



## WG-5 First Version of Administration of Simple Grid Jobs, Software Components, and Documentation<sup>1</sup>

Deliverable	5.2
Main Authors	Rainer Spurzem, Thomas Brüsemeister, Jürgen Steinacker, Hans-Martin Adorf
Further Contributions	Members of WG-5 (see History of Document) and the other members of the <i>Task Force Scheduler Broker</i> Michael Braun, Frank Breitling, Christian Grimme (C3), Thomas Röblitz, Robert Tucker
Date	June 15, 2007
Document Version	1.0.3
Current Version	1.0.3
Previous Versions	1.0.0, 1.0.1, 1.0.2

### A: Status of this Document

This second working draft is based on the RS draft compiled from the internal lenya pages <http://mintaka.aip.de:8080/lenya/intranet/live/workpackages/wg5/TaskForceSchedulerBroker.html>. According to the project plan (M18), the deliverable D5.2 is due on February, 28th 2007.

### B: Reference to project plan

---

<sup>1</sup>This work is part of the AstroGrid-D project and D-Grid. The project is funded by the German Federal Ministry of Education and Research (BMBF).

This deliverable refers to the task TA V-2 *Entwicklung des Grid-Job Managements* and milestone M18 in the project plan.

## **C: Abstract**

This document summarizes the requirements for a resource broker (RB) software component to handle simple grid jobs which are submitted to AstroGrid-D for execution.

The criteria for a desired software solution for the AstroGrid-D project are specified, and a structure of a first prototype of the job metadata is given. It provides the assessment of different existing software which could be used for this task. The packages GridWay, Condor-G, CSF, GridBus/GSB, Nimrod/G, Ascalon, GRMS, WMS, Moab/Maui, as well as the option to develop an AstroGrid-D-specific RB software within the project were evaluated. As a result of the evaluation, GridWay was chosen as the software solution that currently appears to best meet the needs and specifications of AstroGrid-D. A first description is given how GridWay can be used as a RB within the AstroGrid-D project, and especially how to submit jobs from the Planck Process Coordinator (ProC) workflow engine.

**D: Change History**

<b>Version</b>	<b>Date</b>	<b>Name</b>	<b>Brief summary</b>
1.0.0	until 20 Dec 2006	Jürgen Steinacker	Collection of data in a lenya page
1.0.1	23 Dec 2006	Rainer Spurzem	Working draft creation
1.0.1	– 07 Mar 2007	Thomas Brüsemeister, Jürgen Steinacker	Editing
1.0.2	31 Mar 2007	Hans-Martin Adorf	numerous minor changes
1.0.3	1 Apr 2007	Rainer Spurzem	few minor changes

## Contents

<b>1</b>	<b>Introduction</b>	<b>6</b>
1.1	Description of the requirements of a RB software component for use within AstroGrid-D . . . . .	6
1.1.1	General requirements of the RB software . . . . .	7
1.1.2	Metadata prototype . . . . .	7
1.1.3	Design requirements for a RB . . . . .	8
<b>2</b>	<b>Assessment of the existing open source RB software</b>	<b>10</b>
2.1	GridWay . . . . .	10
2.2	Condor-G . . . . .	11
2.3	CSF Community Scheduler Framework . . . . .	12
2.4	GridBus . . . . .	12
2.5	NIMROD/G . . . . .	13
2.6	ASKALON . . . . .	14
2.7	GRMS GridLab Resource Management System . . . . .	15
2.8	WMS . . . . .	15
2.9	Moab/Maui . . . . .	16
2.10	Self-written RB for AstroGrid-D . . . . .	16
<b>3</b>	<b>Result of the assessment of the existing open source RB software and outlook on the usage of GridWay in AstroGrid-D</b>	<b>17</b>
3.1	Selection of the RB software GridWay . . . . .	17
3.2	Sketch of the usage of Gridway within AstroGrid-D . . . . .	17
3.2.1	General submission of jobs to the central Gridway instance . . . . .	17
3.2.2	Job submission from the ProC workflow engine to the central Gridway instance	17
3.2.3	Pre-selection of an execution host . . . . .	18
3.2.4	General submission of jobs to the local Gridway instance . . . . .	18
3.2.5	Job submission from the ProC workflow engine to the local Gridway instance	18
3.3	Concerns . . . . .	20
<b>4</b>	<b>AstroGrid-D JSDL extensions (preliminary)</b>	<b>20</b>
4.1	Structure . . . . .	21

---

4.2	CustomHardware Element . . . . .	21
4.2.1	Multiplicity . . . . .	21
4.2.2	Type . . . . .	21
4.2.3	Attributes . . . . .	21
4.2.4	Pseudo Schema . . . . .	21
4.3	Device Element . . . . .	22
4.3.1	Multiplicity . . . . .	22
4.3.2	Type . . . . .	22
4.3.3	Pseudo Schema . . . . .	22
4.3.4	DeviceName Element . . . . .	22
4.3.5	Multiplicity . . . . .	23
4.3.6	Type . . . . .	23
4.3.7	Pseudo Schema . . . . .	23
4.3.8	Examples . . . . .	23
4.4	DeviceCapabilites element . . . . .	23
4.4.1	Multiplicity . . . . .	23
4.4.2	Type . . . . .	23
4.5	Capability element . . . . .	23
4.5.1	Multiplicity . . . . .	23
4.5.2	Type . . . . .	24
4.5.3	Pseudo Schema . . . . .	24
4.5.4	Attributes . . . . .	24
4.6	Dependencies . . . . .	24
4.6.1	Multiplicity . . . . .	24
4.6.2	Type . . . . .	25
4.6.3	Pseudo Schema . . . . .	25
4.7	SoftwareComponent . . . . .	25
4.7.1	Multiplicity . . . . .	25
4.7.2	Type . . . . .	25
4.7.3	Pseudo Schema . . . . .	25
4.8	Version . . . . .	25

