



**pallas**

Member of the ExperTeam Group

## **The UNICOREpro Client**

### **Programming Client Plug-Ins**

Ralf Ratering  
Pallas GmbH  
Hermülheimer Straße 10  
50321 Brühl, Germany

[ralf.ratering@pallas.com](mailto:ralf.ratering@pallas.com)  
<http://www.unicorepro.com>

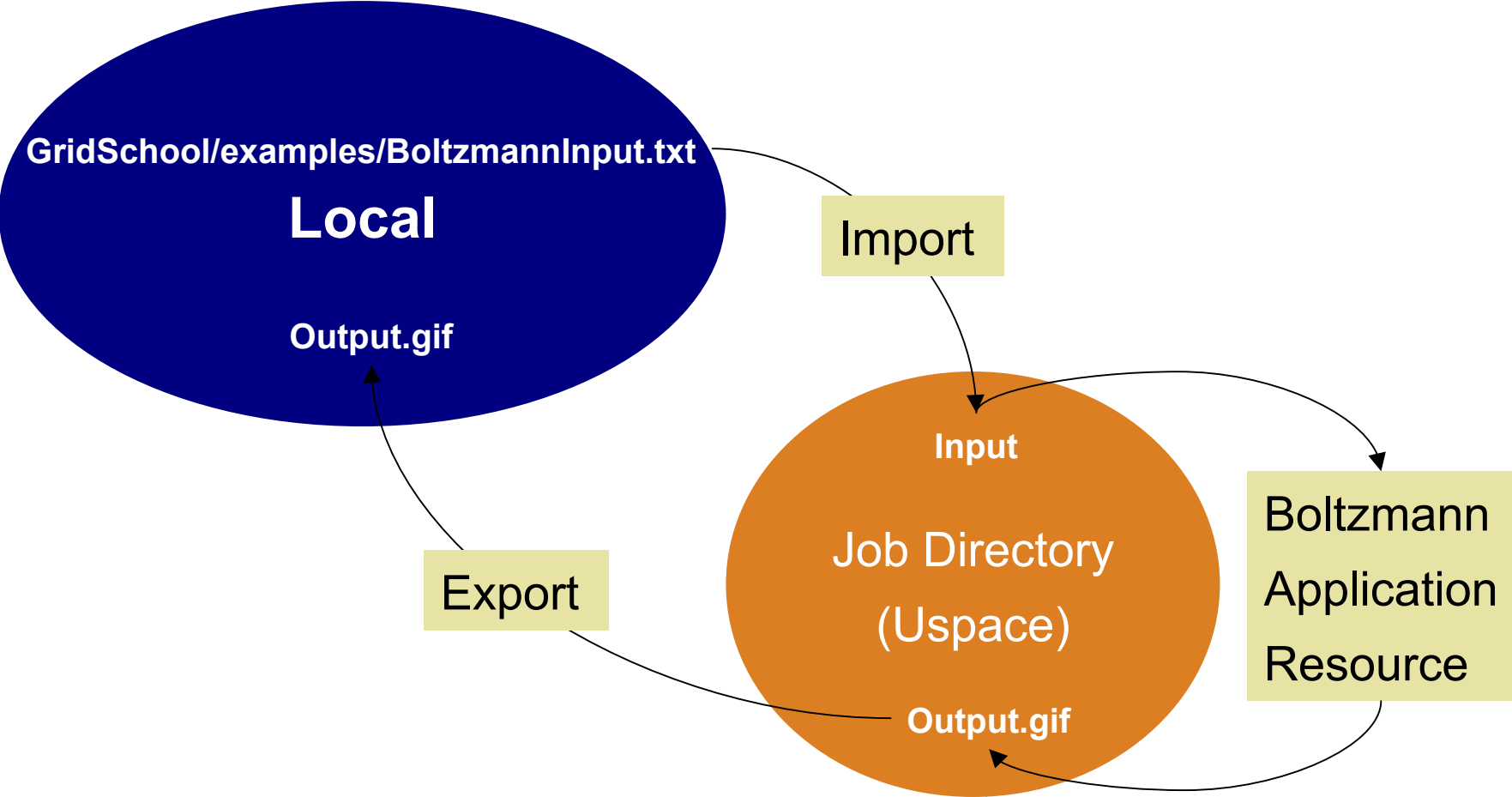


- Scenario: A site wants to make new application available on the Grid
- Example: Lattice Boltzmann
  - Simulation of fluent mixing
  - Output: a gif animation
  - Intermediate sample files are generated
  - A control file can change parameters while application is executing
- Integrate Boltzmann application into Client GUI with a Plugin!



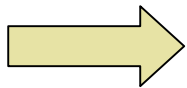
- **Step 1**
  - Use Command Task to run application
  - Specify input and output files in import/export panels
- **Step 2**
  - Write a specialized Boltzmann plugin task
- **Step 3**
  - Edit and automatically send the input file
  - Automatically set output file export
- **Step 4**
  - Get sample files while application is executing
  - Visualize sample files in outcome area

# Step1: Executing a Command Task





- Disadvantages of Command task
  - Input file has to be edited outside Client
  - Imports and Exports have to be specified manually
  - No integrated GUI for parameters
  - Results have to be visualized outside client
  - No additional functionality possible (sample and control files)



Write a specialized Boltzmann Plugin Task!

# Task Plugins



- Add a new type of task to the Client GUI
- New task can be integrated into complex jobs
- Application support: CPMD, Fluent, Gaussian, etc.

The screenshot displays the UNICOREpro Client interface. On the left, a vertical toolbar contains various 'Add' options. A yellow box labeled 'Add task item' points to the 'Add POV-Ray' option. Below it, a yellow box labeled 'Settings item' points to the 'POV-Ray Defaults' option in the 'Settings' menu. A yellow box labeled 'Icon' points to the 'POV-Ray' icon in the 'Task Dependencies' panel. A yellow box labeled 'Plugin info' points to the 'UNICOREpro: Plugin Info' dialog box, which lists 'POV-Ray Plugin 1.0' with author 'Ralf Ratering' and copyright '(C)2003 Pallas GmbH'. The main window shows a 'Job' configuration with tabs for 'Dependencies', 'Resources', and 'Special Settings'. The 'Task Dependencies' panel shows a 'POV-Ray' task icon. The status bar at the bottom indicates the user 'ralf.ratering' and the job status 'Job not saved yet.'.

