

# ASKALON Grid Environment

## User Guide

Version 1.2

ASKALON Team  
Prof. Fahringer

September 3, 2015

Distributed and Parallel Systems Group  
Institute of Computer Science  
University of Innsbruck



## Revision Log

Date	Version	Author	Log
2007/02	1.0	Jun Quin	Initial
2013/07	1.1	Walter Gugenberger	Updated
2014/12	1.2	Roland Mathá	Updated

## Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Terminology</b>	<b>2</b>
<b>3</b>	<b>Install Manual</b>	<b>4</b>
3.1	Askalon Installation Steps . . . . .	4
3.1.1	Start Globus container on localhost . . . . .	5
3.1.2	Update Steps . . . . .	5
3.2	GroudSim Installation Steps . . . . .	6
<b>4</b>	<b>System Overview</b>	<b>7</b>
4.1	From Abstract to Concrete . . . . .	7
4.2	ASKALON Components Overview . . . . .	8
<b>5</b>	<b>Resource Management</b>	<b>9</b>
5.1	Resource Brokerage . . . . .	9
5.1.1	Installation . . . . .	10
5.1.2	Configuration . . . . .	10
5.1.3	Customization . . . . .	11
5.1.4	Tools . . . . .	12
5.2	GLARE . . . . .	13
5.2.1	Installation . . . . .	14
5.2.2	Configuration . . . . .	14
5.2.3	Tools . . . . .	14
<b>6</b>	<b>Port Your Application onto the Grid</b>	<b>16</b>
6.1	Describing & Registering Your Application . . . . .	16
<b>7</b>	<b>Compose Your Workflow</b>	<b>21</b>
7.1	Start the Workflow Composition Tool . . . . .	21
7.2	Main GUI . . . . .	21
7.3	Activity Types . . . . .	23
7.4	Compose Workflows . . . . .	23
7.4.1	Create Workflow Initial Node and Final Node . . . . .	24
7.4.2	Create Atomic Activities . . . . .	25
7.4.3	Create Compound Activities . . . . .	26
7.4.4	Create Control Flows . . . . .	27
7.4.5	Create Data Flows . . . . .	27
<b>8</b>	<b>Execute your Workflow</b>	<b>30</b>
8.1	Setup the Environment . . . . .	30
8.2	Start the Workflow Execution . . . . .	30
<b>9</b>	<b>Monitor Your Workflow</b>	<b>32</b>
<b>A</b>	<b>UML Representation of AGWL Activities</b>	<b>35</b>

## List of Figures

1	Relationship among Activities . . . . .	2
2	Concretizing an activity in AGWL . . . . .	3
3	Development process . . . . .	7
4	The ASKALON Components . . . . .	8
5	GridARM architecture . . . . .	9
6	GridARM GLARE URI configuration . . . . .	13
7	GridARM Console with physical resource . . . . .	13
8	Activities on the GLARE . . . . .	15
9	Sample gwdd . . . . .	18
10	Sample buildFile . . . . .	19
11	Deploying an application onto the Grid . . . . .	20
12	Digital signature warning . . . . .	21
13	Main GUI . . . . .	22
14	The povray Workflow . . . . .	24
15	Workflow Input . . . . .	24
16	Automatically created DataIn ports . . . . .	25
17	Workflow checking . . . . .	29
18	The GridARM and GLARE Settings dialog . . . . .	30
19	Execution Mode of the Workflow Editor . . . . .	31

## 1 Introduction

The goal of Askalon[1] is to simplify the development and optimization of applications that can harness the power of Grid computing. The Askalon project crafts a novel environment based on new innovative tools, services, and methodologies to make Grid application development and optimization for real applications an everyday practice.

Many real world Grid workflow applications from different domains have been ported into Askalon, such as Wien2K from materials science, Invmod from hydrology, MeteoAG from meteorology, GRASIL from astrophysics, povray from image rendering, etc. This document will use the povray workflow application as an example to describe how to use Askalon. For more background information about povray, go to <http://www.povray.org>

This document does not include the complete installation steps of all Askalon services, which is also not required for the users who want to have a try Askalon within our test bed (<http://www.dps.uibk.ac.at/askalon/techpreview01>). For detailed information about Askalon, go to <http://www.askalon.org>.

## 2 Terminology

### AGWL

AGWL[2] stands for Abstract Grid Workflow Language. It is the workflow language used in ASKALON

### Activity

Activity in AGWL represents a computational task, which can be an Atomic Activity or a Compound Activity

### Atomic Activity

An Atomic Activity is an activity which is implemented by a single computational entity (an executable, a web service operation, etc.)

### Compound Activity

A Compound Activity is an activity which encloses some Atomic Activities or other Compound Activities. The inner activities are connected by control and data flows. A workflow or sub-workflow can also be considered as a Compound Activity.

### Inner Activity

The Inner Activities are the outermost activities, e.g. Atomic Activities or Compound Activities, in Compound Activities.

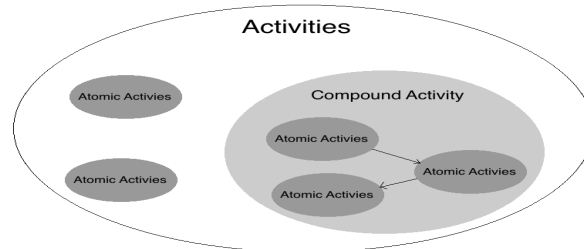


Figure 1: The relationship among Activities, Atomic Activities and Compound Activities

### Activity Type (AT)

An Activity Type represents a kind of functionality in the Grid. It is defined by an Activity Type name and input and output data ports. Each Atomic Activity in AGWL workflows has an Activity Type. An AT can be implemented by different Activity Implementations. The data types used in AGWL are based on the data types system supported by XML schema. In order to process files, some user-defined data types such as `agwl:file`, which denotes a file, and `agwl:collection`, which denotes multiple files, are defined. ATs are stored in the GLARE service in the ASKALON Grid infrastructure.

### Activity Implementation (AI) & Activity Deployment (AD)

An Activity Implementation (AI) is a specific implementation of a computational task. It usually refers to a computational entity which can be an executable, a web service operation, etc. The AIs associated with a AT must have the exactly same functional behavior and the exactly same input and output data structure (Performance behavior may be different). An AI deployed on a Grid site is called Activity Deployment (AD). ADs are stored the GLARE service in the ASKALON Grid. The AIs and ADs are not visible in AGWL. AIs will not be discussed in this document.

### Activity Instance

An Activity Instance is an invocation or an execution of an Activity Deployment. An Activity Deployment can be invoked several times on several Grid sites in which case they are different Activity Instances. Activity Instances are not visible in AGWL and will therefore not be discussed in this document.

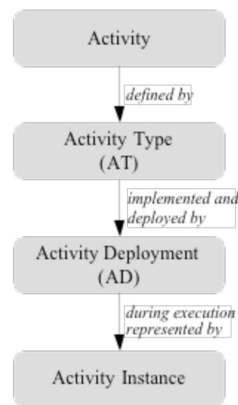


Figure 2: Concretizing an activity in AGWL

## 3 Install Manual

### Requirements

- Java (Version: 1.6+)
- Ant (Version: 1.7+)

### 3.1 Askalon Installation Steps

1. Copy „ws-core-4.0.8-bin.tar.gz“ into your home directory:

```
scp sunray:/home/simon/zip/ws-core-4.0.8-bin.tar.gz .
```

2. Unzip „ws-core-4.0.8-bin.tar.gz“:

```
tar -xvfz ws-core-4.0.8-bin.tar.gz [Target Directory]
```

3. Copy Askalon from a repository:

```
hg clone -b default ssh://kreusspitze.dps.uibk.ac.at  
/home/matthias/hg-repos/askalon3
```

**Note:** if hg is not installed, install it first (Fedora 20):

```
sudo yum install mercurial
```

4. Set the variables *ASKALON\_HOME* and *GLOBUS\_LOCATION*:

- (a) Using export

```
export ASKALON_HOME="/home/(username)/askalon3"  
export GLOBUS_LOCATION="/home/(username)/ws-core  
-4.0.8"
```

- (b) **Permanently**

Open /etc/environment with an editor

```
sudo gedit /etc/environment
```

and append the following lines

```
ASKALON_HOME="/home/(username)/askalon3"  
GLOBUS_LOCATION="/home/(username)/ws-core-4.0.8"
```

5. Execute the following commands:

```
cd $ASKALON_HOME  
ant clean  
ant dist
```

6. Start Askalon (in *ASKALON\_HOME* Directory):

```
ant exec
```



### 3.1.1 Start Globus container on localhost

1. **Generate private/public key pair**

```
ssh-keygen -t rsa -C "_your@email.adress"
```

2. **Copy public key to authorized\_keys**

```
cd .ssh  
cp id_rsa.pub authorized_keys
```

3. **Copy keys from local machine to karwendel@dps.uibk.ac.at: ~/.ssh**

```
scp authorized_keys karwendel@dps.uibk.ac.at:~/.ssh  
scp id_rsa karwendel:~/.ssh  
scp id_rsa.pub karwendel:~/.ssh
```

4. **Set ANT\_HOME**

```
export ANT_HOME=/usr/share/ant
```

5. **Deploy ee2**

```
cd $ASKALON_HOME/ee2/  
ant deploy
```

6. **Deploy GridARM**

```
cd $ASKALON_HOME/gridarm/  
ant deploy
```

7. **Start Globus container**

```
$GLOBUS_LOCATION/bin/globus-start-container -p <  
portnumber> -nosec
```

### 3.1.2 Update Steps

1. **Pull changes into your repository**

```
cd $ASKALON_HOME  
hg pull
```

2. **Update your working directory:**

```
hg update
```

3. **Deploy Askalon, EE2 and GridARM:**

```
cd $ASKALON_HOME  
ant clean  
ant dist  
cd $ASKALON_HOME/ee2/  
ant deploy  
cd $ASKALON_HOME/gridarm/  
ant deploy
```

## 3.2 GroudSim Installation Steps

The following steps show a correct way to install and run a GroudSim-enabled Globus container from the ASKALON source code.

