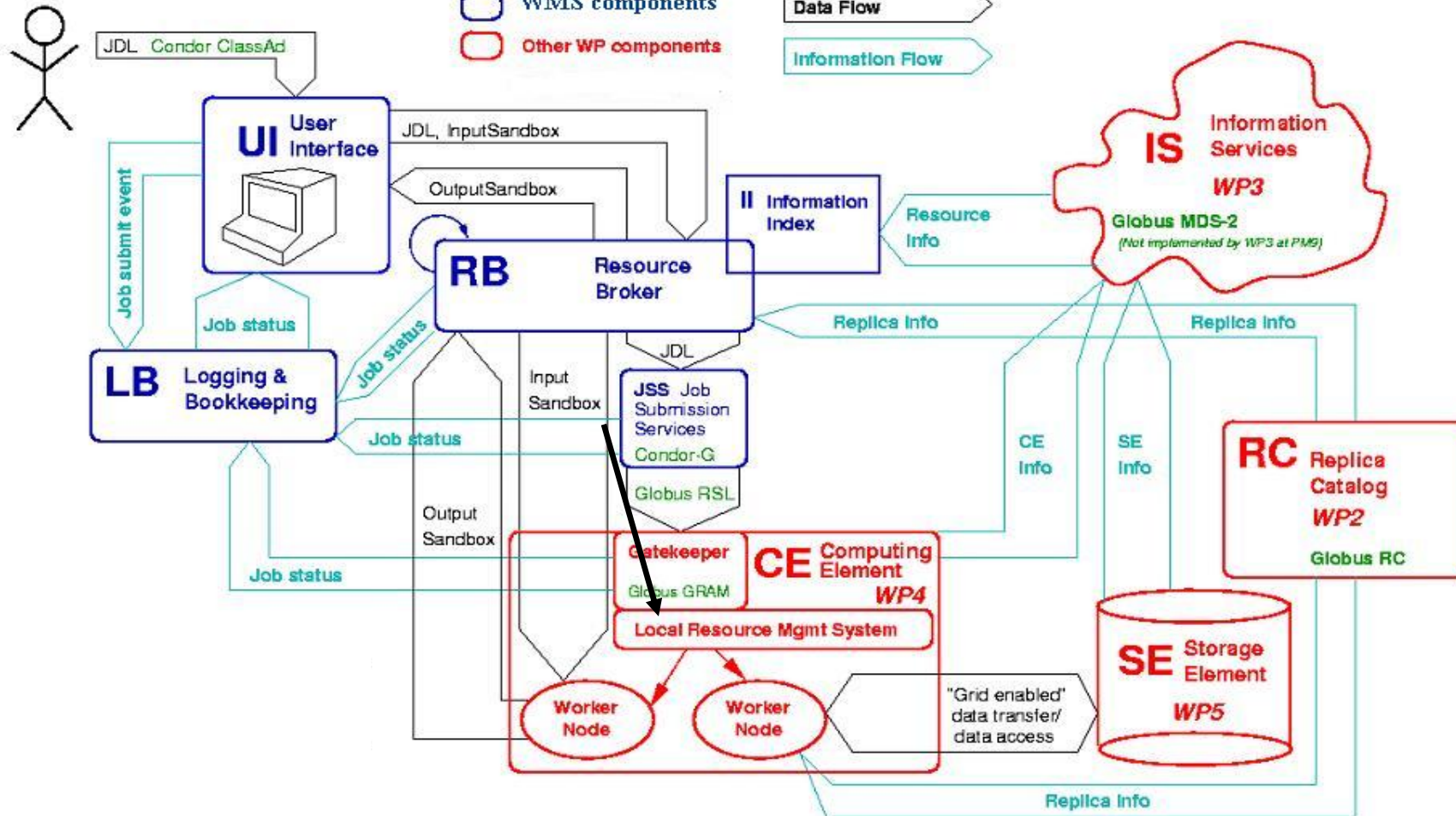
End User WP8,9,10

- Specifies job using JDL
- Submits job using UI
- Controls and monitors job(s)
- Provides feedback on JDL and UI



