



Integration of Applications in UNICORE 6

Rebecca Breu, Sandra Bergmann

Jülich Supercomputing Centre
Forschungszentrum Jülich GmbH

29th October 2008

Outline

Defining Applications

Using Applications

Developing GridBeans

GridBean Example

Applications

- ▶ Applications are defined on server side
 - ▶ Bash
 - ▶ Python
 - ▶ ...
- ▶ User only needs to know name of application (and maybe version)
- ▶ Admin configures applications on server side:
 - ▶ Path to executable
 - ▶ Parameters passed to application

The Incarnation Database

- ▶ Applications are defined in the IDB (Incarnation Database) of the XNJS
- ▶ XNJS translates abstract job descriptions into executable scripts with the help of the IDB

```
<?xml version="1.0" encoding="UTF-8"?>
<idb:IDB xmlns:idb="http://www.fz-juelich.de/unicore/xnjs/i
  <idb:IDBApplication>
    ...
  </idb:IDBApplication>

  <idb:TargetSystemProperties>
    <jSDL:Resources xmlns:jSDL="http://schemas.ggf.org/jSDL
      ...
    </jSDL:Resources>
  </idb:TargetSystemProperties>
</idb:IDB>
```

Simple Applications

Use `IDBApplication` tag to add as many application as you like:

```
<idb:IDBApplication>  
  <idb:ApplicationName>Date</idb:ApplicationName>  
  <idb:ApplicationVersion>1.0</idb:ApplicationVersion>  
  <jSDL:POSIXApplication xmlns:jSDL="http://schemas.ggf.org"  
    <jSDL:Executable>/bin/date</jSDL:Executable>  
  </jSDL:POSIXApplication>  
</idb:IDBApplication>
```

→ Application Date gets mapped to /bin/date

Applications With Parameters

Use **Argument** tags to add as many parameters as you like:

```
<idb:IDBApplication>  
  <idb:ApplicationName>LS</idb:ApplicationName>  
  <idb:ApplicationVersion>1.0</idb:ApplicationVersion>  
  <jSDL:POSIXApplication xmlns:jSDL="http://schemas.ggf.org/  
    <jSDL:Executable>/bin/ls</jSDL:Executable>  
    <jSDL:Argument>-l</jSDL:Argument>  
    <jSDL:Argument>-t</jSDL:Argument>  
  </jSDL:POSIXApplication>  
</idb:IDBApplication>
```

→ Application LS that gets mapped to `/bin/ls -l -t`

Applications With Parameters

Use question mark ? to denote conditional parameters:

```
<idb:IDBApplication>
  <idb:ApplicationName>Bash shell</idb:ApplicationName>
  <idb:ApplicationVersion>3.1.16</idb:ApplicationVersion>
  <jSDL:POSIXApplication xmlns:jSDL="http://schemas.ggf.org"
    <jSDL:Executable>/bin/bash</jSDL:Executable>
    <jSDL:Argument>-v$VERBOSE?</jSDL:Argument>
    <jSDL:Argument>$ARGUMENTS?</jSDL:Argument>
    <jSDL:Argument>$SOURCE?</jSDL:Argument>
  </jSDL:POSIXApplication>
</idb:IDBApplication>
```

Conditional arguments are only included if the according environment variables are set.

Interactive Applications

Applications that should not be submitted to the Batch System can be declared interactive:

```
<idb:IDBApplication>  
  <idb:ApplicationName>Date</idb:ApplicationName>  
  <idb:ApplicationVersion>1.0</idb:ApplicationVersion>  
  
  <idb:PreferInteractive>true</idb:PreferInteractive>  
  
  <jSDL:POSIXApplication xmlns:jSDL="http://schemas.ggf.org"  
    <jSDL:Executable>/bin/date</jSDL:Executable>  
  </jSDL:POSIXApplication>  
</idb:IDBApplication>
```

Using Applications With The UCC

Application is specified in the job file:

```
{  
  ApplicationName: Date,  
  Version: 1.0 # Optional  
}
```

With environment variables:

```
{  
  ApplicationName: Bash shell,  
  Environment: [ "SOURCE=input.sh" ],  
  Imports: [ {File: "/home/rbreu/myscript.sh",  
             To: "input.sh"} ],  
}
```