### Prerequisites:

- checked-out ASKALON source code
- installed GT4 (WS-Core) and env-variable GLOBUS\_LOCATION pointing to it
- installed Apache Ant and JDK 1.5+

Assuming you're located at ASKALON's base directory, simply run the following commands to build the project, deploy the necessary projects and launch a Globus container.

```
ant dist
ant -f shared/build.xml deploy
ant -f ee2/build.xml deploy
ant -f gridarm/build.xml deploy
ant -f groudsm/build.xml deploy
ant -f groudsm/build.xml setupFreshGroudSimDirectory
ant -f groudsm/build.xml justExecGroudSimContainer
```

Finally, you can start Teuta by running:

```
ant justExec
```

## 4 System Overview

### 4.1 From Abstract to Concrete

Figure 3 shows an overview of the development process of Grid workflow applications in the ASKALON Grid application development environment[1] from an abstract representation to an actual execution on a Grid infrastructure. The development process consists of three fundamental procedures: Composition, Reification and Execution.

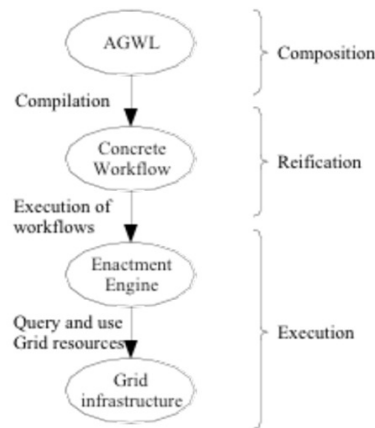


Figure 3: The development process of Grid workflow applications in ASKALON

**Composition:** The user composes the Grid workflow by using a graphical user interface or writing an AGWL program directly. At this level, activities are associated with activity types. There is no notion of how the input data is actually delivered to activities and how activities are implemented, invoked and terminated. AGWL contains all the information specified by the user during the workflow composition.

**Reification:** A transformation system in ASKALON compiles AGWL into a concrete representation through mapping abstract activities into specific Activity Deployments deployed in the Grid. Concrete representations can be scheduled and executed.

**Execution:** A concrete representation is interpreted by the underlying workflow runtime environment of ASKALON to construct and execute the Grid workflow application on a Grid infrastructure.

## 4.2 ASKALON Components Overview

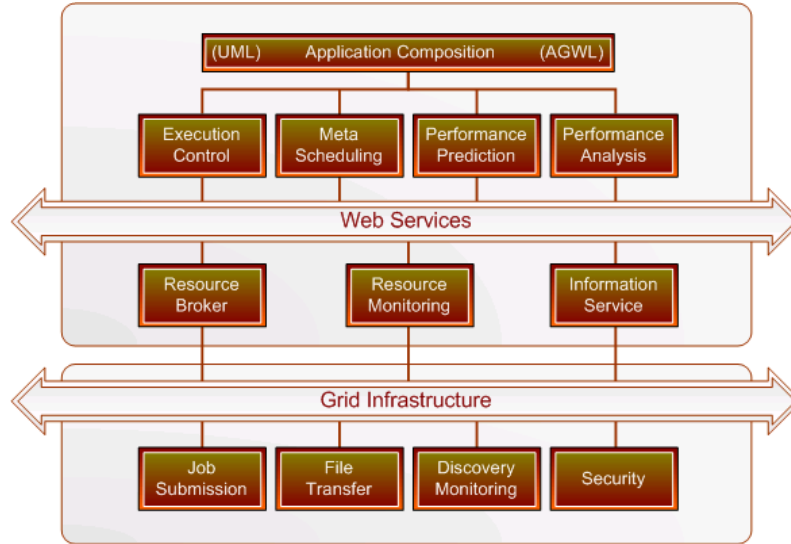


Figure 4: The ASKALON Components

**Resource broker** service targets negotiation and reservation of resources required to execute a Grid application.

**Resource monitoring** supports the monitoring of Grid resources by integrating and extending existing Grid resource monitoring tools and by developing new techniques for Grid resource monitoring (e.g. rule-based monitoring).

**Information service** is a general purpose service for scalable discovery, organization, and maintenance of resource and application-specific data (including online and post-mortem).

**Workflow executor** service targets coordinated activation, and fault tolerant completion of activities onto the remote Grid sites. (Meta)-scheduler performs appropriate mapping of single or multiple workflow applications onto the Grid. This work continues our initial scheduling efforts in the context of the ZENTU-RIO experiment management and optimization tool.

**Performance prediction** is a service through which we are currently investigating new techniques for accurate estimation of execution time of atomic activities and data transfers, as well as of Grid resource availability.

**Performance analysis** is a service that unifies the performance monitoring, instrumentation and analysis for Grid applications and supports the performance bottleneck interpretation.

## 5 Resource Management

This chapter introduces ASKALON's Grid resource management system also called GridARM, and describes how to deploy and use it on a Grid infrastructure. GridARM (ASKALON's Grid Resource Management system) provides the following functionality:

- Resource Brokerage
- Advance Reservation
- Capacity Planning

The overall architecture ASKALON resource management is depicted in Figure 5. Resource brokerage service provides resource discovery and selection. GLARE (A Grid Activity Registration, Deployment, and Provisioning Framework) is part of the GridARM and enables user to manage logical resources of the Grid for on-demand provisioning. Logical resources include software components (activity deployments) and Grid/Web services. This requires installation of two sub-packages (GridARM and GLARE) of ASKALON software.

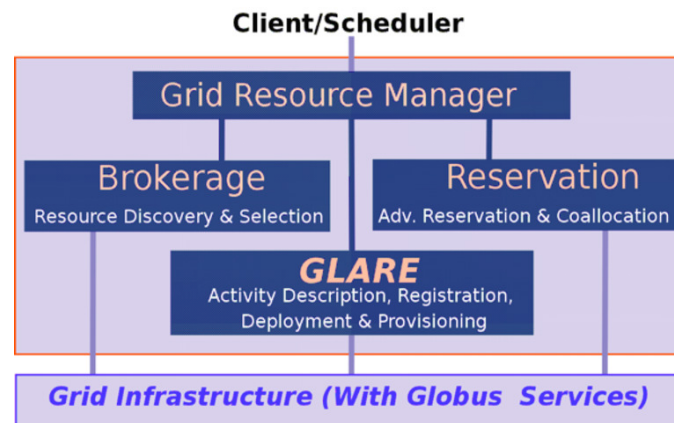


Figure 5: GridARM architecture

Advance reservation and capacity planning is the second part of the resource management and it provides advance reservation of both physical and logical resources while optimizing resource utilization with proper capacity planning for the resource allocations. This is provided by another sub-package (Agreement) of ASKALON.

### 5.1 Resource Brokerage

Resource discovery and selection in a distributed environment like ASKALON is quite important and needs to be automated. This section describes installation and use of GridARM package which provides brokerage of physical resources.

### 5.1.1 Installation

The GridARM brokerage service is part of ASKALON core package and available under `$ASKALON_HOME/gridarm/gridarm` directory. It can be extended with Agreement and GLARE packages as described in section 5.2.

#### Prerequisites

- This package depends on GT4.x (wsrf-core) that can be downloaded from <http://www.globus.org/toolkit/> and installed.
- `GLOBUS_LOCATION` environment variable must be set to proper location before starting deployment of GridARM. Please refer to installation instruction of GT4.x available on <http://www.globus.org/toolkit> for more details on how to install GT4 and how to set `GLOBUS_LOCATION` environment.

#### Deployment

- Download ASKALON software from [www.askalon.org](http://www.askalon.org) and un-bundle (un-tar or unzip) in a directory. We refer to this directory as `ASKALON_HOME`.
- In a system shell execute the following steps in order to install entire resource management:

```
$ set GLOBUS_LOCATION=/path/to/GT4.x
$ set ASKALON_HOME=/path/to/askalon
$ cd $ASKALON_HOME/gridarm
$ ant deploy
```

It is also possible that you get a pre-compiled package in the form of .gar (Grid Archive). GridARM.gar package is created by making ‘\$ant dist’ in the same directory, which can be used on any GT4-enabled machine to deploy GridARM services. Use the following command to do this:

```
$GLOBUS_LOCATION/bin/globus-deploy-gar GridARM.gar
$GLOBUS_LOCATION/bin/globus-deploy-gar gridarm_glare.gar
$GLOBUS_LOCATION/bin/globus-deploy-gar gridarm_agreement.gar
```

### 5.1.2 Configuration

After successful deployment, we need to provide Grid Information Service (GIS) Configurations. Currently the default supported GIS is MDS2. MDS2 is a monitoring and Discovery Service provided by Globus. Please refer to <http://www.globus.org/toolkit/docs/4.0/info/> for more details. Multiple GIS configurations can be provided in separate files under `$GRIDARM_HOME/gis/` directory. `$GRIDARM_HOME` environment is optional and if not set then it refers to `~/.gridarm`. The name of the GIS configuration file should follow the convention `<gis-name>.gisconf`, otherwise the configuration file will be ignored. Here is a

sample configuration file (also available at `$GLOBUS_LOCATION/etc/gridarm`) which points to an MDS2 installation in the AustrianGrid infrastructure.

```
<GISConfiguration xmlns="http://GridARM.askalon.org"
  Address = "agrid.uibk.ac.at"
  basedDn = "mds-vo-name=local,o=grid"
  name = "austrianGrid"
  port = "2170"
  type = "mds2"/>
```

Here is the detail of each element specified in the above configuration file. Most of the information should be available from GIS system admin.