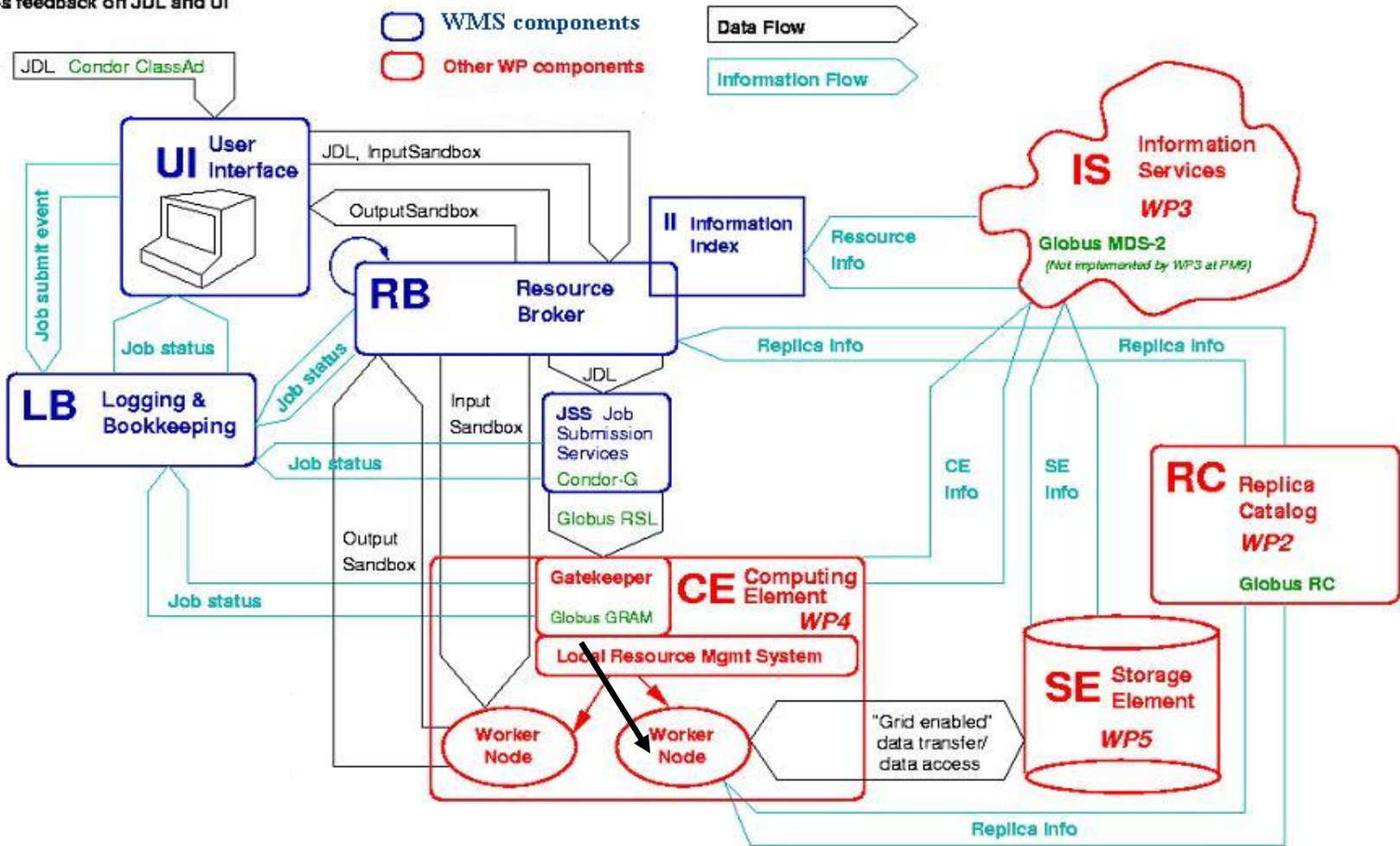
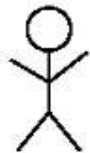
End User WP8,9,10

- Specifies job using JDL
- Submits job using UI
- Controls and monitors job(s)
- Provides feedback on JDL and UI