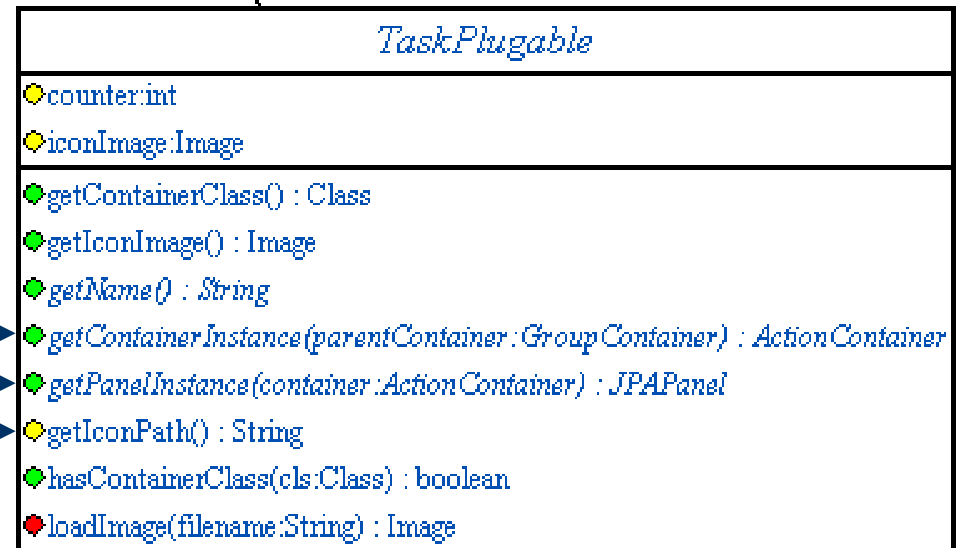
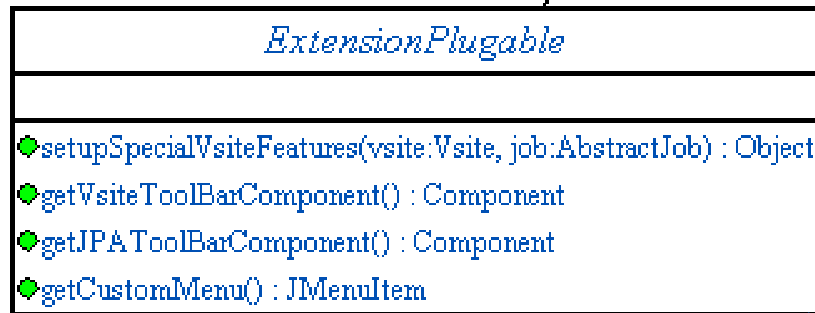
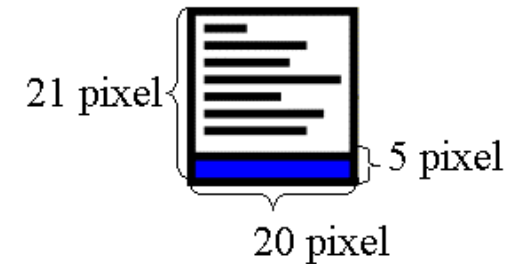
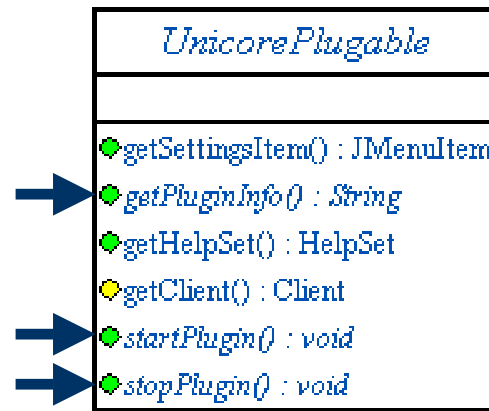


- Implement 3 Classes
  - Main plugin class
  - Plugin Container
  - JPAPanel
- Build a Jar Archive named „\*Plugin.jar“
- Sign the Jar with your Certificate