<b>Address</b>	This is a fully-qualified DNS name of GIS server. In case MDS2, it should be LDAP server address.
<b>Port</b>	The port on which GIS service is running
<b>baseDn</b>	Refers to base name only required for MDS2. MDS2 system admin provides this information.
<b>Type</b>	Type of information service
<b>Name</b>	Unique name among all GIS configurations
<b>Xmlns</b>	This is XML namespace that is used for (global) uniqueness of XML elements. If you are not familiar to XML and XML namespace, please refer to eXtensible Markup Language documentation.

### 5.1.3 Customization

ASKALON resource manager is customizable. One can override or transform broker service according to the client requirement. In order to override some of the static attributes of a particular resource then provide a file called `$GRIDARM_HOME/nodes/<nodeName>`. This file can include any of the following attributes:

```
# Copy this file to $GRIDARM_HOME/nodes/<nodeName> and
# change/uncomment the following values.
# LRMSType           = pbs
# LRMSVersion        = 2.1.6
# TotalCPUs          = 16
# AccessPoint        = <manager-str>
# e.g. altix1.uibk.ac.at:2119/jm-pbs
# OSName             = Redhat
# ProcessorModel      = Itanium
# ProcessorVendor     = Intel
# InstructionSet      = ia64
...
# ProcessorClockSpeed = 2048
```

Here are some special flags which can be used for bypassing or ignoring discovery of actual or customized resources. For example, presence of a dummy file `$GRIDARM_HOME/nodes/.ignoreLocal` will ignore the customized resources, whereas `$GRIDARM_HOME/nodes/.ignoreRemote` enables broker to ignore resource information available from any of the configured GIS service. If none of the above flags are provided then attributes of the resources discovered by MDS2 will be overridden as given in the configuration file. By default resource manager (broker) runs user query on an aggregated set of resources discovered from different Grid information services, but you can also create your own customized resource manager without involving system administrator. This can be useful in certain situations: sometimes during experiments, a user may want to hide some Grid nodes, or override some of the attributes of a certain node. For example, a client may be interested in simulating 32b machine as if it is 64b. There is no need to change scheduler or executor configuration to make it work. Just create custom RM using command line tool and configure your broker URI properly in the GLARE. This is replacement of old style customization that is available by using `~/.GridARM/nodes/***`. The replacement enables you to use resource manager not running under your control, which is quite normal in production quality environments. Use the following command in order to create your customized resource broker: `$GLOBUS_LOCATION/bin/GridARM-resource-manager -s <serviceURI>-X configFile` configFile is an XML file in which customization can be specified. A sample file is available under directory `$GLOBUS_LOCATION/etc/GridARM/` as `customized-rm-config.xml`. Please note that a proxy delegation mechanism should be configured in all services like scheduler and executor in order to work with this kind of customization. Please refer to globus toolkit documentation for more details about proxy delegation mechanism and configuration.

#### 5.1.4 Tools

If you are not using resource brokerage capability directly then just skip this section. The GridARM brokerage service can be accessed using the following command line tool as:

```
$GLOBUS_LOCATION/bin/GridARM-resource-manager [-h] <-s|-e  
URI/EPR>[-D|-S] [-R <name exact lower upper>] ...
```

Type the following command in a shell for detailed usage of this command:  
**`$GLOBUS_LOCATION/bin/GridARM-resource-manager -h`**

Here is an example usage:

```
$GLOBUS_LOCATION/bin/GridARM-resource-manager -s <URI  
of GlobusContainer with GridARM>-C - 16 48 -R ProcessorClock-  
Speed 1500 1000 2000
```

`-s <URI>` refers to the GT4 container where GridARM services are running.

Note that “`-C - 16 48`” defines a constraint for TotalCPUs in which no exact value is specified (provided ‘-’ instead), but lower bound(=16) and upper bound(=48) are given. The `-R` flag provides a named constraint. In the example



above, ProcessorClockSpeed constraint is passed to the broker with 1500MHz speed as preferred value along with lower bound (1000MHz) and upper bound (2000MHz).

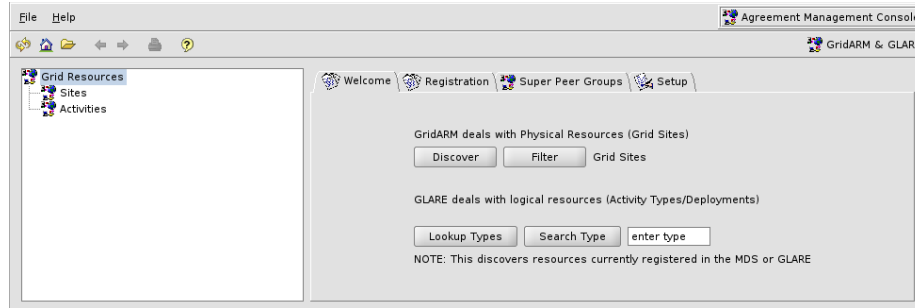


Figure 6: GridARM GLARE URI configuration

The GridARM console is a graphical tool available as part of ASKALON main GUI. It can be invoked from ASKALON resources menu or standalone by using the following command: `$GLOBUS_LOCATION/bin/GridARM-console`. The first thing that needs to be done is to configure GLARE URI. Go to setup/configure tab as shown in Figure 6, and enter the GLARE service URI. Please refer to section 5.2 GLARE for more detail about GLARE configuration and usage.

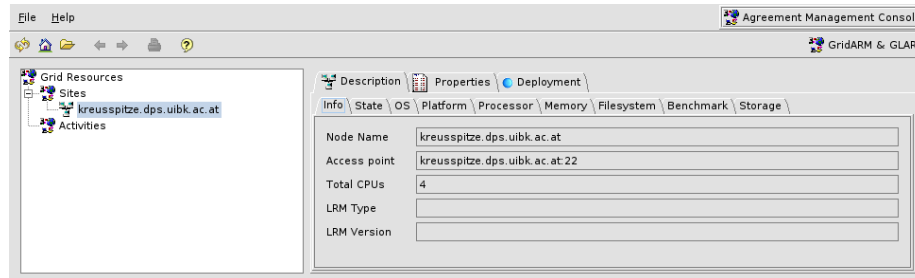


Figure 7: GridARM Console with physical resource

## 5.2 GLARE

GLARE is the second and very essential component of resource management that deals with logical resources including components legacy scientific application (executables) and Grid/Web services. It provides activity deployments (AD) on-demand. In order to deploy workflow applications on the Grid automatically, you need to install GridARM package on all grid sites and GLARE package on at least one highly available grid site. Practically this should be done by site administrators so that application providers can focus on porting their application independently. It is possible that you can use one GLARE service instance on a single site. In this case a subset of GLARE features can

be used, like on-demand provisioning of activity deployments. In such a case automatic deployment on other sites will not be possible.

### 5.2.1 Installation

Along with GridARM, GLARE package needs to be installed on at least one grid site. It is available as part of ASKALON under `$ASKALON_HOME/gridarm/glare`. Please follow the steps described in section 5.1.1 for GLARE package installation.

**Prerequisites** Please note that GLARE depends on GridARM package and needs Globus Toolkit 4.x with WS-MDS. WS-MDS is Globus Toolkit 4 information service. Please refer to GT4.x documentation for more details.

### 5.2.2 Configuration

GLARE can work as standalone service (without auto-deployment feature enabled) or can be configured in a distributed environment as a cooperating set of services. GLARE configuration is quite simple. As a standalone service deploy it under GT4 and start the container. Please refer to GT4 installation documents for more details about container. This is, in fact, minimum requirement in order to start working with GLARE. After having a single GLARE instance, you can start deployment of GridARM on other grid sites. The only change that needs to be done in GridARM configuration is as:

Edit the file `$GRIDARM_HOME//GridARM-config.properties` and add:

`GLAREURI=<GLAREServiceURI>`,

e.g. `GLAREURI=http://agrid.uibk.ac.at:40105`

Multiple GLARE services can be run in a way that they can cover a set of grid-sites in a super-peer distributed model and refer to each other for more activity deployment discovery and provisioning. In order to enable this feature, do the following: Edit `$GRIDARM_HOME/GridARM-config.properties` and add **GridARMReferences** property pointing to some or all other GridARM services excluding itself.