Using Applications With The Graphical Clients

GridBeans:

- ▶ Pluggable GUIs for job creation
- ▶ Equivalent to Plugins in UNICORE 5, but ...
 - ▶ support new Grid standards (JSDL, ...)
 - ▶ independent from UNICORE clients
 - ▶ easily distributed via GridBean Download Service
- ▶ Based on high level Grid API
- ▶ GridBeans specialising on one application allow for specialised GUI components for:
 - ▶ input parameters and input files
 - ▶ visualising the output

PovRay GridBean

PovRay GridBean for use with povray application

Navigator Grid Browser Input for POV-Ray1

Name:

Width: Height: Anti Aliasing:

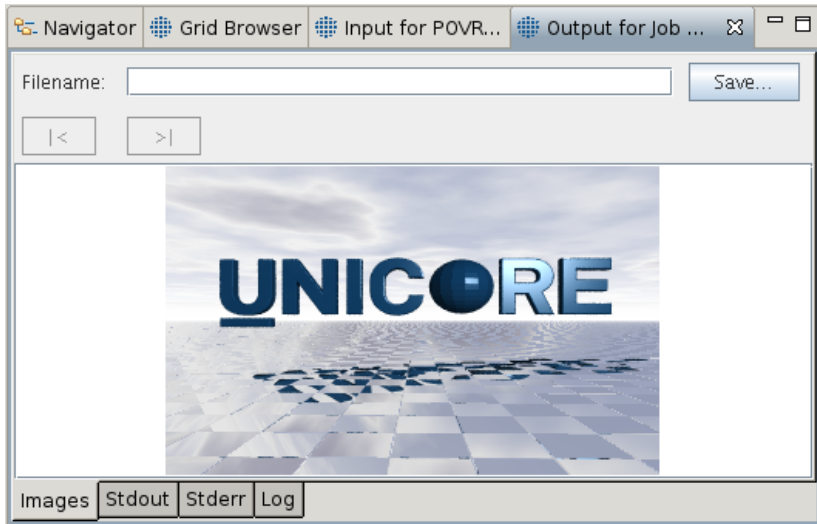
Animate Initial Frame: Final Frame:

Subset Start Frame: Subset End Frame:

Initial Clock: Final Clock:

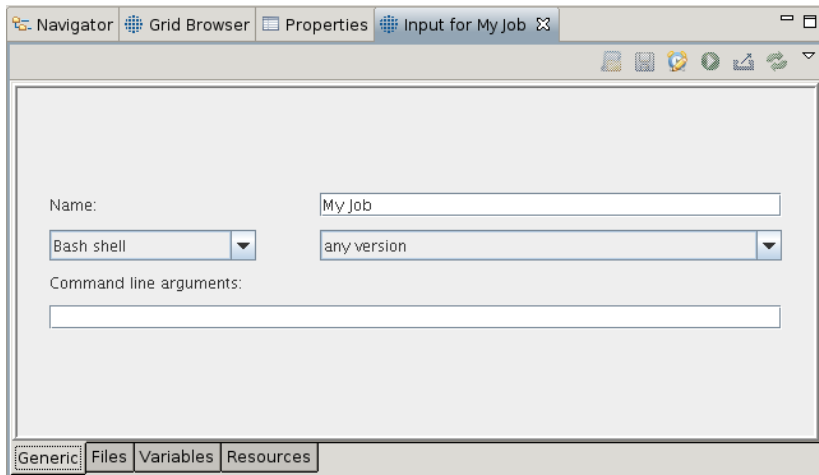
Options Source File Info Files Variables Resources

PovRay GridBean



Generic GridBean

Generic GridBean for use with arbitrary applications



The screenshot shows a software window titled "Input for My Job" with a standard toolbar. The main area contains the following configuration fields:

- Name:** A text input field containing "My Job".
- Shell:** A dropdown menu currently showing "Bash shell".
- Version:** A dropdown menu currently showing "any version".
- Command line arguments:** An empty text input field.

At the bottom of the window, there is a tabbed interface with four tabs: "Generic" (which is selected and highlighted with a dashed border), "Files", "Variables", and "Resources".