---

4.8.1	Multiplicity	25
4.8.2	Type	26
4.8.3	Pseudo Schema	26
4.9	DeviceConfiguration	26
4.9.1	Multiplicity	26
4.9.2	Type	26
4.9.3	Pseudo Schema	26
4.10	ConfigurationItem	26
4.11	Multiplicity	26
4.12	Type	26
4.13	Attributes	27
4.14	Pseudo Schema	27

## 1 Introduction

Within working group 5 (henceforth WG-5), we address the problem of how to distribute incoming tasks (compute jobs or tasks for virtual observatories) to the resources available on the grid. Specifically we consider the task of resource brokering for grid jobs. Resource brokering is a generalization of the functionality of batch or queue management systems as they are usually provided on clusters and on high performance computers.

### 1.1 Description of the requirements of a RB software component for use within AstroGrid-D

In Deliverable 5.1, elements of the resource management were outlined. They were (a) job description, (b) job or resource brokerage, (c) submission and execution, (d) job monitoring and steering and (e) job post-processing.

This document deals with step (b), job or resource brokerage. The task is to deliver a job from the user (client) side to a central resource broker, which then submits the job, after evaluating available resources and boundary conditions, to an execution queue at a compute node or observatory resource.

According to previous decisions in WG-5, we assume that the job is provided from the user side in a standardized job description language (JSDL); for this deliverable, we focus on relatively simple jobs which can fully be described in JSDL. We expect that JSDL soon will be supported by Globus, but note that with our presently installed version of the Globus Toolkit 4.0.3 does not support it, and we have to fall back to our JSDL/RSL translator. There is the future need to deal with more complex work flows, which will be an issue for the selection of software already now, but the main goal here is to process simple jobs.

On the resource side we assume that there is a job execution system, which could be the Globus GRAM, PBS, or Sun Grid Engine, for example.

### 1.1.1 General requirements of the RB software

The RB to be used within the AstroGrid-D project should fulfill the following general requirements:

1. The deployment (installation/configuration) should be easy to perform on all Grid middleware systems like Globus, Unicore, gLite, etc.
2. The use of the software within other projects is counted as a measure for the reliability of the software use and for the potential to have communication partners, if problems are encountered.
3. The description of jobs should meet the structure of the first prototype specified further down.
4. The software should bear the perspective to support more demanding jobs and workflows in future versions.
5. An interface for querying the job status and for canceling jobs would be preferred.
6. The implementation of the broker algorithms should be extendable and easy to be modified.
7. An integration with other AstroGrid-D middleware components (e.g., WS, library, languages) should be possible and preferentially easy to perform.
8. The RB software should be compatible with the needs for data staging before and after the actual job submission.
9. The RB software needs to provide suitable means for job identification through the brokering phase.

### 1.1.2 Metadata prototype

The structure of the first prototype of metadata used by the RB is assumed to be of the form:

- owner (certificate ... to authenticate and to authorize)
- type of hardware (IA32, IA64, Opteron, ...)
- number of processors
- os (win, lin, ...)
- software environment (including compiler, and libraries)
- executable
- input data

- output data
- stdout/err
- intermediate logfiles (same as stdout/err)
- wallclocktime, i.e., a user-defined limit on the execution time of a job
- need of parallel processing via message passing interface (MPI)

### 1.1.3 Design requirements for a RB

We specify the three important aspects *flexibility*, *robustness*, and *usability* for the design of a RB acting on a heterogenous infrastructure with very different application requirements.

*Usability:*

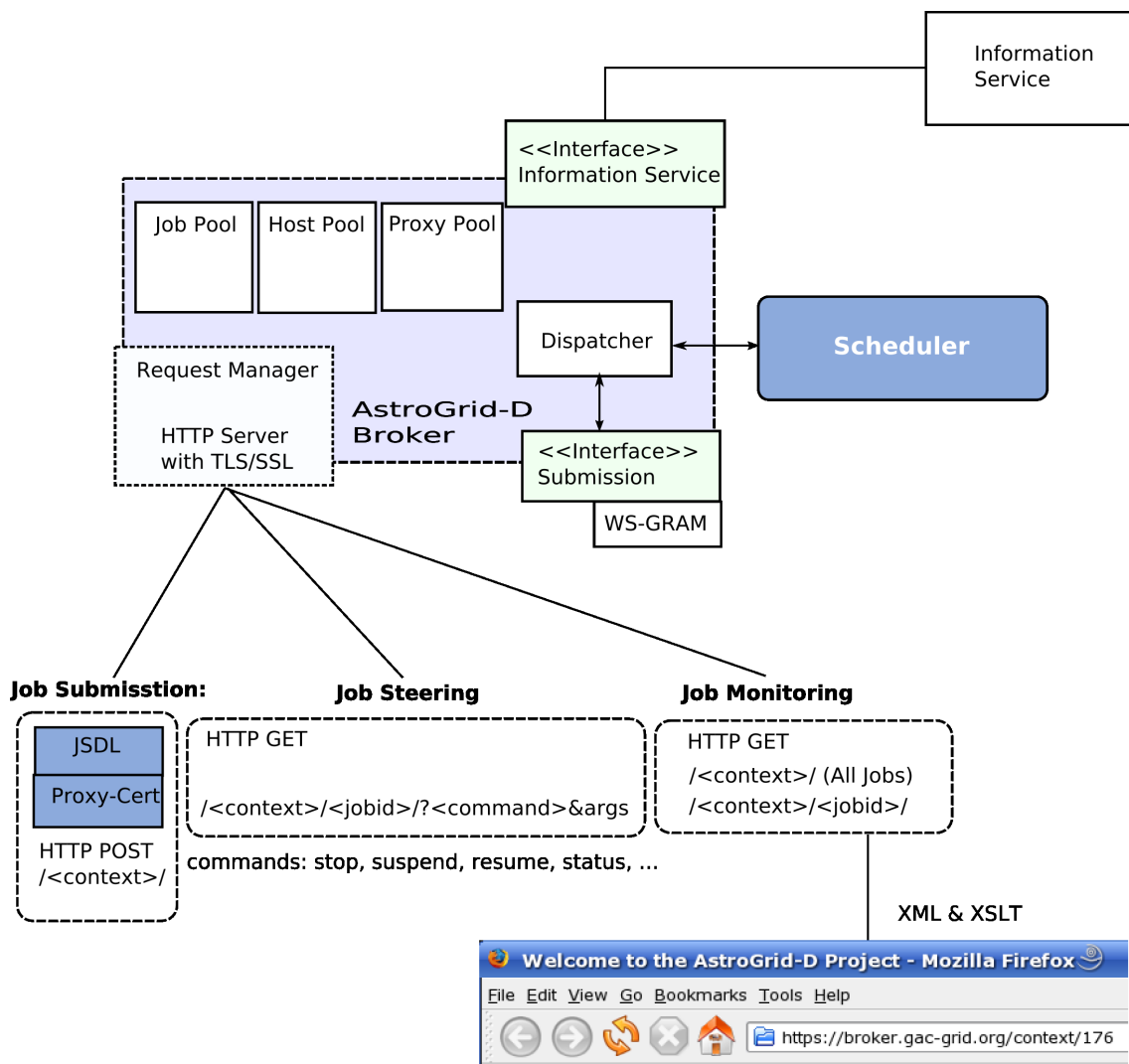
- Open, general purpose protocols should be used.
- The RB actions should be transparent for all applications.
- Usage of the RB should be simple (from users point of view).
- Jobs can be submitted, controlled, and monitored from *any* workstation (without requiring a full Globus installation)