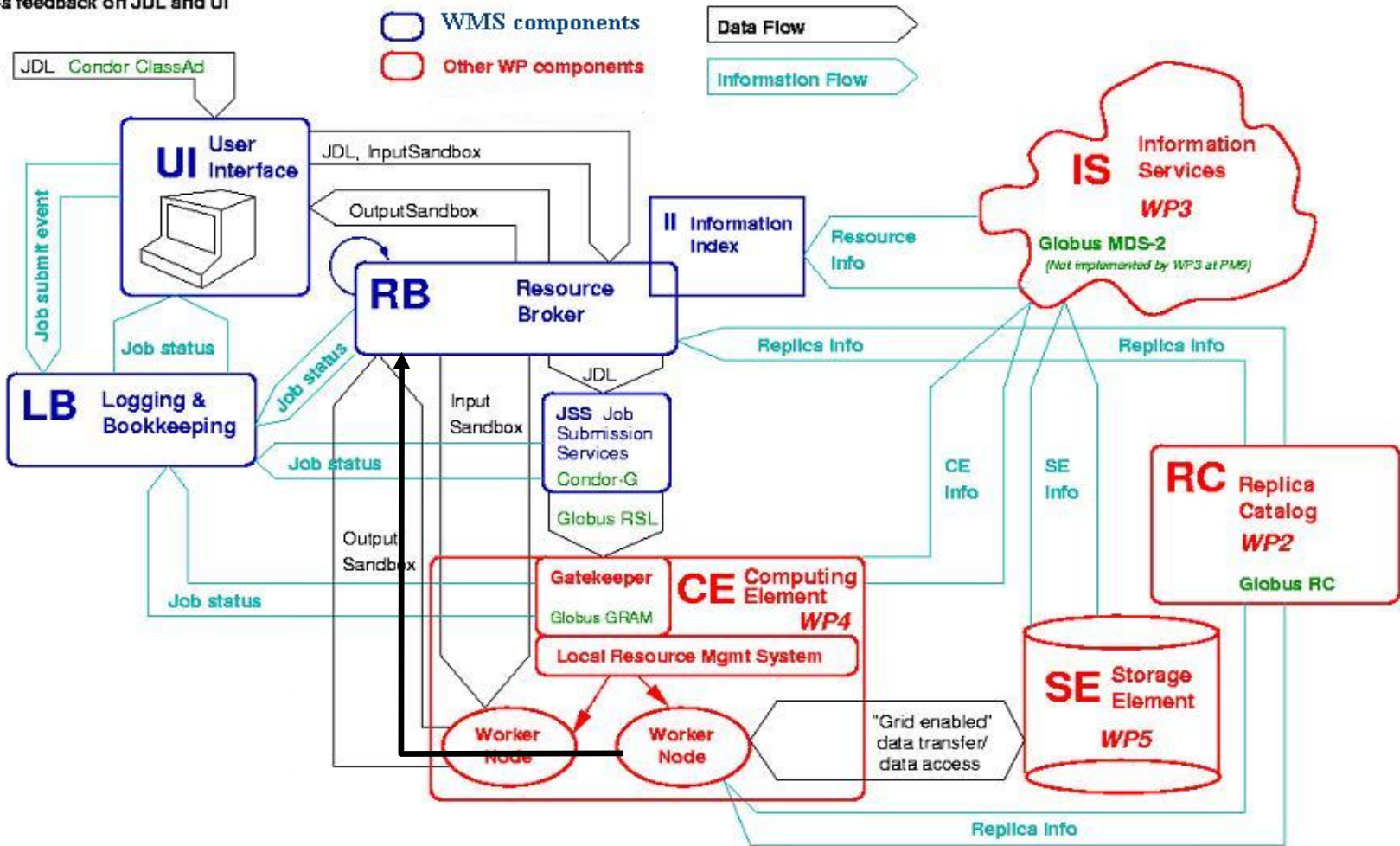
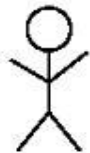
End User WP8,9,10