### 5.2.3 Tools

A command line tool `$GLOBUS_LOCATION/bin/GridARM-deployment-manager` can be used to register new applications in the GLARE and then deploy them on different grid sites. Please type the following command for detailed usage:

**`$GLOBUS_LOCATION/bin/GridARM-deployment-manager -h`**

Applications can always be un-deployed after the use. Furthermore, the same feature can be invoked on-demand by a client application such as scheduler. Please refer to section 6 for more details on how to port your application onto the Grid.

As described in the previous section, the GridARM console can be used to browse available resources on the Grid. Along with physical resources like Grid sites, logical resources such as activity type and deployments can also be discovered, browsed, registered/unregistered and deployed/undeployed. Activities

are organized under application name in the left tree pane. Select application or activity type and then browse details on the right site.

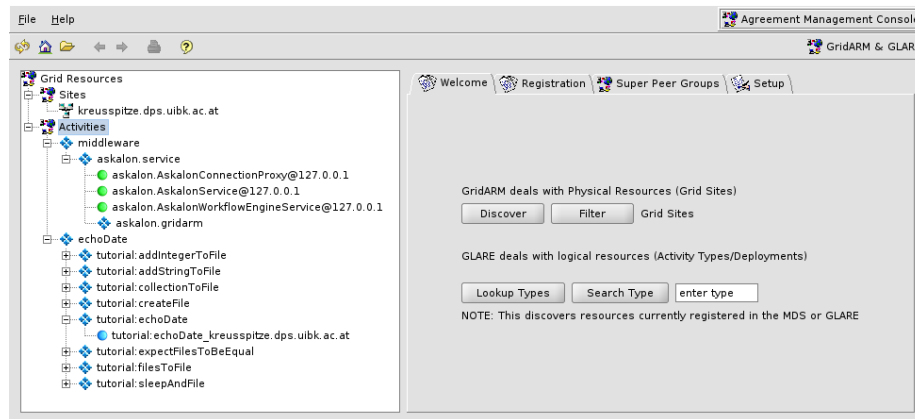


Figure 8: Browsing and managing activities on the GLARE

Applications (for example `invmod` and `wien2k` shown in the Figure 8) can be deployed and undeployed on selected grid sites. First select application, then optionally press Refresh to get all GridARM-enabled grid sites. This will ensure that only those sites show up on which automatic deployment is possible. Once the Deploy button is pressed, a progress dialog will show up. Installation steps being performed on remote sites will be displayed in the progress window. You can hide the progress window while deployment is in progress.

Once you have deployed your application or you have discovered other applications along with their activities, you can browse activity deployments as well. Deployments can be discovered and viewed either by node name or by type names. Client applications can perform more fine grained queries for the deployments including constraints like argument/port types etc.

If you want to update an activity type or activity deployment using console, just select the entry in the tree view and then press Update found under the configure tab.

## 6 Port Your Application onto the Grid

This chapter describes how to port your workflow applications onto the Grid that is enabled with ASKALON middleware services. Some well adapted patterns could increase the application productivity and efficiency while running on the Grid.

Following point could be helpful in preparing and making an application for Grid ready.

1. Carefully identify various components of your application that can be run autonomously or with least dependencies on each other.
2. Identify components (activities) that can be parallelized.
3. Identify activities which are more dependent on other activities. In this case combining those those activities into a single activity could be useful.
4. Merge activities which could create more overhead in running on the Grid independently than merging into a single activity. This can be done by for example wrapping in a single script.
5. Once you have identified components of your application then define input and output along with their data types for each activity. It is more likely that output of one activity is input of other one. Therefore they need to be well defined.

At abstract level i.e. while defining activity type descriptions, types of input and output parameters are more important than actual references to them.

### 6.1 Describing & Registering Your Application

The simple and efficient way of describing your application and preparing for registration in the ASKALON-enabled Grid is to define two files called GWDD (Grid Workflow application Deployment Descriptor) and buildFile and register them in the GLARE. GWDD is a properties file in which application along with its components (activities) are defined.

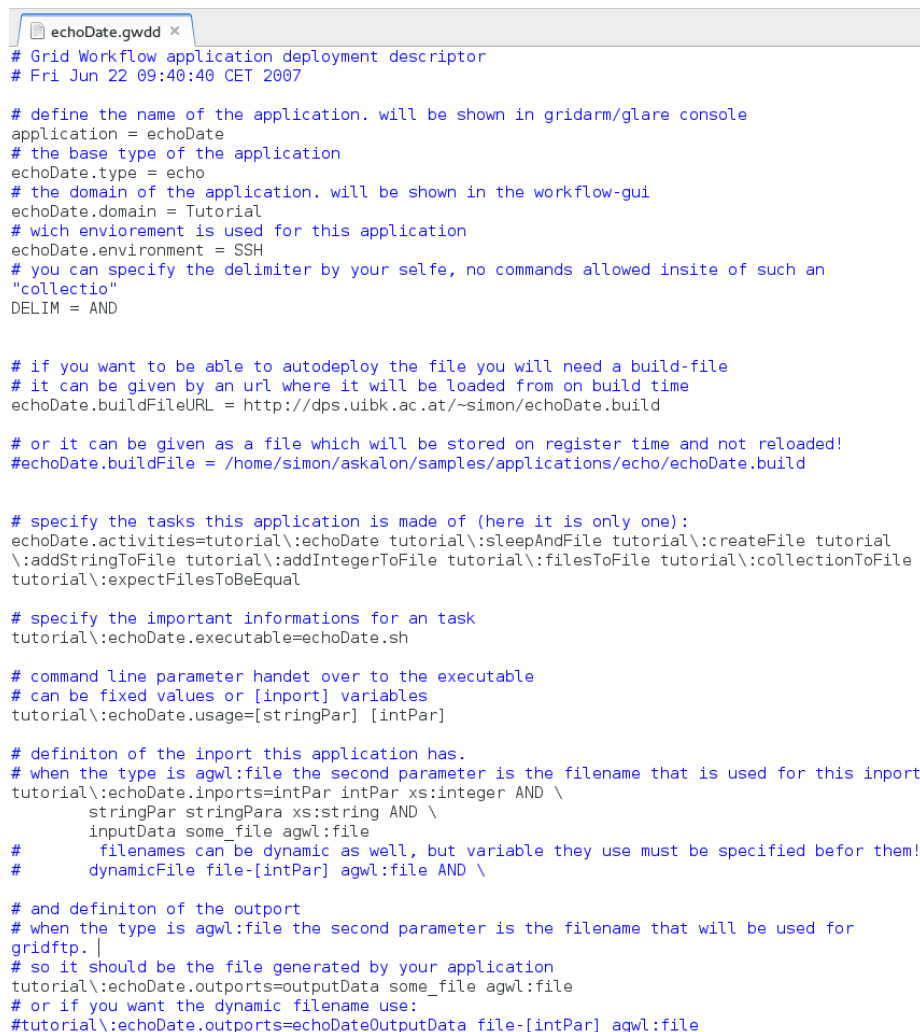
A sample GWDD for the invmod application is shown in Figure 12. You need to define following properties in GWDD file.

application	name of your application
<application>.type	Base type of your application
<application>.domain	Main domain of your application
<application>.environment	Hosting environment of your application
DELIM	Delimiter used to separate multiple items in a value of a single property
<application>.buildFile	/path/to/your/buildFile (as defined later)
<application>.buildFileURL	URL of your build file (required if <application>.buildFile is not specified.
<application>.activities	Space separated names of your activity types
<activities>.executable	Name of executable for a particular activity
<activities>.usage	Usage of your activity executable
<activities>.inports	<DELIM>separated list of your input arguments in the form of <name><value><type>. This is required for each activity.
<activities>.outports	<DELIM>separated list of your output arguments in the form of <name><value><type>. This is required for each activity.
<activities>.accessPaths	specify the grid sites where the binaries are already installed

Properties like *application*, *type*, *domain* etc. define your application; <application>.activities gives space separated list of activities. <activity>.inports and <activity>.outports gives DELIM separated list of all inport and output arguments. Each inport/output argument should be defined as a triple delimited by spaces <argumentName portName type>. Here argument type is important to define and discover underlying concrete descriptions (activity deployments). PortName is name of generated arguments relative to its associated activity.<activity>.[executable |usage] etc. are concrete information which remain same across multiple sites and paths. Other concrete information like <activity>.path and <activity>.host is added after the deployment of application on a particular site.

If you have already deployed your application manually, then add <activity>.path and <activity>.host properties and activity deployments for that particular host will be registered automatically along with application registration (see below for application registration detail).