*Flexibility (generic functionality):*

- A modular design is desired to interface different grid middleware components, and to support important job submission standards (e.g. DRMAA).
- The software should be platform independent, or at least source code portability should be achieved.
- Scheduling algorithms should make fewer assumptions about the available brokering metrics (just a minimum set of mandatory attributes), but should be able to include them if available.
- The RB should be configurable to run as a general purpose centralized instance, as well as as one or more specialized instances distributed over the Grid.

*Robustness:*

- An automatic failure recovery should be implemented.
- The software should be deployable on different sites for redundancy.



The architecture sketch shows a rough layout of components taking into account these requirements (some of the RB discussed in Sect. 2 like, e.g., GridWay, partially have the same components).

Job- submission, steering, and monitoring is managed using standard protocols and open data formats (HTTP, XML).

The Request Manager handles incoming job submission requests over a secured HTTP connection. It checks whether the submitted JSDL document and proxy certificate is valid and sends an acknowledgement back to the client.

The information about available grid resources and brokering metrics is polled from the AstroGrid-D Information Service. and put into a host pool which is used to accomplish the matchmaking. The dispatcher sends the job to an execution module which implements the job submission interface (e.g. WS-GRAM).

## 2 Assessment of the existing open source RB software

The members of the *Task Force Scheduler/Broker* have examined a couple of resource RB software systems, which are available for various purposes. In the following we describe the results of this assessments, which can also be found at.

<http://mintaka.aip.de:8080/lenya/intranet/workpackages/wg5/TaskForceSchedulerBroker.html>  
(access to the Intranet of AstroGrid-D needed).

### 2.1 GridWay

The GridWay Metascheduler is a higher level grid middleware, which uses the Globus Toolkit as the core grid middleware. It is designed to enable large-scale efficient sharing of computing resources (clusters, computing farms, servers, supercomputers...), managed by different LRM (Local Resource Management) systems, such as PBS, SGE, LSF, Condor, within a single organization (enterprise grid) or scattered across several administrative domains (partner or supply-chain grid). Details about the software can be found under [www.gridway.org](http://www.gridway.org). Main features of the architecture are sketched in the diagram under

<http://www.gridway.org/documentation/stable/installconfiguide/x203.htm>.

In our Heidelberg AstroGrid-D testbed, GridWay 5.0.1 was installed on a node with Globus Toolkit 4.0.3. Gridway can be installed in two different ways: single-user and multiple-user installation. In the multiple-user scenario the users are able to submit, control, and monitor their jobs *from* the GridWay server host.

The client tools of GridWay use TCP/IP to communicate with the GridWay server process. For authentication, the client uses the logged-in username of the calling process (which is usually the shell). Thus, Gridway supposes that the user accounts between the client and the GridWay server are shared.

The following supporting arguments for this RB can be named:

1. GridWay has become a Globus Project, and is therefore welcoming code and support contributions from individuals and corporations around the world. After establishing a communication channel with the developers, their quick response and the scope of their answers gave us the impression that a decent support can be expected for this software component.
2. The central server architecture fits well to the concepts given in D5.1.
3. The metascheduler is implemented on top of both the Globus Toolkit 2.4 & 4.x.
4. The job submission and the management support the GGF DRMA standard API.
5. The current dispatching mechanisms are of simple round-robin/flooding-type but can be extended.
6. The European Space Astronomy Centre of the European Space Agency at Villafranca (Spain) has been using GridWay since more than a year for one of its Grid projects.
7. GridWay permits job dependencies, which are a feature of some of the AstroGrid-D use cases.