- Specifies job using JDL
- Submits job using UI
- Controls and monitors job(s)
- Provides feedback on JDL and UI



End User WP8,9,10

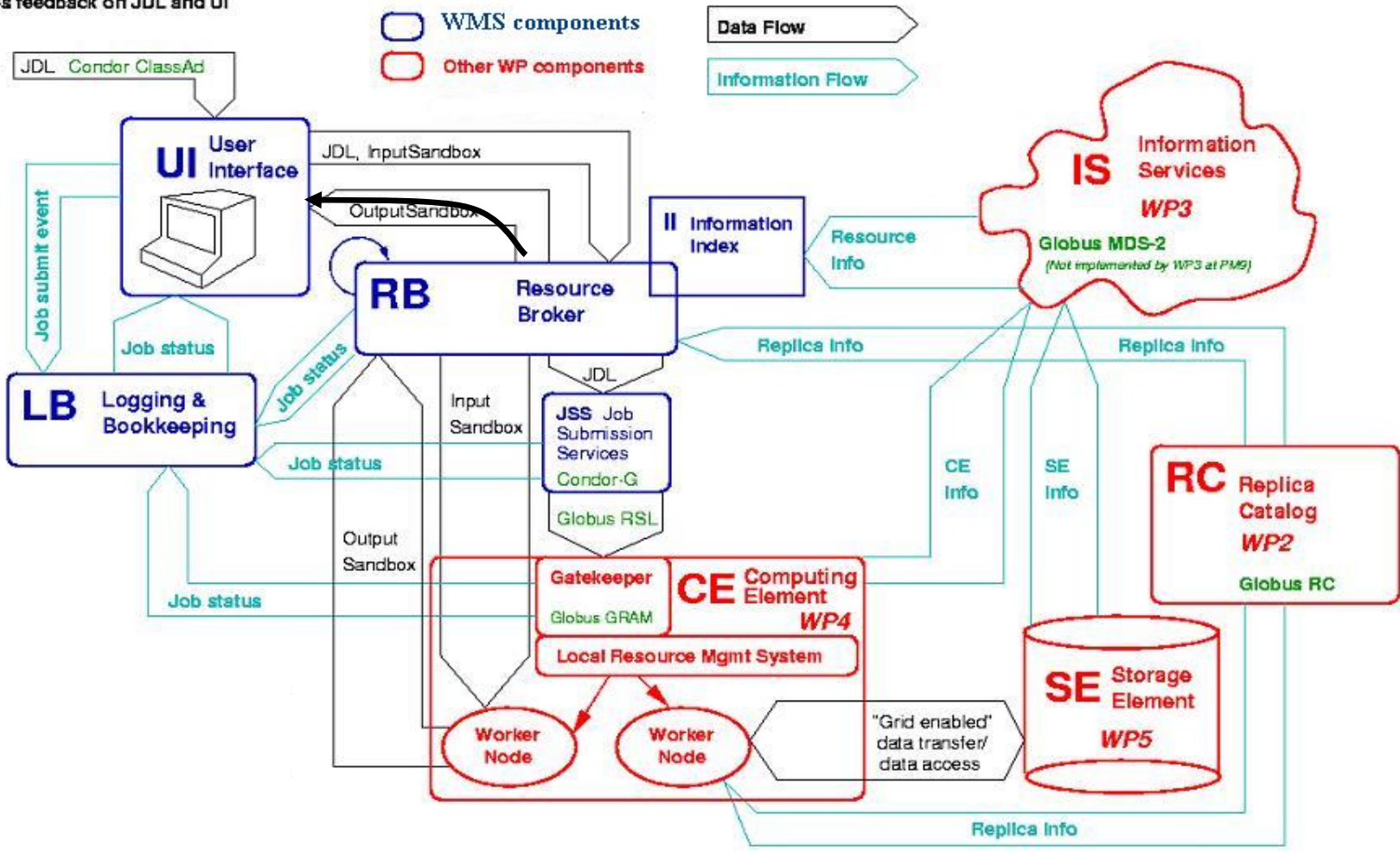
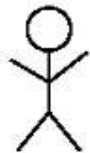
- Specifies job using JDL
- Submits job using UI
- Controls and monitors job(s)
- Provides feedback on JDL and UI