Generic GridBean

The screenshot shows the 'Input for My Job' window in the Generic GridBean application. The window has a title bar with tabs for 'Navigator', 'Grid Browser', 'Properties', and 'Input for My Job'. Below the title bar is a toolbar with icons for file operations and refresh. The main content area is divided into two sections: 'Imports to job directory:' and 'Exports from job directory:'. The 'Imports' section contains a table with one row: 'INPUT 1 Local_File /home/rbreu/input.sh input.sh'. The 'Exports' section contains a table with two rows: 'STANDARD_ERROR stderr Workflow_File stderr' and 'STANDARD_OUT stdout Workflow_File stdout'. At the bottom of the window, there are four tabs: 'Generic', 'Files', 'Variables', and 'Resources'.

Imports to job directory:

Name	Source Type	Source File(s)	File(s) in Job Directory
INPUT 1	Local_File	/home/rbreu/input.sh	input.sh

Exports from job directory:

Name	File(s) in Job Directory	Destination Type	File(s) at Destination / File ID
STANDARD_ERROR	stderr	Workflow_File	stderr
STANDARD_OUT	stdout	Workflow_File	stdout

Generic Files Variables Resources

Generic GridBean

The screenshot shows the Generic GridBean application window. The title bar contains tabs for 'Navigator', 'Grid Browser', 'Properties', and 'Input for My Job'. Below the title bar is a toolbar with icons for file operations and execution. The main area is a table with the following data:

Name	Source Type	Value/Reference
ARGUMENTS	Fixed_Value	
SOURCE	Fixed_Value	input.sh

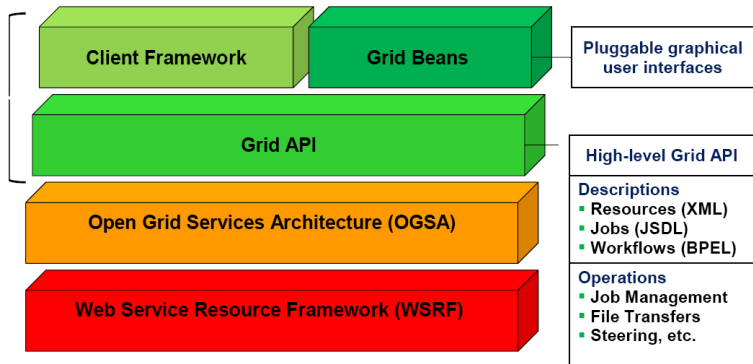
At the bottom of the window, there are four tabs: 'Generic', 'Files', 'Variables', and 'Resources'.

Grid Programming Environment

Grid Programming Environment (GPE) developed by Intel

- ▶ Graphical user interfaces and interoperable GridBeans for application development
- ▶ High-level API for programming Grid clients
- ▶ Language independent definition
- ▶ Java reference implementation
- ▶ Supports UNICORE-based and Globus Toolkit infrastructures

Architecture

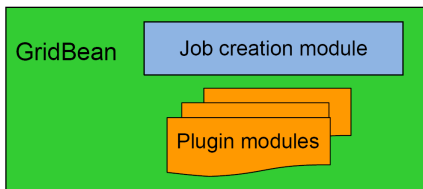


XML = Extensible Markup Language
JSDL = Job Submission Definition Language
BPEL = Business Process Execution Language

Structure

Nice separation of the data model and its graphical views

- ▶ Job creation module (**GridBean Model**)
 - ▶ Defines internal data representation
 - ▶ Translated to actual job specification (JSDL)
- ▶ One or more plugin modules
 - ▶ GUIs for getting user input and presenting job output
 - ▶ Differ depending on environment (e.g. application/portal clients)



GridBean Model, GPE Job

GridBean Model:

- ▶ Maintains the actual state of job settings
- ▶ Stores internal data in pairs: key + value
- ▶ Model must implement `IGridBeanModel` interface
- ▶ Superclass `AbstractGridBean` implements `IGridBeanModel` methods for storing named GridBean parameters:
`get(QName key)`, `set(QName key, Object Value)`

GridBean Model creates `GPE Job` which specifies:

- ▶ Resource requirements
- ▶ Files to stage-in/out
- ▶ Application name and version
- ▶ Named application parameters (fields/options/variations)
- ▶ Application working directory

IGridBean Interface

Model must implement IGridBean interface methods:

- ▶ `setupJobDefinition(Job job)` generates JSDL document from the data in the model
- ▶ `parseJobDefinition(Job job)` transfers values from the JSDL back into the model
- ▶ `getInput/OutputParameters()` used for generating job imports/exports

Plugin Interfaces/Superclasses

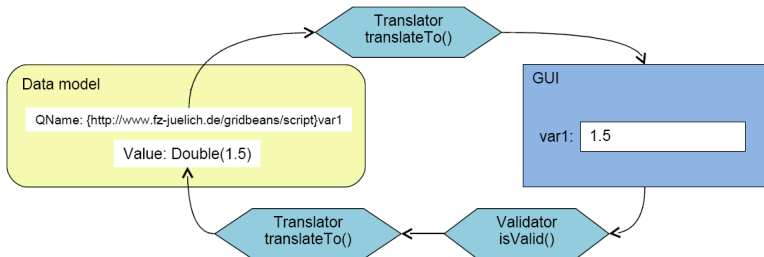
- ▶ Swing GridBean plugins implement IGridBeanPlugin interface or use the superclass GridBeanPlugin
- ▶ Portlet plugins are subclasses of PortletPlugin
- ▶ plugins provide
 - ▶ One or more input panel(s)
 - ▶ One or more output panel(s)

Swing Plugin Panel

- ▶ Input panels: preparation and validation of user input
- ▶ Output panels: showing output files (e.g. images)
- ▶ Input and output panels must implement IGridBeanPanel interface
- ▶ Superclass GridBeanPanel implements IGridBeanPanel methods:
 - ▶ `Component getComponent()`
 - ▶ `String getName()`
 - ▶ `void validate(ErrorSet errors)`

Swing Plugin Panel

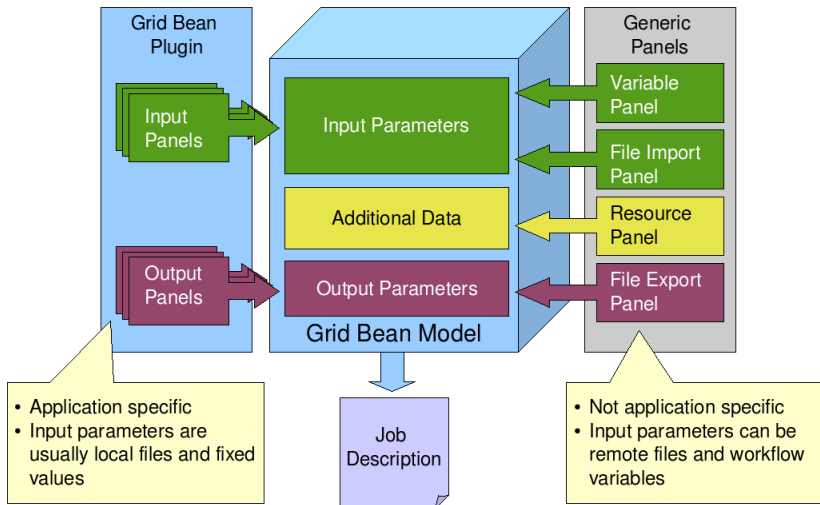
- ▶ Each graphical component (e.g. JTextField) is bound to a data model element (key + value)
- ▶ Validators check input before writing to the model
- ▶ Translators may change data format