8. GridWay support job migration/adaptive execution (grid- and application-initiated).
9. After some discussion with the authors of GridWay we got the impression that one of our most important requirements, the flexible definition of non-standard resource descriptions, is fulfilled.
10. The GridWay server provides a failure recovery feature where the daemon gwd tries to recover its previous state. This is accomplished by storing submitted jobs in a persistent pool, and in case of a crash these jobs are recovered. When starting the GridWay daemon, the state recovery can be circumvented by using the `-c` option.

A few questions will need further investigations in the framework of the next deliverable, and might be resolved only in direct contact with the developers:

1. The full compatibility to JSDL has not yet been verified.
2. The user authentication is not straightforward. It seems possible to establish it by adding of another Globus GRAM component between user clients and central broker.
3. For remote access to the GridWay server host, the GridWay 5 Installation and Configuration Guide <http://www.gridway.org/documentation/stable/installconfiguide.pdf> proposes to share user accounts and home directories via NIS and NFS and that the Gridway directory should be made available by exporting `$GW_LOCATION` via NFS. This operation mode is infeasible in a widely distributed network which goes beyond institute boundaries.
4. GridWay does *not* use open, general purpose protocols (at least not in the application layer) for communication between the client tools and the GridWay server host.

## 2.2 Condor-G

The Condor project has the goal to develop, implement, deploy, and evaluate mechanisms and policies that support High Throughput Computing (HTC) on large collections of distributively owned computing resources. Condor-G is the job management part of Condor using the Globus Toolkit. Further information can be found under Website:<http://www.cs.wisc.edu/condor/>.

Assessing the capabilities of this software package in the framework of the requirement specified above, the following features were noted:

- Condor-G uses Globus protocols like Grid Security Infrastructure, Grid Resource Allocation and Management, Global Access to Secondary Storage.
- Job exit codes are not available.
- Jobs are placed on hold when a problem is encountered.
- Condor-G is not a real broker as there is no scheduling.
- The package treats a grid as local resource.
- The matchmaking works for complete jobs and sites only.

The fact that Condor-G, in the current version, does not really involve a brokerage and scheduling facility excludes its use within the AstroGrid-D project.

### 2.3 CSF Community Scheduler Framework

The Community Scheduler Framework (CSF) is a set of Grid Services, implemented using the Globus Toolkit, which provides an environment for the development of metaschedulers that can dispatch jobs to resource managers such as LSF, SGE, PBS, and Condor. Further details can be found at the two pages

[www.globus.org/toolkit/docs/4.0/contributions/csf/](http://www.globus.org/toolkit/docs/4.0/contributions/csf/)  
<http://sourceforge.net/projects/gcsf/>.

The assessment of this open source project as part of the Globus project led to the following list of remarks:

- It supports implementation of grid metaschedulers based on GT services (MDS, GRAM, RFT, etc.).
- The RB enables communication between heterogeneous schedulers on local level.
- It offers simple dispatching mechanisms.
- The package is extensible via scheduling plug-ins
- It allows for submission, control, and monitor jobs at a grid level.
- In CSF, creation and management of advanced reservations at the grid level is possible.
- CSF can send jobs and advanced reservation operations to the local resource managers.
- It now features a plug-in interface so that site and user specific dispatching policies can be implemented regardless of the underlying resource manager.
- Creation of queues of jobs, each with separately definable dispatching policies, is enabled.

For CSF it was not clear, if the simple scheduling mechanisms are sufficient for the AstroGrid-D applications and how much effort realistically has to be invested to implement a new improved scheduling plug-in.

### 2.4 GridBus

The Gridbus project is engaged in the creation of open-source specifications, architecture, and a reference Grid toolkit implementation of service-oriented grid and utility computing technologies for eScience and eBusiness applications. The Gridbus software is being used in Grid-enabling a number of applications in science, engineering, and commerce. More details about GridBus and the GRB Grid Service Broker can be found under <http://www.gridbus.org>.

A sketch of the overall GridBus Middleware Architecture is shown at <http://www.gridbus.org/middleware/>, a publication about Gridbus by the project director

Rajkumar Buyya, Melbourne Univ. Australia is available under <http://www.gridbus.org/papers/gridbus2004.pdf>.

The following remarks have been collected during the assessment of GridBus:

- Gridbus is a free open source software and downloadable under LGPL license.
- The RB capabilities are more advanced in Gridbus than, e.g., in GridWay.
- Gridbus supports Globus (GT4), Alchemi, PBS, Unicore, and SGE.
- It deploys and monitors job execution on selected resources.
- It is designed to optimally map jobs to resources.
- When testing the software, a JSDL document was rejected as not all JSDL language elements are supported
- A communication with developers could not be established.
- The architecture of Gridbus has no central server, so it is in contradiction with concepts described in D5.1.
- It became not clear during the assessment how it can handle access failures of single nodes.
- Gridbus can access data from local or remote sources.
- This RB is used in real high energy physics science cases.
- It supports the WSRF webservice standard, and there is a Gridbus Broker Workbench GUI available for easier composition, initialization, and management of grid applications.
- It features Workflow support by the GridBusWorkFlow Engine GWFE which uses an XML based language to describe workflows. It also supports GT4.0 GRAM, PBS, and SGE.
- The project collaborates with grid economy, trading, and accounting services
- Gridbus Interacts with the Workflow Management Service.
- The package leverages an economic scheduling model.

The deviation from the central server architecture is a severe contradiction with the concepts established in D5.1. The pending communication attempts with the developer leave it open in which way software support can be expected for this RB.