The second file that needs to be provided along with your application description is build file. The buildFile as name suggests, describes all steps required for automatic or on-demand deployment/installation of your application. Figure 10 shows a sample build file for the sample application GWDD shown in Figure 9. Note that the build file looks similar to an ant build file but with some flexibility. For instance, it could be possible that you would like to pass official URL of your application without modifying its installable bundle. In this case things would



```

echoDate.gwdd x
# Grid Workflow application deployment descriptor
# Fri Jun 22 09:40:40 CET 2007

# define the name of the application. will be shown in gridarm/glare console
application = echoDate
# the base type of the application
echoDate.type = echo
# the domain of the application. will be shown in the workflow-gui
echoDate.domain = Tutorial
# wich enviroement is used for this application
echoDate.environment = SSH
# you can specify the delimiter by your selfe, no commands allowed insite of such an
"collectio"
DELIM = AND

# if you want to be able to autodeploy the file you will need a build-file
# it can be given by an url where it will be loaded from on build time
echoDate.buildFileURL = http://dps.uibk.ac.at/~simon/echoDate.build

# or it can be given as a file which will be stored on register time and not reloaded!
#echoDate.buildFile = /home/simon/askalon/samples/applications/echo/echoDate.build

# specify the tasks this application is made of (here it is only one):
echoDate.activities=tutorial\:echoDate tutorial\:sleepAndFile tutorial\:createFile tutorial
\:addStringToFile tutorial\:addIntegerToFile tutorial\:filesToFile tutorial\:collectionToFile
tutorial\:expectFilesToBeEqual

# specify the important informations for an task
tutorial\:echoDate.executable=echoDate.sh

# command line parameter handet over to the executable
# can be fixed values or [inport] variables
tutorial\:echoDate.usage=[stringPar] [intPar]

# definiton of the inport this application has.
# when the type is agwl:file the second parameter is the filename that is used for this inport
tutorial\:echoDate.inports=intPar intPar xs:integer AND \
    stringPar stringPara xs:string AND \
    inputData some_file agwl:file
# filenames can be dynamic as well, but variable they use must be specified before them!
# dynamicFile file-[intPar] agwl:file AND \

# and definiton of the outport
# when the type is agwl:file the second parameter is the filename that will be used for
gridftp. |
# so it should be the file generated by your application
tutorial\:echoDate.outports=outputData some_file agwl:file
# or if you want the dynamic filename use:
#tutorial\:echoDate.outports=echoDateOutputData file-[intPar] agwl:file

```

Figure 9: A Sample Grid Workflow application Deployment Descriptor

be complicated if your installation bundle prompts for user input during installation, like it may ask for license acceptance. In such a situation, you can pass your input as send/expect like pattern using dialog element of your build step as

```
<Dialog expect="Agree with term and conditions?" send="yes"/>
```

Beside this, the possibility of defining various steps in a well defined order enables your to flexibly deal with automatic installation of your application. Following is a list of elements or attributes that can be used in a buildFile.

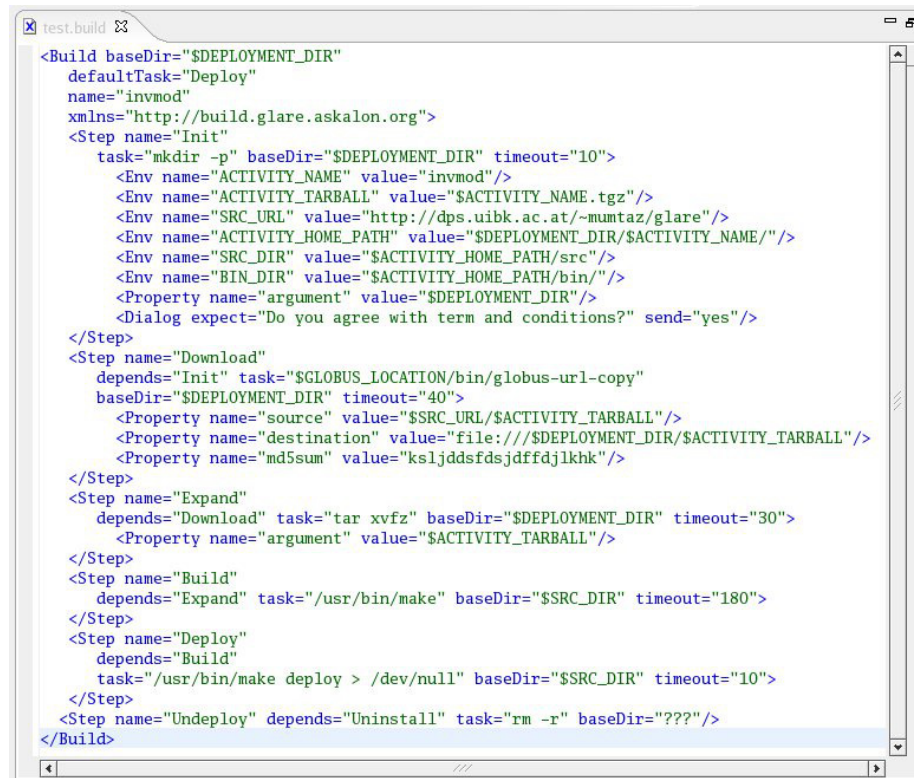


Figure 10: Sample buildFile

Step	Element that describes a single task that is to be perform atomically
Env	Element that describes a single environment variable that needs to be set before performing the task (optional).
Property	Element that describes a single argument or parameter that is passed to the task (optional).
Task	An attribute that defines tool/command that is executed for each step
defaultTask	Starting task
baseDir	Base directory in which step should be executed (optional).
depends	Provides step name that should be performed before 'this' step.

\$GLOBUS\_LOCATION/bin/GridARM-deployment-manager  
 -s <GLAREServiceURI> -R test.gwdd

This will register generic description of your application (as given in GWDD)

and underlying activities.

The registered application and its activities can then be discovered and viewed in GridARM console and operation like deploy/undeploy and register/unregister can be performed on different sites.

If you want to deploy your newly registered application on a certain grid site, proceed as:

- Select registered application in the tree view of the console window.
- Go to configure tab on right pane and select grid site in sites comboBox. A refresh is always recommended before this step. The refresh will show up only with those sites where automatic deployment will be possible.
- Finally press “Deploy” button to initiate installation process on the selected site.
- A dialog as depicted in Figure 11, will appear showing installation progress. Press hide on top left of the progress dialog to send it to the background.

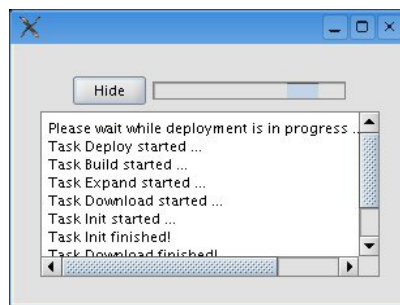


Figure 11: Deploying an application onto the Grid

The command line `gridarm-deployment-manager` can also be used for this purpose. Enter following command for detailed usage.

`$GLOBUS.LOCATION/bin/gridarm-deployment-manager -h`

The alternative way of porting your application onto the Grid, is by using graphical tool (console). You can add new activity types as follows:

- Go to main console window and select registration tab, and press Create Activity Type button to add new type.
- Press Create Activity Type button to add new type.
- A new dialog will show up that can be used to add activity type description.

While registering your application graphically (without GWDD file), the first type you enter should describe your application. Once you add the application type (the base activity type), then you can add subtypes under the application describing its activities.



## 7 Compose Your Workflow

We assume that your applications have already been ported and deployed in the Grid as explained in Chapter 6 Port Your Application onto the Grid. In this chapter, we will show you how to combine your individual activities into a workflow.

### 7.1 Start the Workflow Composition Tool

The workflow composition tool is distributed via Java Web Start. To start it, open your web browser, go to: <http://www.askalon.org> and click on "ASKALON GUI Webstart version". When you are asked to trust the developer, please click "run" (Figure 12).

**Distribution:**

- Teuta (the old ASKALON workflow composition tool) is distributed free of charge for educational and research purposes upon request. Until now we have received requests. In order to start the download procedure please [click here](#).
- Click [here](#) to run the **new** ASKALON workflow composition tool (~ 55 MB, requires J2SE Runtime Environment 5.0+) using [Java™ Web Start](#) (Last update: 2013-01-10)



Figure 12: Digital signature warning

**Note that if the workflow composition tool does not start after you click, please check:**

1. whether you have Java Web Start installed on your machine
2. whether your web browser recognizes ".jnlp" files correctly

### 7.2 Main GUI

As shown in Figure 13, the main GUI mainly consists of an Activity Types panel, a workflow diagram, a Properties panel, a UML Toolbar. The buttons on the UML toolbar (move your mouse onto the button, you will see the tooltips) are as follows:

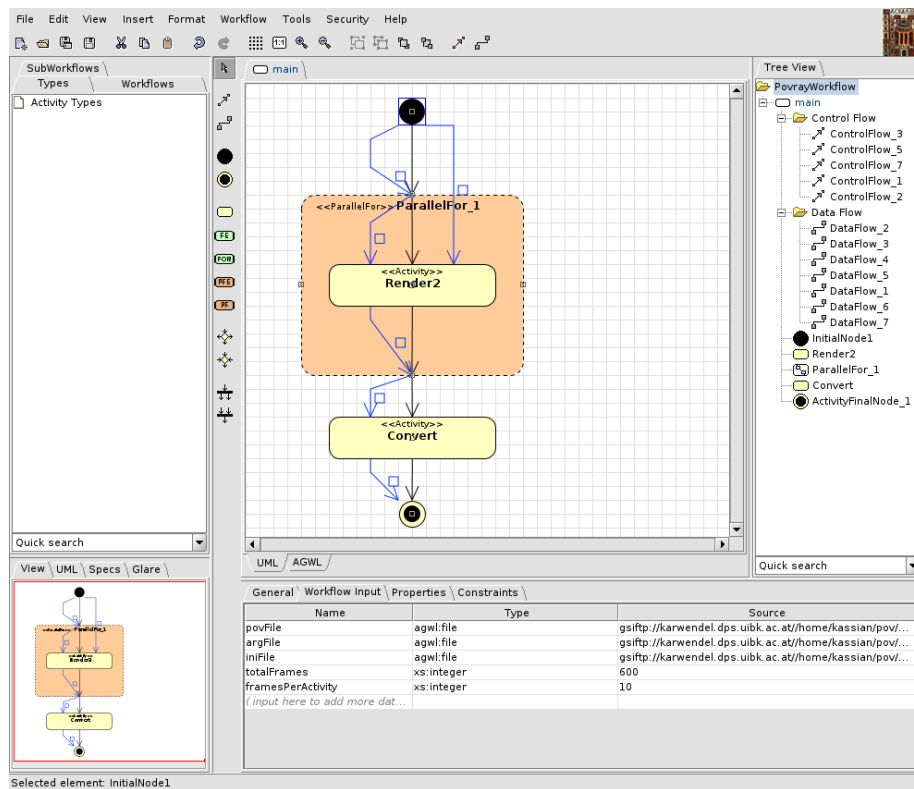


Figure 13: Main GUI



### Marquee Selection

By selecting this button, you can move nodes and edges (and control points on the edges), select multiple nodes and edges by drawing a rectangle in the workflow diagram. Among marquee selection button, Data flow button and Control flow button, only one can be selected at a time.