Input and Output Parameters

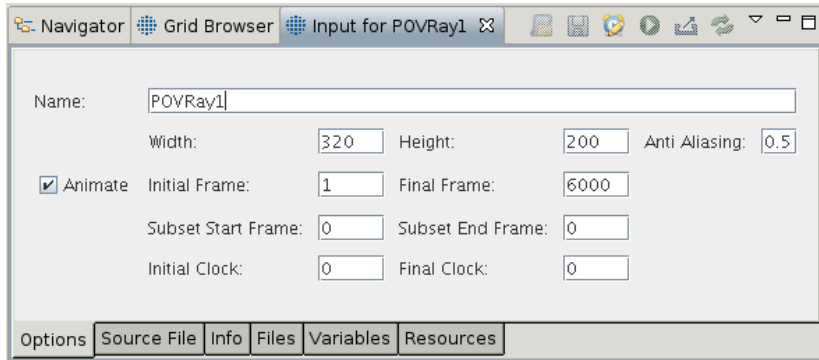
- ▶ Input parameters declare input files, obtained from a file storage or other jobs
- ▶ Output parameters declare output files to be transferred to a file storage or other jobs
- ▶ Four different types of input and output parameters:
 - ▶ GPE File (file at a local or remote location)
 - ▶ File Set (a set of files, wildcards allowed)
 - ▶ Environment variables
 - ▶ Resource Parameters

Input and Output Parameters



PovRay GridBean

PovRay GridBean for use with povray application



The screenshot shows the PovRay GridBean configuration window. The window title is "Input for POV-Ray1". The interface includes a "Name" field with the value "POV-Ray1". Below this are several input fields for rendering parameters: "Width" (320), "Height" (200), and "Anti Aliasing" (0.5). There is a checked "Animate" checkbox followed by "Initial Frame" (1) and "Final Frame" (6000). Below these are "Subset Start Frame" (0) and "Subset End Frame" (0), and "Initial Clock" (0) and "Final Clock" (0). At the bottom, there is a tabbed interface with tabs for "Options", "Source File", "Info", "Files", "Variables", and "Resources".

Name:	POV-Ray1				
Width:	320	Height:	200	Anti Aliasing:	0.5
<input checked="" type="checkbox"/> Animate	Initial Frame:	1	Final Frame:	6000	
	Subset Start Frame:	0	Subset End Frame:	0	
	Initial Clock:	0	Final Clock:	0	

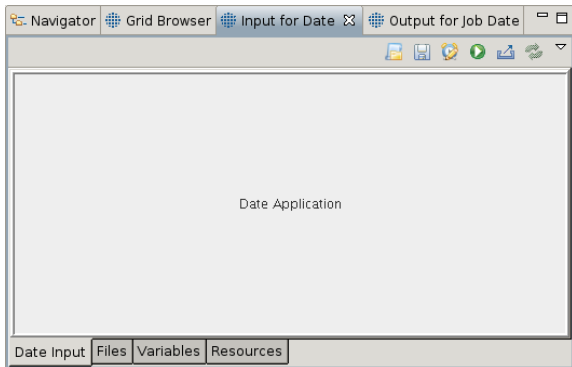
Further Information

- ▶ GridBean Developer Guide on GPE4GTK Project web page, example GridBeans and their source code:
<http://gpe4gtk.sourceforge.net>
- ▶ gpe4unicore and GridBeans sources
<http://sourceforge.net/projects/unicore>

Example 1

Example 1: DateGridBean with no input parameters:

- ▶ gridbean.xml
- ▶ DateGridBean.java
- ▶ DatePlugin.java
- ▶ DateInputPanel.java



gridbean.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<gb:GridBeansInfo xmlns:gb="http://gpe.intel.com/gridbeans"
  <gb:Name>Date</gb:Name>
  <gb:Author>Sandra Bergmann</gb:Author>
  <gb:Version>1.0</gb:Version>
  <gb:Application>Date</gb:Application>
  <gb:Description>Date GridBean</gb:Description>
  <gb:PluginVersion>1.0</gb:PluginVersion>
  <gb:GridBean>
    de.fzj.gpe.gridbeans.date.DateGridBean
  </gb:GridBean>
  <gb:Plugin type="gb:Swing">
    de.fzj.gpe.gridbeans.date.plugin.DatePlugin
  </gb:Plugin>
  <gb:ApplicationName>Date</gb:ApplicationName>
  <gb:ApplicationVersion>1.0</gb:ApplicationVersion>
</gb:GridBeansInfo>
```

DateGridBean.java