## 2.5 NIMROD/G

Nimrod is a specialized parametric modeling system. It uses a simple declarative parametric modeling language to express a parametric experiment and provides machinery that automates the task of formulating, running, monitoring, and collating the results from the multiple individual experiments. Nimrod incorporates a distributed scheduling component that can manage the scheduling of individual experiments to idle computers in a local area network. Nimrod/G is the Grid aware

version of Nimrod. It takes advantage of features supported in the Globus toolkit such as automatic discovery of allowed resources. Furthermore, the concept of computational economy is introduced as part of the Nimrod/G scheduler. The architecture is extensible enough to use any other grid-middleware services such as Legion, Condor, and NetSolve. Details about Nimrod are given under <http://www.csse.monash.edu.au/~davida/nimrod/>. The architecture of Nimrod/G is sketched under <http://www.csse.monash.edu.au/~davida/nimrod/nimrodg.htm>.

A few more details have been given during the assessment:

- A user GUI is provided.
- The information given at the webpages are much less deep and clear than in the case of Gridbus or other RB software.
- Gridbus lists Nimrod/G as an optional sub-component.
- The latest release news refer to the date 27 July 2005.

As a clear reason to exclude Nimrod/G for the use within the AstroGrid-D project, from the architecture it became clear that Nimrod/G relies on Enfuzion API which is a commercial software.

## 2.6 ASKALON

Askalon is a Programming Environment and Tool Set for Cluster and Grid Computing developed by the University of Innsbruck (group of Thomas Fahringer). The set includes the following tools:

- the automatic performance bottleneck analysis tool AKSUM
- the performance modeling and prediction system PROPHET
- the performance instrumentation, measurement, and analysis tool SCALEA
- the automatic performance experiment management system ZENTURIO
- and an integrated GUI (Java-based coordination and visualization system for tools).

The webpages of the project [www.dps.uibk.ac.at/projects/askalon/parallel](http://www.dps.uibk.ac.at/projects/askalon/parallel) are detailed and contain further links for each tool.

Tests with this RB have been performed, leading to the following assessment remarks:

- The GUI (TechPreview) was easy to install on AstroGrid-D resources. The actual compute resources including the broker were already installed on a testbed in Innsbruck.
- Single actions (job submission) need some time (10s - 20s).
- Since Askalon is not available as Open Source, the technical and license conditions for a usage by AstroGrid-D need to be resolved first.
- The architecture and the scheduling algorithms are documented in publications.
- Each activity in a workflow can pose constraints on the hardware self-written application.

- Most ASKALON tools so far support mostly Fortran90 (including MPI, OpenMP, HPF, mixed OpenMP/MPI, and mixed HPF/OpenMP) based distributed and parallel programs.
- A Java-based programming paradigm for performance-oriented parallel and distributed computing as been developed (JavaSymphony), which will be supported by the ASKALON tools in the near future.

## 2.7 GRMS GridLab Resource Management System

The GridLab Resource Management System (GRMS) is an open source meta-scheduling system, developed under the GridLab Project, which allows developers to build and deploy resource management systems for large scale distributed computing infrastructures. The GridLab page is [www.gridlab.org](http://www.gridlab.org), and the GRMS pages can be found under <http://www.gridlab.org/WorkPackages/wp-9/>.

The assessment lead to the following remarks:

- GRMS is based on GT 2.4 (assured to work on GT 4.x).
- Job dependencies are planned to be supported.
- It defines a Job Definition Language (GJD).
- It uses a simple scheduling mechanisms which is extensible
- GRMS allows job migration.
- A GSI enabled Web Service interface is provided.

The GridLab Project finished in April 2005 (funding by the European Commission), the last update of the webpages is dated to 09-Mar-2006. An attempt to contact the key persons in Poland was not successful as the person is no longer reachable. It appears that this software is no longer maintained and supported.

## 2.8 WMS

The Workload Management System (WMS) is designed as a set of Grid middleware components responsible for the distribution and management of tasks across Grid resources, in such a way that applications are conveniently, efficiently and effectively executed. Details about WMS can be found under <http://egee-jra1-wm.mi.infn.it/egee-jra1-wm/wms.shtml>.

The WMS was developed within the EGEE-EU project which has funding until March 2008. The EGEE grid has about 20000 CPUs and 5 Petabytes storage. WMS is no metascheduler, and it submits only to gLite compute elements (at least work nodes have to be installed). The use within AstroGrid-D would require installation of gLite on all AstroGrid-D nodes, which is not trivial (estimated to take about one year before all components are running).

## 2.9 Moab/Maui

The Moab Grid Suite is a commercial grid workload management solution used by some of the world's largest grids that integrates scheduling, management, monitoring and reporting of workloads across independent clusters. The Maui Cluster Scheduler, the precursor to Moab Cluster Suite, is an open source job scheduler for clusters and supercomputers. It is an optimized, configurable tool capable of supporting an array of scheduling policies, dynamic priorities, extensive reservations, and fairshare capabilities.

Details can be found at the homepage <http://www.clusterresources.com/pages/products/maui-cluster-scheduler.php>. The following details have been found during the assessment:

- Cluster Resources is a leading provider of workload and resource management software and has an impressive list of commercial customers with a large number of nodes.
- Maui is an advanced resource management systems with features like a built-in HPC simulator for analyzing workload, resource, and policy changes
- Maui requires PBS, Loadleveler, SGE, BProc, SSS XML, LSF or Wiki FlatText Scheduling APIs.
- It can run on Linux, AIX, OSF/Tru-64, Solaris, HP-UX, IRIX, FreeBSD, and other UNIX platforms.

While Maui is a solution for cluster batch system, it was not clear if the software can be used for Grid resource brokerage.