### Data Flow

Select this button to draw a data flow edge.



### Control Flow

Select this button to draw a control edge.



### Initial Node

Click this button to create a workflow InitialNode in the workflow diagram.

**Activity Final Node**

Click this button to create a workflow ActivityFinalNode in the workflow diagram.

**Atomic Activity**

Click this button to create an atomic activity in the workflow diagram.

**Decision Node**

Click this button to create a DecisionNode in the workflow diagram. DecisionNode is used when you compose an If, Switch, While, DoWhile compound activity.

**Merge Node**

Click this button to create a MergeNode in the workflow diagram. MergeNode is used when you compose an If or Switch compound activity.

**Fork Node**

Click this button to create a ForkNode in the workflow diagram. ForkNode and JoinNode are used when you compose a Parallel compound activity.

**Join Node**

Click this button to create a JoinNode in the workflow diagram.

## 7.3 Activity Types

Check if you have some deployed activity types in the Activity Types panel. For details about how to deploy activity types and activity deployments into the Grid, please refer to Chapter 6 (Port Your Application onto the Grid). If you do not see any available activity types in the panel, check:

1. Your GLARE configuration through the menu Tools | Configure GLARE ...
2. Refresh the Activity Types panel through the combo box at the bottom of the panel

## 7.4 Compose Workflows

We now use the povray workflow (Figure 14) as an example to show you how to compose a workflow in ASKALON.

The povray workflow is a movie rendering workflow that has an activity Render in it. The activity Render is associated with an activity type povray:Render, which is mapped at runtime to an specific implementation POV-Ray raytracer.

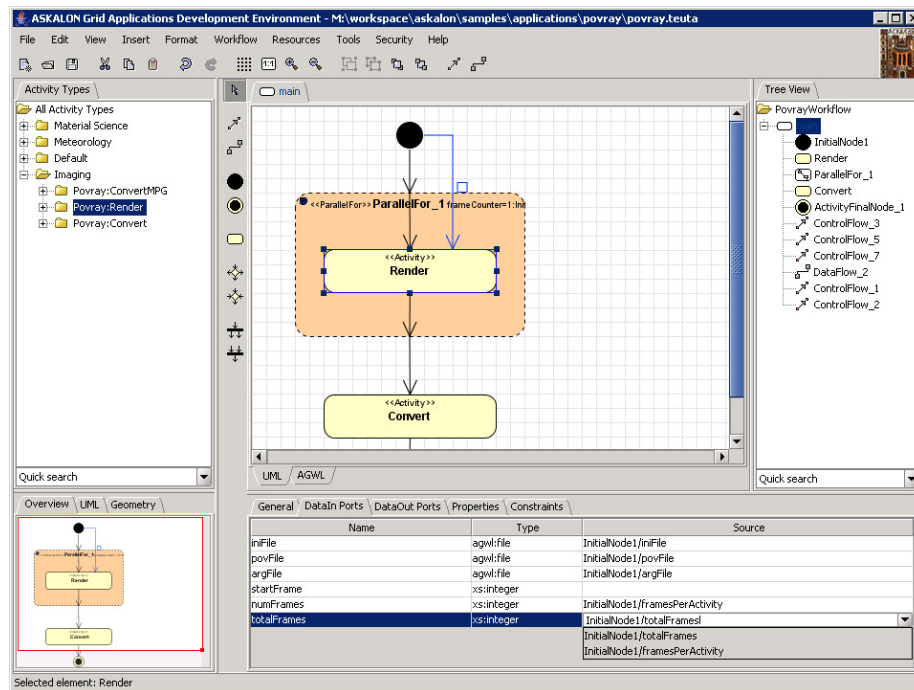


Figure 14: The povray Workflow

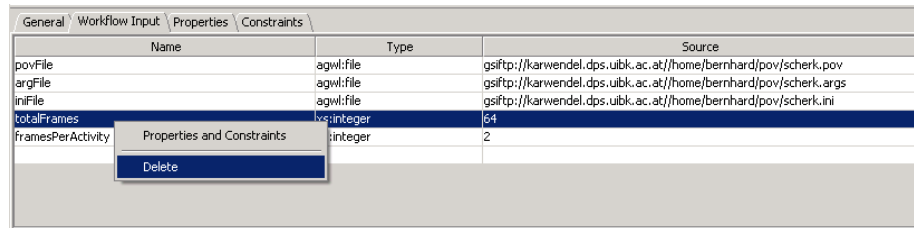
#### 7.4.1 Create Workflow Initial Node and Final Node

1. Create the InitialNode by click the corresponding button on the UML toolbar
2. Select the created InitialNode and set the workflow input through the properties panel (Figure 15).

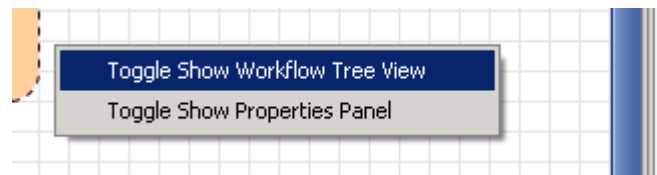
General   Workflow Input   Properties   Constraints			
Name	Type	Source	
povFile	agwl:file	gsiftp://karwendel.dps.uibk.ac.at/home/bernhard/pov/scherk.pov	
argFile	agwl:file	gsiftp://karwendel.dps.uibk.ac.at/home/bernhard/pov/scherk.args	
inFile	agwl:file	gsiftp://karwendel.dps.uibk.ac.at/home/bernhard/pov/scherk.ini	
totalFrames	xs:integer	64	
framesPerActivity	xs:integer	2	

Figure 15: Workflow Input

- To add a new input data port (a new row in the table), just input the data port name in the first column of the last line, a new empty line will be added after the this line
- To delete an existing data port, use the context menu:



- To show or hide the properties panel (or workflow tree view), use the context menu in the workflow diagram:



#### 7.4.2 Create Atomic Activities

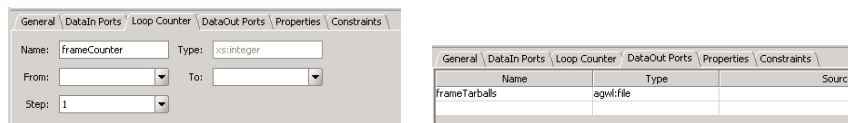
To create an atomic activity, drag an existing activity type from the activity types panel, and drop it into the workflow diagram. In this case, drag and drop the activity type Convert (Imaging domain) into the workflow diagram. Select the created Convert activity, you will see that in the properties panel, the names and types of the DataIn/DataOut ports are already there (Figure 16): this information comes from the GLARE service. It is not allowed to change them here (To change them, you need to go to GLARE, see Chapter 6). The only thing left is the Source column, which is empty so far. We will fill them in section 7.4.5 Create Data Flows.

General   DataIn Ports   DataOut Ports   Properties   Constraints		
Name	Type	
frameTarballs	agwl:collection	
outFileName	xs:string	

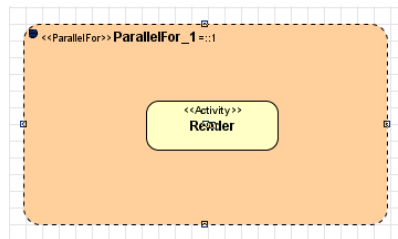
Figure 16: Automatically created DataIn ports

### 7.4.3 Create Compound Activities

In the povray workflow, another activity *Render* needs to be executed in a parallel loop. For this, we create a *ParallelFor* loop by selecting *Insert — ParallelFor* from the menu bar. To set the loop counter and data ports, select the created *ParallelFor* loop node, set the loop counter name in the *LoopCounter* tab and data output port names in the *DataOut* Ports tab in the properties panel. We do not have data input ports other than the loop counter for the *ParallelFor* loop in this workflow application.



- To put the activity *Render* into the loop body of this *ParallelFor* loop, drag the *Render* activity type from the activity types panel and drop it into the workflow diagram to create the activity *Render*. Then drag and drop the created activity *Render* into the *ParallelFor* loop node. Now, if you move the *ParallelFor* loop node, the activity *Render* should be moved along with it.