Integration

End User WP8,9,10

- Specifies job using JDL
- Submits job using UI
- Controls and monitors job(s)
- Provides feedback on JDL and UI



Integration



WMS Match Making 1/4

- The RB is the core component of WMS.
- It has to find the best suitable computing resource (CE) where the job will be executed
- It interacts with Data Management service and Information Service
 - They supply RB with all the information required for the resolution of the matches
- The CE chosen by RB has to match the job requirements (e.g. runtime environment, data access requirements, and so on)
- If 2 or more CEs satisfy all the requirements, the one with the best Rank is chosen



WMS Match Making 2/4

- The RB has to deal with three possible scenarios.

1. Scenario : Direct Job Submission

- ◆ Job is scheduled on a given CE (specified in the dg-job-submit command via -r option)
- ◆ RB doesn't perform any matchmaking algorithm



WMS Match Making 3/4

2. Scenario : Job Submission without data-access Requirements

- ◆ Neither CE nor input data are specified.
- ◆ RB starts the matchmaking algorithm, which consists of two phases:
 - Requirements check (RB contacts the IS to check which CEs satisfy all the requirements)
 - If more than one CE satisfies the job requirements, the CE with the best rank is chosen by the RB



WMS Match Making 4/4

3. Scenario : Job Submission with also data-access Requirements

- ◆ CE is not specified in the JDL
- ◆ RB interacts with Data Management service to find out the most suitable CE taking into account also the SEs where both input data sets are physically stored and output data sets should be staged on completion of job execution
- ◆ RB strategy consists of submitting jobs close to data
- ◆ The main two phases of the match making algorithm remain unchanged:
 - . Requirements check
 - . Rank computation
- ◆ What changes with respect to the second scenario?

Now, the RB executes the two phases for each class of CEs that satisfy the data-access requirements (i.e. which are close to data)



Example of Job Submission Sequence

- User logs in on the UI
- User issues a *grid-proxy-init* and enters his certificate's password, getting a valid Globus proxy
- User sets up his or her JDL file
- Example of Hello World JDL file :

Executable = "/bin/echo";

Arguments = "Hello World";

StdOutput = "Message.txt";

StdError = "stderr.log";

OutputSandbox = {"Message.txt", "stderr.log"};



Example of Job Submission Sequence Cont'd

- User issues a: *dg-job-submit HelloWorld.jdl*
and gets back from the system a unique Job Identifier (JobId)
- User issues a: *dg-job-status JobId*
to get logging information about the current status of his Job
- When the "OutputReady" status is reached, the user can issue a
dg-job-get-output JobId
and the system returns the name of the temporary directory where the job output
can be found on the UI machine.



Job Submission Example

```
[reale@testbed002 EliJDL]$ dg-job-submit HelloWorld.jdl
```

```
Connecting to host lxshare0381.cern.ch, port 7771
```

```
Logging to host lxshare0381.cern.ch, port 15830
```

```
*****
```

JOB SUBMIT OUTCOME

The job has been successfully submitted to the Resource Broker.

Use dg-job-status command to check job current status. Your job identifier (dg_jobId) is:

```
-  
https://lxshare0381.cern.ch:7846/137.138.181.214/12183940774010?lxshare0381.cern.ch:7771
```

```
*****
```

JobId



Job Submission Example Cont'd

```
[reale@testbed002 EliJDL]$ dg-job-status  
https://lxshare0381.cern.ch:7846/137.138.181.214/12183940774010?lxshare0381.cern.ch:7771
```

Retrieving Information from LB server <https://lxshare0381.cern.ch:7846>

Please wait: this operation could take some seconds.