## 2.10 Self-written RB for AstroGrid-D

In two communities, the HEP and the C3 communities, projects are underway to develop community-specific solutions for the resource brokering problem. At least for the HEP community, resource brokering includes storage space. The main reason not to use a common grid scheduler is the implemented hierarchical concept of meta schedulers like GridWay. A central server being aware of all resources, connection properties, and job requests works well for a limited spectrum of community applications, but encounters problems for cross-community cooperations. In turn, giving up the hierarchical concept for a flexible market place solution, each scheduler can decide for itself which workflow and task he accepts first. Moreover, self-written software can be optimized for the use of the special communities. The C3 grid has developed a simple scheduler and they hope to use external collaborations with other projects to enhance the brokering abilities of this scheduler within the AstroGrid-D project lifetime. They will explicitly rely on the usage of standard interfaces to make a future adaption to other solutions possible. For the AstroGrid-D project, doubts were raised by the assessing team whether the project has enough man power equipped with the experience required to write a RB with sufficient functionality within the lifetime of the project.

### **3 Result of the assessment of the existing open source RB software and outlook on the usage of GridWay in AstroGrid-D**

#### **3.1 Selection of the RB software GridWay**

As a result of these assessments, *the Gridway software was selected for further testing and processing of simple grid jobs*. The second candidate, GridBus, will be also further observed, as a reserve. From the specific list of requirements given above, the main reasons to chose this RB were its central server architecture, the expected developer support for this Globus Proto Project, the supported GGF DRMA standard API, and its flexibility to use job dependencies. GridWay 5.0.1 was consequently installed on a node with Globus Toolkit 4.0.3 within the Heidelberg ZAH Astrogrid testbed. Intensive testing has been started and is expected to continue.

#### **3.2 Sketch of the usage of Gridway within AstroGrid-D**

The RB is intended to optimally distribute individual jobs as well as complete workflows (submitted, e.g., by the Planck Process Coordinator (ProC) from the submitting to the executing host.

AstroGrid-D will offer at least one central instance of Gridway. In addition, for special purposes, AstroGrid-D nodes may run one or more local Gridway instances. The central Gridway instance will be accessible via the ordinary Globus job submission mechanism.

##### **3.2.1 General submission of jobs to the central Gridway instance**

The following scenario describes the brokering mechanism: The user submits a job via Globus Toolkit 4 to a central Globus server instance. Its GRAM is configured in such a way that it passes the job to the central Gridway instance. Gridway analyzes the job description, consults the *Stellaris* AstroGrid-D information system for information about the resources that are available at the execution hosts, performs the job-to-resource matching operation, selects an execution host from the list of capable execution hosts, and redirects the job to the selected execution host. Pre-staging and post-staging of data files will be handled completely transparently to the user, either as if the job were executed on the central Globus server instance that is hosting Gridway, or as if the job had been directly submitted to the execution host selected by Gridway. It is possible to define resource descriptions for each execution host. Examples are the availability of special hardware (e.g. grape boards), the availability of software (e.g. Subversion, GNU Make, Ant, Java 1.4, GNU Scientific Library, ImageMagick), and the availability of data (e.g. Millennium simulation).

##### **3.2.2 Job submission from the ProC workflow engine to the central Gridway instance**

The Planck Process Coordinator (ProC) workflow engine will be able to submit jobs (which may contain workflows) to the central instance of the Gridway resource broker usually via the built-in GAT Globus Toolkit adapter. The resource description must be passed through the GAT adapter via Globus GRAM to the central instance of Gridway which is installed behind Globus GRAM.

### **3.2.3 Pre-selection of an execution host**

It will be possible to send a job to a particular pre-selected execution host. One simple mechanism will be to define a unique host name property (e.g. Lx32ia1@LRZ, astrogrid@TUM, or the full URL) describing an execution host, and to add to the job description a list of desired hosts. In case the list has just one entry, the job will end up at the desired execution host.

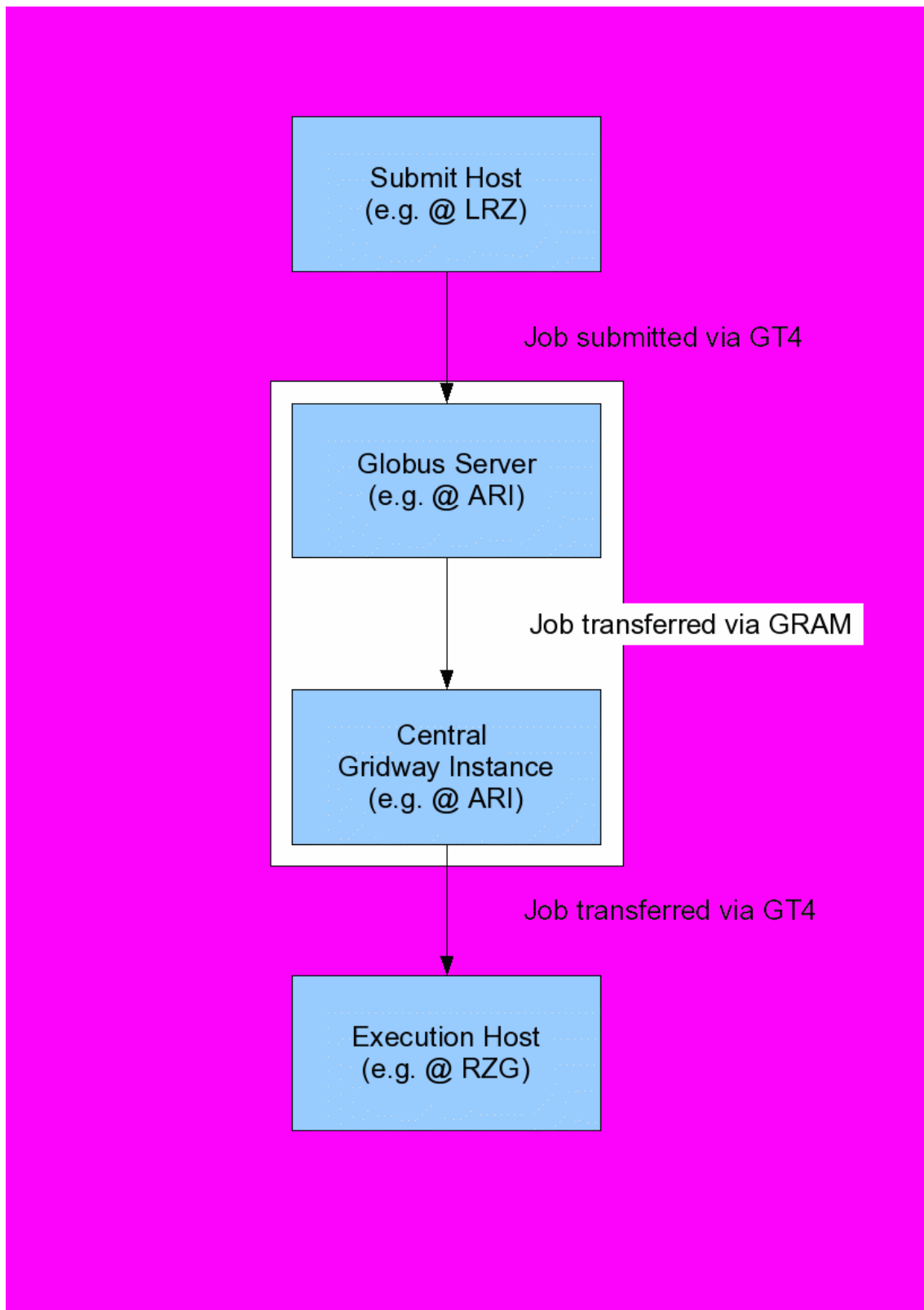
The need for such a pre-selection may arise e.g. during testing, or because there are deployment problems at other execution hosts that needs to be solved first.