- To get rid of activities from the loop body, just drag them outside of the *ParallelFor* loop node or delete them.
- To create *For*/*ForEach*/*ParallelForEach* compound activities, the steps are similar.

For a complete reference of the graphical notation of all AGWL activities, please refer to Appendix A: UML Representations of AGWL Activity.

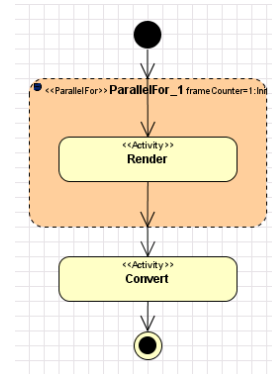
When the activities are in place, they need to be connected with control flows and data flows.

#### 7.4.4 Create Control Flows

To create control flows, select the Control Flow button on the UML toolbar, click and drag with your mouse from one node to another node.

For compound activities like `ParallelFor`, there are four connection points on the four sides: top-center, bottom-center, left-middle, and right-middle. In this case, connecting the top-center point to the activity `Render` means when the `ParallelFor` gets invoked, the control flow is handed over to the activity `Render`, and connecting the activity `Render` to the bottom-center point means when the activity `Render` finishes, it hands the control flow back to the `ParallelFor`.

- To show all connection point on all nodes, hold down the ALT key

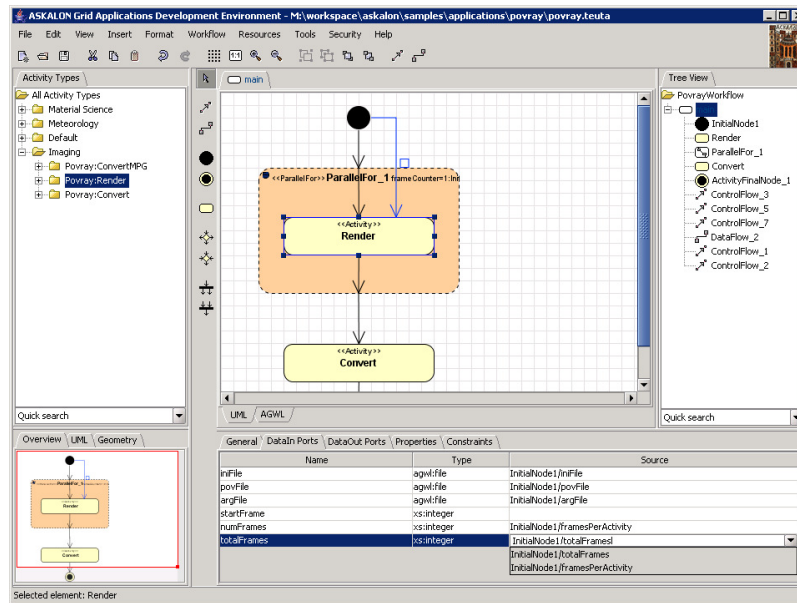


#### 7.4.5 Create Data Flows

To create data flows, select the Data Flow button on the UML toolbar, and drag with your mouse from the data flow source node to the data flow destination node.

For example, because the activity `Render` needs to consume some data specified in the workflow input (associated with the `InitialNode`), let's connect the `InitialNode` and the activity `Render` with a data flow edge, then select the activity `Render`, go to the properties panel and set the source of each data port through the combo box. If the source value is a constant value like "3", just type it in the corresponding field of the table directly.

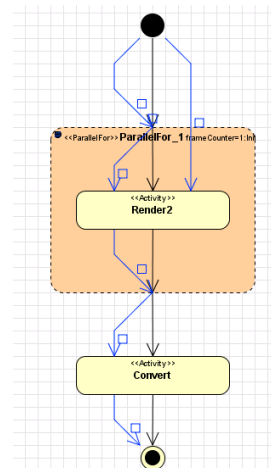
- To add a bend point on the edge, make sure the Marquee selection button in the UML toolbar is selected, then click any point on the edge while holding SHIFT



In the similar way, you can create the other data flow edges and set the corresponding source value for each data port.

It is not allowed to directly connect the activity `Render` and the activity `Convert` with a data flow edge because the activity `Render` is an inner activity of the compound activity `ParallelFor` and there will be multiple instances of the activity `Render` running at the same time. It would not be clear if the output data of all instances of the activity `Render` should be transferred to the activity `Convert`, or just one of them should be transferred.

Our solution for this is to redirect the output data ports of the activity `Render` to the output data ports of the activity `parallelFor`, which should have a data type of `agwl:collection`. Later, the collection can be used as a whole or individual elements in the collection can be used by specifying AGWL constraint element-index for the sink data port. In the povray workflow, the entire collection is consumed by the activity `Convert`.





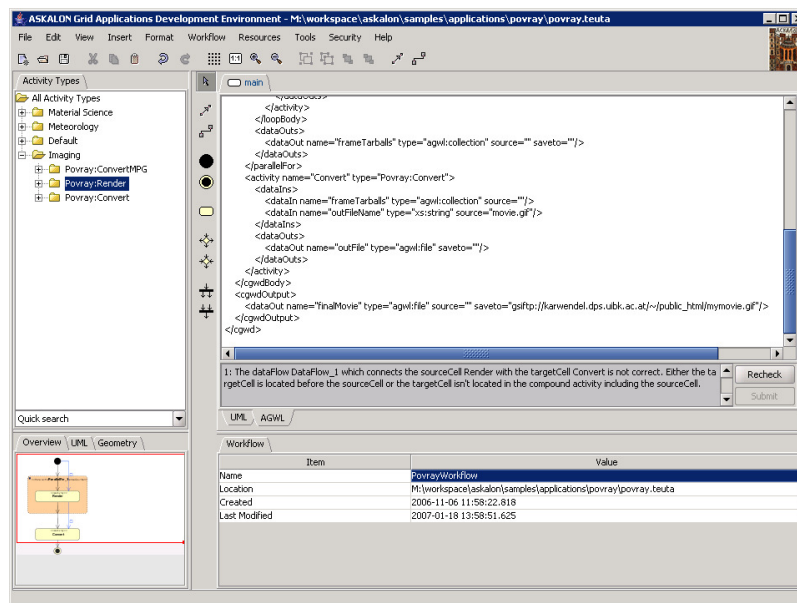
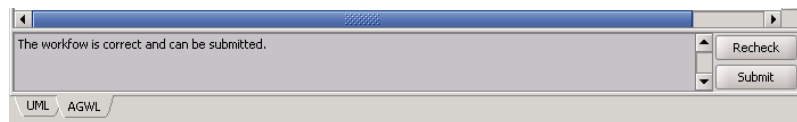


Figure 17: Workflow checking

By the way, when you click the “AGWL” tab (Figure 17), the internal model checking component model will traverse the workflow model, generate the corresponding AGWL representation of the workflow, and tell you if some errors are found.



If you see “The workflow is correct and can be submitted” when you click AGWL tab, your workflow is ready to be submitted for execution in the Grid.

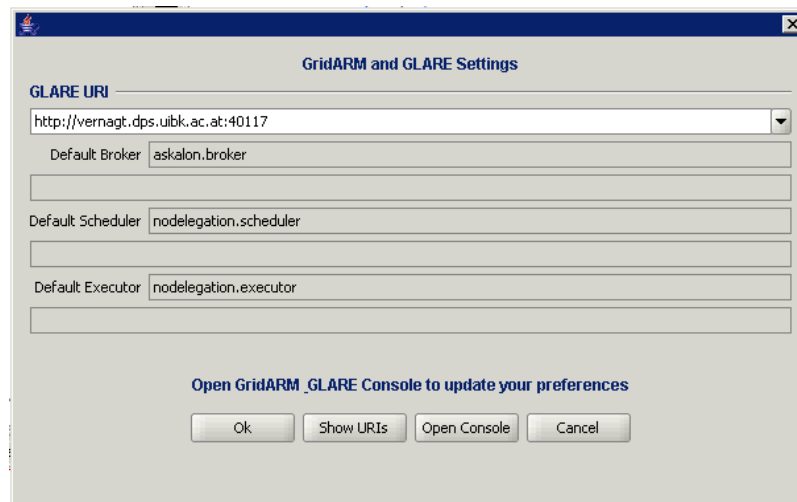


Figure 18: The GridARM and GLARE Settings dialog

## 8 Execute your Workflow

We have defined the workflow. We can execute it now.

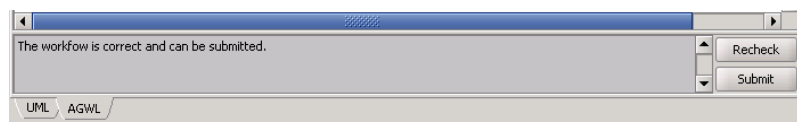
### 8.1 Setup the Environment

If you have not read all previous chapters and want to execute your workflow, you will need to enter the URI of the GridARM/GLARE service into the dialog that comes up after choosing Tools — Configure GLARE... in the Menu of the Workflow Composition Tool as described in chapter 5.1.4.

Click the OK Button to save the settings. This will automatically configure the URIs of the default Broker, Scheduler and Executor Services stored in GLARE.

### 8.2 Start the Workflow Execution

To execute your workflow, make sure that you have correctly set the URIs to the Broker, the Scheduler and the Executor Service (see Section 6.1), then click the “Submit” Button in the AGWL tab or select Workflow — Submit from the menu.