BOOKKEEPING INFORMATION:

Printing status info for the Job :

<https://lxshare0381.cern.ch:7846/137.138.181.214/12183940774010?lxshare0381.cern.ch:7771>

dg_JobId =
<https://lxshare0381.cern.ch:7846/137.138.181.214/12183940774010?lxshare0381.cern.ch:7771>

Status = OutputReady

Last Update Time (UTC) = Wed Aug 21 12:19:39 2002

Job Destination = testbed008.cnaf.infn.it:2119/jobmanager-pbs-short

Status Reason = terminated

Job Owner = /C=IT/O=INFN/OU=Personal Certificate/L=CNAF/CN=Mario
Reale/Email=Mario.Reale@cnaf.infn.it

Status Enter Time (UTC) = Wed Aug 21 12:19:39 2002



Job Submission Example Cont'd

```
[reale@testbed002 EliJDL]$ dg-job-get-output --dir result  
https://lxshare0381.cern.ch:7846/137.138.181.214/12183940774010?lxshare0381.cern.ch:7771
```

```
*****
```

JOB GET OUTPUT OUTCOME

Output sandbox files for the job:

- <https://lxshare0381.cern.ch:7846/137.138.181.214/12183940774010?lxshare0381.cern.ch:7771>

have been successfully retrieved and stored in the directory:

</shift/lxshare072d/data01/UIhome/reale/EliJDL/result/12183940774010>

```
*****
```

```
[reale@testbed002 EliJDL]$ more result/12183940774010/Message.txt
```

Hello World

```
[reale@testbed002 EliJDL]$ more result/12183940774010/stderr.log
```




Common Error Messages 1/2

- The UI commands accept some arguments in input. If the user makes a mistake via command line, the following messages can appear:

Argument * is not allowed (the argument is not known)

Argument * must be specified at the end of the command (both the jobId and JDL file name must be put at the end of the command line)

Argument * is missing for the "--output" option (the user forgot to add the parameter, required by the argument)

Argument "-all" cannot be specified with argument "--input" (some arguments are OR-exclusive)

CEId format is: <full hostname>;<port number>/jobmanager-<service>. The provided CEID: "http://lx01.absolute.com:10854/jobmanager" has a wrong format. (the user has mis-spelled the CE identifier after -resource)

- During the calling of the RB API, the following can happen:

Resource Broker "grid013g.cnaf.infn.it:7771" not available (can't open a connection with the RB specified in the UI configuration file)

Unable to get LB address from RB "grid013g.cnaf.infn.it" (the function get_lb_contact returned an error)



Common Error Message 2/2

- While the UI commands are checking the JDL file, the following errors may occur:
 - Mandatory Attribute default error in the configuration file
"/opt/edg/etc/UI_ConfigENV.cfg" (there aren't any default values)
 - Mandatory Attribute missing in JDL file "Executable" (Executable is one of the mandatory attributes)
 - Multiple "InputSandbox" attribute found in JDL file (InputSandbox attribute is repeated twice)
 - Wrong function call for list attribute *. Function usage is:
"Member/IsMember(List, Value)" (e.g. in the requirements attribute the function Member/IsMember is used with a wrong syntax)
- Proxy (this refers to the security grid proxy and not to a proxy machine)
 - If the user specifies a duration for the proxy that he wants to provide, using the option -h of dg-job-submit, a possible message is
Proxy certificate will expire in less than X hours. Creating a new X-hours-duration certificate (this to make sure that at least the required proxy validity is granted)



WMS Proxy Renewal

- Why?
 - To avoid job failure because it outlived the validity of the initial proxy
- WMS support automatic proxy renewal mechanism as long as the user credentials are handled by a proxy server.
 1. Create a proxy using
grid-proxy-init
 2. Register this proxy with the MyProxy server using
myproxy-init -s <server> [-t <cred> -c <proxy>]
server is the server address (e.g. lxshare0375.cern.ch)
cred is the number of hours the proxy should be valid on the server
proxy is the number of hours renewed proxies should be valid
 3. Short term proxies can then be used to start jobs using
grid-proxy-init -hours <hours> command
 4. The Proxy is automatic renewed by WMS without user intervention for all the job life



Further Information

- The EDG User's Guide

<http://marianne.in2p3.fr>

- EDG WP1 Web site

<http://www.infn.it/workload-grid>

In particular WMS User & Admin Guide and JDL docs

- ClassAd

<https://www.cs.wisc.edu/condor/classad>