```
public class DateGridBean extends AbstractGridBean {  
  
    private static final long serialVersionUID = 1L;  
  
    public DateGridBean() { set(JOBNAME, "Date"); }  
  
    public void setupJobDefinition(Job job) throws GridBeanException {  
        super.setupJobDefinition(job);  
  
        if (job instanceof GPEJob) {  
            GPEJob gpeJob = (GPEJob) job;  
            gpeJob.setApplicationName("Date");  
            gpeJob.setApplicationVersion("1.0");  
            gpeJob.setWorkingDirectory(  
                GPEConstants.JobManagement.TEMPORARY_DIR_NAME);  
            gpeJob.setId("Date");  
        }  
        else {  
            throw new GridBeanException("Invalid job type.");  
        }  
    }  
  
    public void parseJobDefinition(Job job) throws GridBeanException {  
        super.parseJobDefinition(job);  
    }  
  
    public String getName() { return "Date"; }  
}
```

DatePlugin.java

```
public class DatePlugin extends GridBeanPlugin {  
  
    public void initialize(Client client, INode node) {  
        super.initialize(client, node);  
        addInputPanel(  
            new DateInputPanel(client, getParameter()));  
    }  
}
```

DateInputPanel.java

```
public class DateInputPanel extends GridBeanPanel {
    private static final long serialVersionUID = 1L;

    public DateInputPanel(Client client, INode node) {
        super(client, "Date Input", node);
        try {
            buildComponents();
        } catch (DataSetException e) {
            client.showException("Failure creating panel.", e);
        }
    }

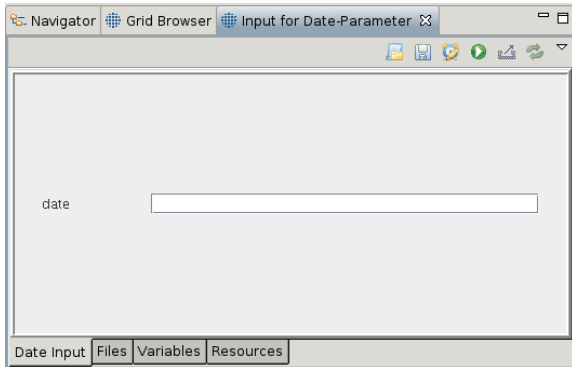
    private void buildComponents() throws DataSetException {
        setLayout(new BorderLayout());
        setBorder(BorderFactory.createCompoundBorder(
            BorderFactory.createRaisedBevelBorder(),
            BorderFactory.createLoweredBevelBorder()));

        JLabel label= new JLabel("Date Application");
        label.setHorizontalAlignment(JLabel.CENTER);
        label.setVerticalAlignment(JLabel.CENTER);
        add(label, BorderLayout.CENTER);
    }
}
```

Example 2

Example 2: DateGridBean with one input parameters:

- ▶ gridbean.xml
- ▶ DateGridBean.java
- ▶ DatePlugin.java
- ▶ DateInputPanel.java



DateGridBean.java

Setup environment variables in the DateGridBean class:

```
public DateGridBean() {
    set(JOBNAME, "Date-Parameter");

    EnvironmentVariableParameterValue arguments =
        new EnvironmentVariableParameterValue("");

    getInputParameters().add(new GridBeanParameter(
        ARGUMENTS,
        GridBeanParameterType.ENVIRONMENT_VARIABLE, true));

    set(ARGUMENTS, arguments);
}
```

DateInputPanel.java

Add text input field and link it with the corresponding attribute of the GridBean Model:

```
JTextField nameTextField = new JTextField(30);  
add(argsTextField, BorderLayout.CENTER);  
  
linkTextField(DateGridBean.ARGUMENTS, argsTextField);  
  
setValueTranslator(DateGridBean.ARGUMENTS,  
                    StringValueTranslator.getInstance());
```

IDB

The according IDB entry:

```
<idb:IDBApplication>  
  <idb:ApplicationName>Date</idb:ApplicationName>  
  <idb:ApplicationVersion>1.0</idb:ApplicationVersion>  
  <jSDL:POSIXApplication xmlns:jSDL="http://schemas.ggf.org"  
    <jSDL:Executable>/bin/date</jSDL:Executable>  
    <jSDL:Argument>${ARGUMENTS?}</jSDL:Argument>  
  </jSDL:POSIXApplication>  
</idb:IDBApplication>
```