# Main Plugin Class



- Start and stop plugin
- Manage plugin objects



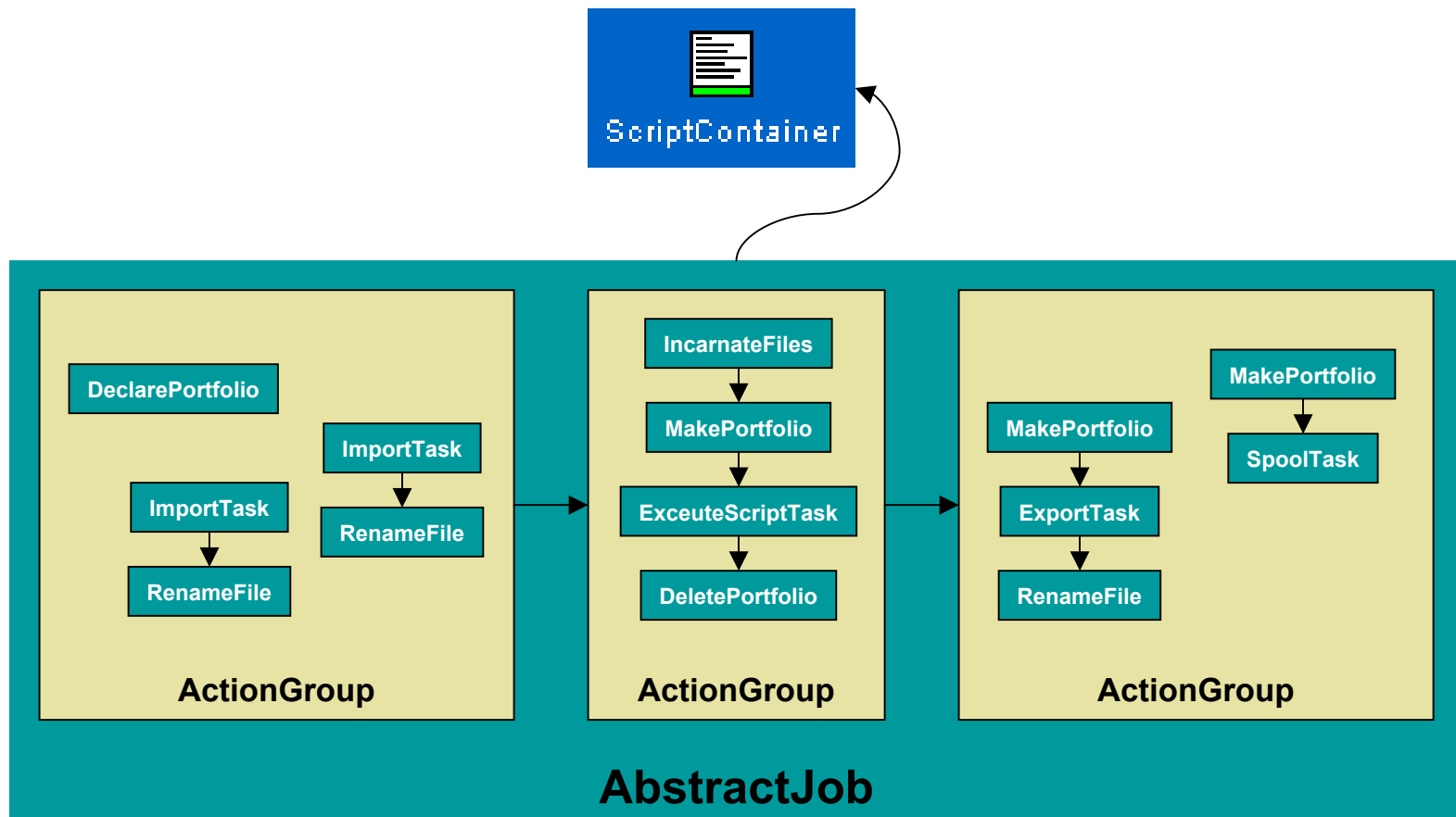




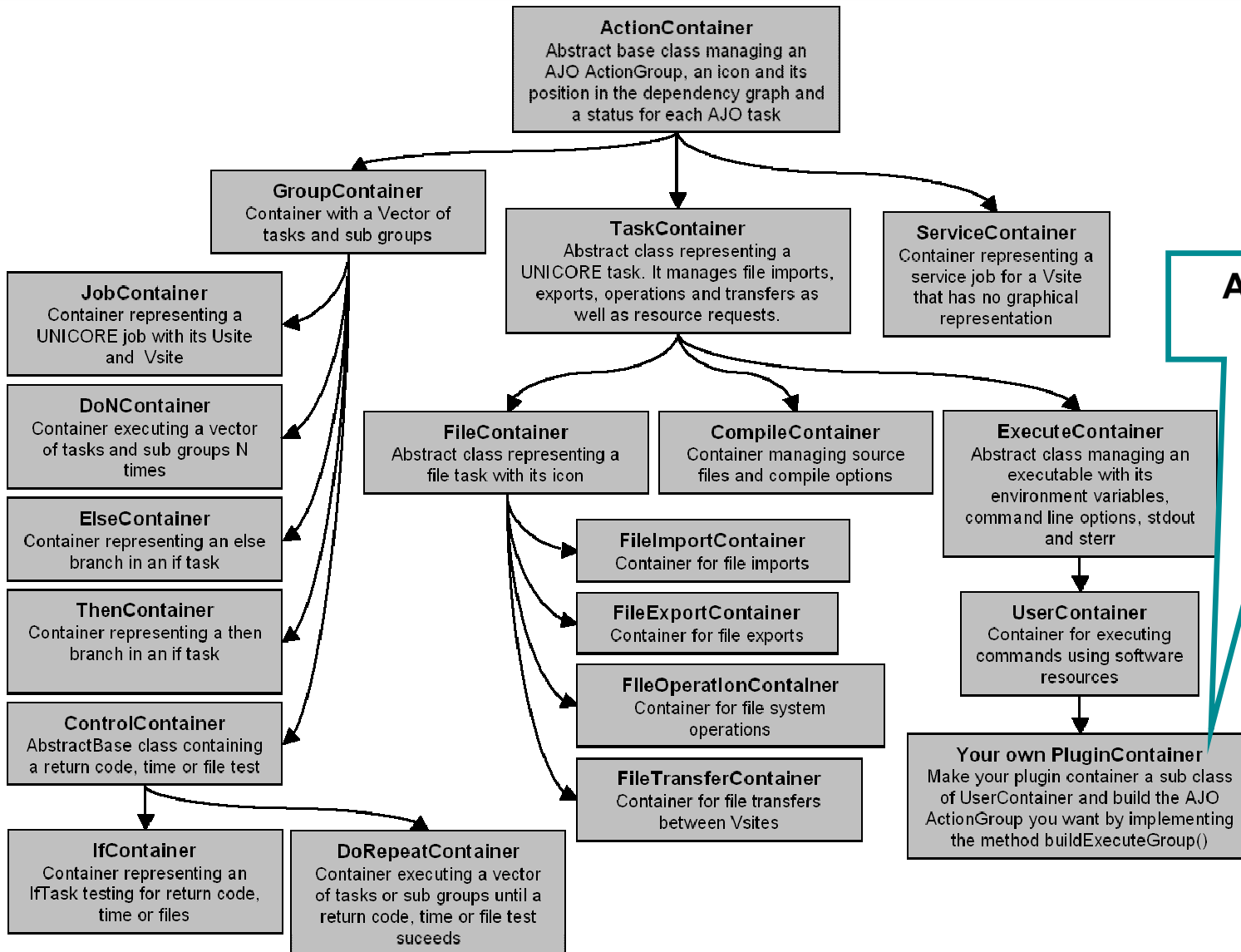
- Build **Abstract Job Object** (AJO)
- Manage imports, exports and execution
- Hold parameters
- Keep status
- Check errors



- AJO is the low-level „UNICORE language“
- Client containers encapsulate complex AJOs