### **3.2.4 General submission of jobs to the local Gridway instance**

For special purposes a local instance of Gridway may be used. This may either be configured similarly to the central Gridway instance, i.e. running behind a Globus GRAM, or in a stand-alone fashion. This may be necessary when the configuration of the central instance does not meet the requirements of a particular user or application, or simply for testing purposes.

### **3.2.5 Job submission from the ProC workflow engine to the local Gridway instance**

It may be desirable to submit jobs from the ProC directly to the Gridway RB. In this case, we would want to go through a GAT adaptor to the DRMAA interface of Gridway, obviating the need for an intermediary Globus instance. Such an adaptor does currently not exist, but is presumably easy to create, based on the existing GAT-DRMAA adaptor for the SGE developed at the Albert-Einstein-Institut, Golm. As long as jobs can satisfactorily be submitted via the GAT GT4 adapter to the central instance of Gridway and from there further-on into the Grid, there is no need for a GAT-DRMAA adaptor to Gridway.



Flow-chart of the resource brokering process via the Gridway gateway. [TBS]

### 3.3 Concerns

The default scenario with a centralized Gridway instance acting as a gateway has raised some concerns, which may be summarized as follows:

- The architecture may be too complex. A Globus instance needs to accept the job request, pass it on, through GRAM, to the central Gridway instance, which then re-submits (or migrates) the job to a Grid compute node.
- Is Globus capable of passing through all the job information required by the Gridway instance so that it can accomplish its job?
- What about the Job ID? Will the Job ID, which is (presumably) being assigned by the central Globus instance to the job submitted, be maintained by Gridway, so that job monitoring doesn't become a nightmare? One way of accomplishing this might be job migration. (To us it is still unclear whether GT4 offers such a mechanism.)
- What about data staging? Usually data pre-staging takes place before job submission, and data post-staging takes place after the job has finished. The data is being transferred from the submit host to the execution host. If the latter is a 'virtual' compute node, only hosting Gridway gateway, but not actually executing any jobs, it has to transfer the data further to the real execution host. This mechanism might create an unbearable I/O load on the Gridway host, and an unnecessary increase the network data traffic.
- Gridway appears not to accept standard HTTP requests (or being interfaced to Web services). Such a standard protocol would allow (restricted) job submissions from any user host including laptops. In addition, a simple job submission GUI could be conceived, e.g. behind a GridSphere portal, that would obviate the need for installing any special Grid software on the submit host.

The concerns above will be addressed in the imminent cooperation between ZAH/ARI and MPA, where the Gridway software will be further examined and tried. The present status is that job submission via Gridway has successfully been tested, but the problem remains to be checked how a job initiating from a remote client (with respect to the Gridway server) could be handled.

## 4 AstroGrid-D JSDL extensions (preliminary)

JSDL – the job description language used in AstroGrid-D was designed to cover all required elements for standard (POSIX) Applications including common resource requirements and data staging operations. To be platform independent it does not specify features dedicated to a specific LRMS or special job types, although the GGF is working on the specification to address a wider class of jobs, namely web services.

To comply all the special requirements in AstroGrid-D it is necessary to extend the job description to fulfill our needs. This section defines an extended JSDL document referred to as the *AstroGrid-D JSDL profile*.

One extension to the JSDL specification concerns the description of specialized resources such as GRAPE boards. This extension should allow the user to specify the requirement of custom hardware

devices. The vocabulary is very generic and allows detailed description of requirements such as the features and configuration of the hardware.

### **[FIXME: Description of other extensions]**

The AstroGrid-D JSDL profile is based on the Job Submission Description Language 1.0 and the GRAM JSDL extensions for the Globus Toolkit, defined by the Globus Alliance. Workflows are (at this time) not in the scope of this specification.