The ASKALON frontend now switches from editing-mode to execution-mode, which helps you tracking the workflow execution. (To switch back to editing mode, click the “Edit Mode” button.)

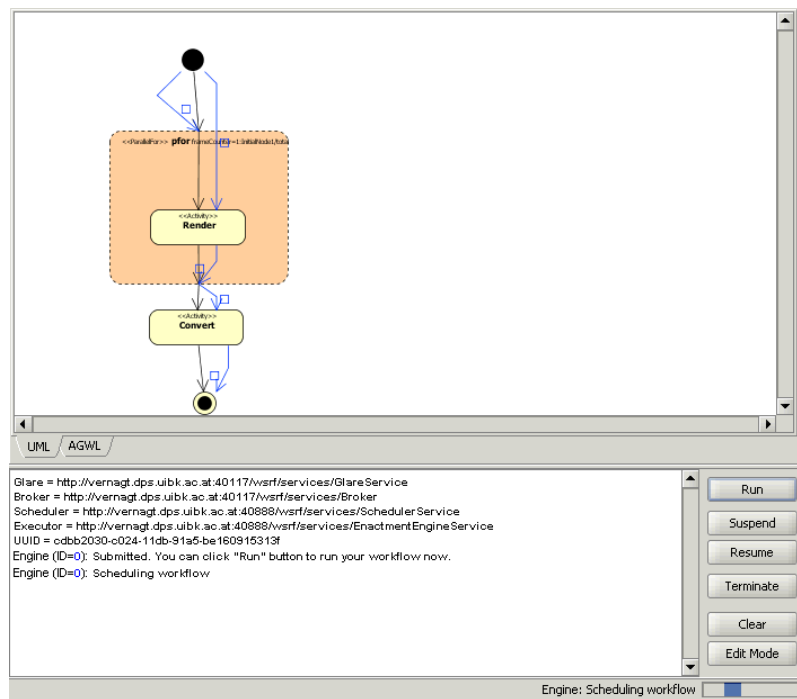
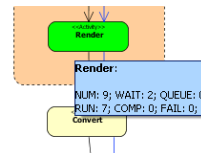


Figure 19: Execution Mode of the Workflow Editor

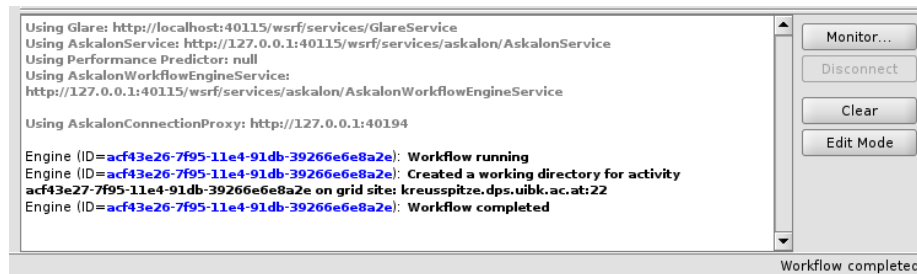
To start the execution of the workflow, click the Run Button. While the workflow is running, you will be informed about the status of the different activities by a changing background color (for atomic activities) and tooltip information upon mouse-over (for complex activities). The following picture shows that the parallelFor activity that contains the Render activity has 9 iterations, 2 of them are waiting to be executed, none are queued, 7 are currently running on the grid, none of them has yet completed and none of them failed.

If you want more information than this simple view can give you, read Chapter 9 Monitor Your Workflow. After the workflow execution has finished, every atomic activity will have turned blue (showing that they have successfully completed) and the status bar as well as the message window will inform you about it. Any error or failure during the execution will also be shown in the message window.

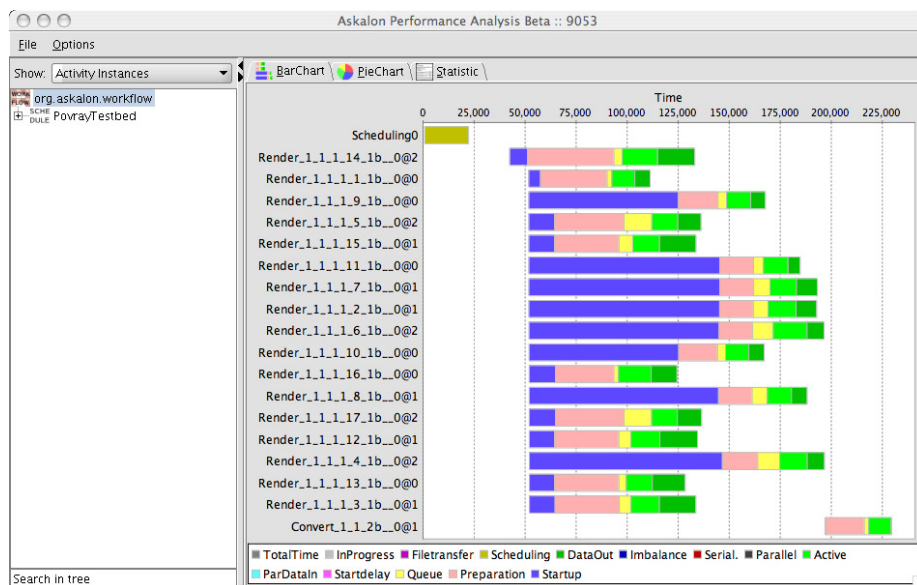


## 9 Monitor Your Workflow

When your workflow is running you will see the following view:

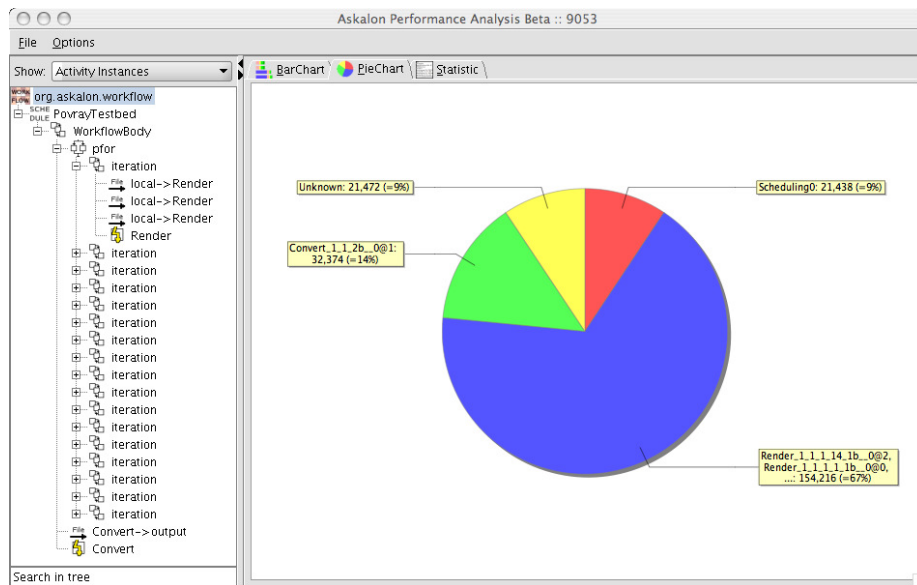


After the workflow got an ID assigned you have the following options: suspend, terminate or monitor the execution. With the “Monitor...” button you start the monitoring tool, which will give you a more detailed view of the workflow-execution than the informations integrated into the UML.



On the left hand side you can see a tree representation of the execution where the first level represents the schedules of the workflow. By selecting nodes in this tree all the views will be reduced to show only informations of the selected parts. On the right hand side you have the choice of three views of the execution (Tabs):

- Stacked Bar Chart
- Pie Chart
- Text Statistics



Additional you have the options to advance the detail level of the Charts (Options Menu):

- Show Total Time
- Show File Transfers

You have the options to simplify the diagram via the combo box selections (Combo box):

- Activity Instances (default)
- Workflow Regions (will show parallel sections as one unit)
- File transfers Only
- Specific Grid Site (only show task which run on a specific grid site)

The “Workflow Regions” option is especially useful if you have complex or big workflows where each parallel section may consists of hundred or more tasks. This way these will be shown as a single parallel section in the diagram. You can see some metrics already calculated and shown for these parallel sections then (parallel data in, start delay, imbalance and serialization).

If you are not used to the workflow structure in the Askalon system you can also click on a activity or a section in the UML in ASKALON’s workflow composition tool and it will automatically be selected in the workflow-tree in the monitoring tool and the connected task-informations will be shown. A similar feature can be used via the search-box under the tree-view, which will also highlight every entry in the tree containing the search-phrase you entered.

## References

- [1] T. Fahringer, R. Prodan, Rubing Duan, F. Nerieri, S. Podlipnig, Jun Qin, M. Siddiqui, Hong-Linh Truong, A. Villazon, and M. Wiecek. Askalon: A grid application development and computing environment. In *Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing, GRID '05*, pages 122–131, Washington, DC, USA, 2005. IEEE Computer Society.
- [2] T. Fahringer, Jun Qin, and S. Hainzer. Specification of grid workflow applications with agwl: an abstract grid workflow language. In *Cluster Computing and the Grid, 2005. CCGrid 2005. IEEE International Symposium on*, volume 2, pages 676–685 Vol. 2, 2005.

## A UML Representation of AGWL Activities