## **4.1 Structure**

The structure of this section follows the schema of the Job Submission Description Language (JSDL) specification document, Version 1.0

*Prefixes and namespaces used in this specification:*

xsd	<a href="http://www.w3.org/2001/XMLSchema">http://www.w3.org/2001/XMLSchema</a>
agd	<a href="http://www.gac-grid.org/2007/jsdl">http://www.gac-grid.org/2007/jsdl</a>

## **4.2 CustomHardware Element**

When the job requires special hardware to run the job the description needs to contain resource selection information for the resource broker. The CustomHardware element is a complex type specifying a set of devices which are required to run the job.

### **4.2.1 Multiplicity**

The multiplicity of this element is zero or one.

### **4.2.2 Type**

This is a complex type. It MUST support the following elements:

- Device

### **4.2.3 Attributes**

No attributes are defined.

### **4.2.4 Pseudo Schema**

```
<agd:CustomHardware>
  <agd:Device ... />+
  <xsd:any###other />*
</agd:CustomHardware>?
```

### 4.3 Device Element

The Device element contains information about the device which **MUST** be available at the execution host. If there is no resource available providing all devices the job **MUST NOT** be executed.

#### 4.3.1 Multiplicity

The multiplicity of this element is one or more.

#### 4.3.2 Type

This is a complex type. It **MUST** support the following elements:

- DeviceName
- DeviceCapabilities
- Dependencies
- DeviceConfiguration
- DeviceDescription

#### 4.3.3 Pseudo Schema

```
<agd:Device>
  <agd:DeviceName ... />
  <agd:DeviceCapabilities />?
  <agd:Dependencies />?
  <agd:DeviceConfiguration ... />?
  <agd:DeviceDescription ... />?
  <xsd:any###other />*
</agd:Device>+
```

#### 4.3.4 DeviceName Element

The DeviceName element specifies a name for device and is used as an identifier for the consuming system.

### 4.3.5 Multiplicity

The multiplicity of this element is one.

### 4.3.6 Type

The type of this element is `xsd:string`.

### 4.3.7 Pseudo Schema

```
<agd:DeviceName>xsd:string</agd:DeviceName>
```

### 4.3.8 Examples

```
<agd:DeviceName>GRAPE6a</agd:DeviceName>
```

## 4.4 DeviceCapabilites element

The DeviceCapabilities element provides a generic way to describe requirements for attributes and features of a device.

### 4.4.1 Multiplicity

The multiplicity of this element is zero or one.

### 4.4.2 Type

This is a complex type. It must support the following elements:

- Capability

## 4.5 Capability element

The Capability element describes one feature of a device. The feature itself can be further described using the Attribute element. All device capabilities **MUST** be present at the execution hosting to comply the requirements.

### 4.5.1 Multiplicity

The multiplicity of this element is one or more

## 4.5.2 Type

The type of this element is xsd:string.

## 4.5.3 Pseudo Schema

```
<agd:Capability name="xsd:NCName">
  <agd:Attribute .../>*
  <xsd:any##other/>*
</agd:Capability>+
```

## 4.5.4 Attributes

The following attributes are defined:

- name - A mandatory name for the Capability. Its type is xsd:string.

**4.5.4.1 Attribute** The Attribute element defines an attribute for a device capability. All defined attributes must comply to comply the device capability.

**4.5.4.2 Multiplicity** The multiplicity of this element is zero or more.

**4.5.4.3 Type** The type of this element is xsd:string.

```
<agd:Attribute name="xsd:NCName">
  xsd:string
</agd:Attribute>*
```

## 4.5.4.4 Pseudo Schema

## 4.6 Dependencies

The Dependencies element describes all dependencies for a device which must be met to perform the job. If this is not present then it is not defined and the consuming system MAY choose any value.

### 4.6.1 Multiplicity

The multiplicity of this element is zero or one.

## 4.6.2 Type

This is a complex type. It MUST support the following elements:

- SoftwarePackage

## 4.6.3 Pseudo Schema

```
<agd:Dependencies>
  <agd:SoftwareComponent ... />*
  <xsd:any##other />*
</agd:Dependencies>?
```

## 4.7 SoftwareComponent

The SoftwarePackage element specifies a software component or package which must be available at the execution host.

### 4.7.1 Multiplicity

The multiplicity of this element is zero or more.

### 4.7.2 Type

The is a complex type. It MUST support the following elements:

- Version

### 4.7.3 Pseudo Schema

```
<agd:SoftwareComponent name="xsd:NCName">
  <agd:Version ... />?
  <xsd:any##other />*
</agd:Dependencies>?
```

## 4.8 Version

The Version element defines a version for the SoftwareComponent

### 4.8.1 Multiplicity

The multiplicity of this element is zero or one.

### 4.8.2 Type

The type of this element is xsd:string.

### 4.8.3 Pseudo Schema

```
<agd:Version>xsd:string</agd:Version>?
```

## 4.9 DeviceConfiguration

The DeviceConfiguration element specifies how a device MUST be configured to run the job. If the DeviceConfiguration element is not used the Consuming system MAY choose any value.

### 4.9.1 Multiplicity

The multiplicity of this element is zero or one.

### 4.9.2 Type

This is a complex type. It MUST support the following elements:

### 4.9.3 Pseudo Schema

```
<agd:DeviceConfiguration>  
  <agd:ConfigurationItem .../>+  
</agd:DeviceConfiguration>?
```

## 4.10 ConfigurationItem

A ConfigurationItem is a name value pair which describes one configuration item of a device.

### 4.11 Multiplicity

The multiplicity of this element is one or more.

### 4.12 Type

The type of this element is xsd:string.

### 4.13 Attributes

The following attributes are defined:

- name – A mandatory name for the ConfigurationItem element. This name is defined by the resource/device provider

### 4.14 Pseudo Schema

```
<agd:ConfigurationItem name="xsd:NCName">  
  xsd:string  
</agd:ConfigurationItem>
```