# Container Hierarchy



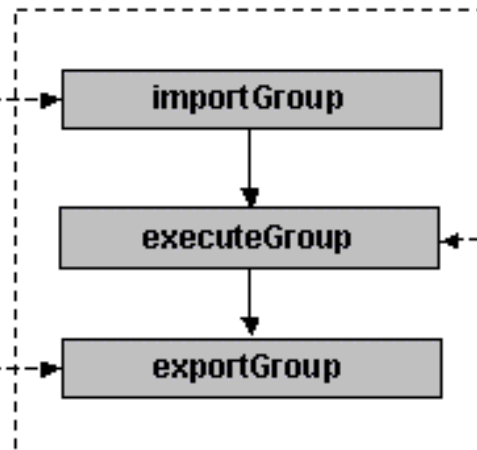
**Add your own container**



## ActionGroup in TaskContainer

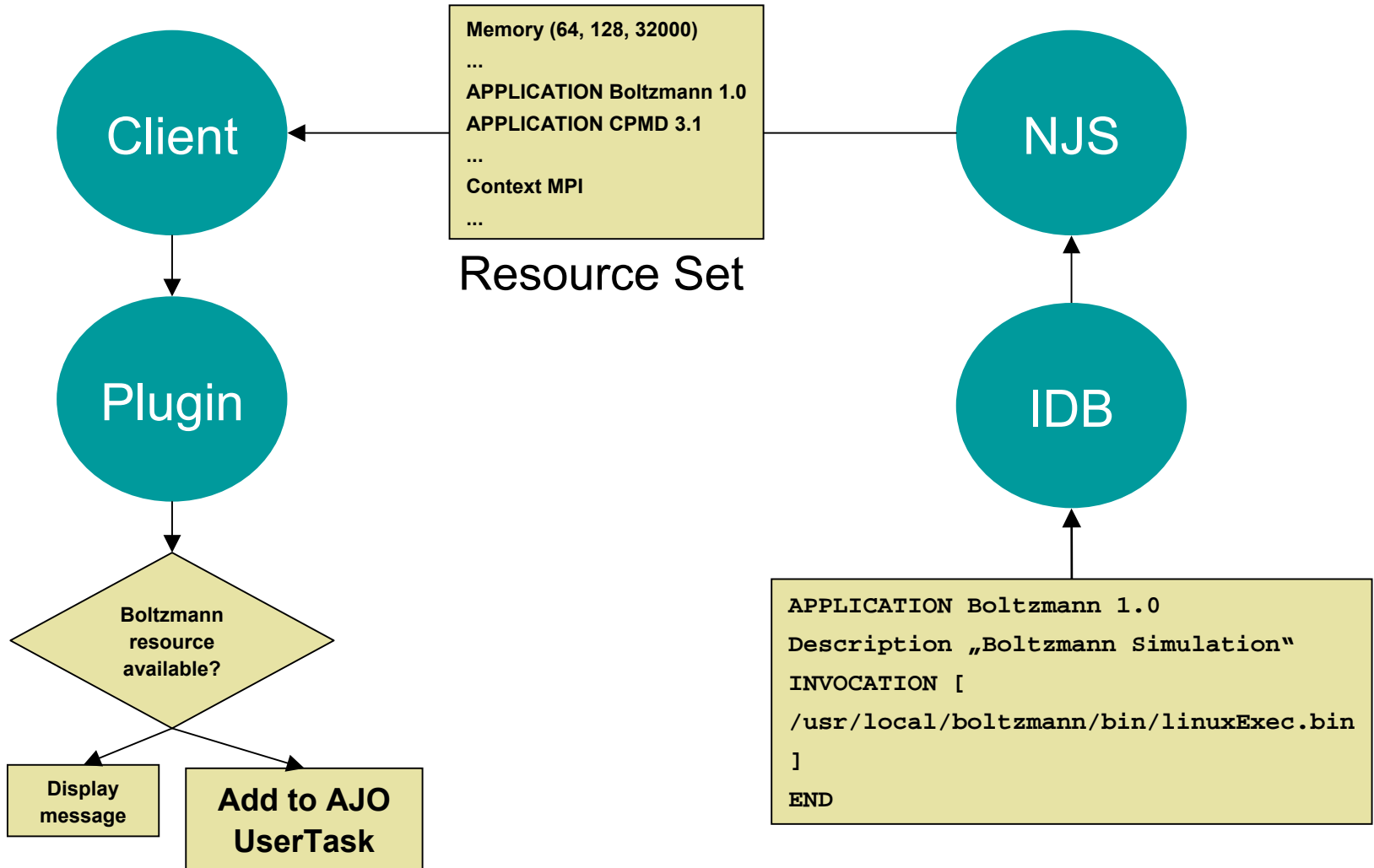
**You use**  
`setFileImports()`  
`addFileImport()`

**You use**  
`setFileExports()`  
`addFileExport()`



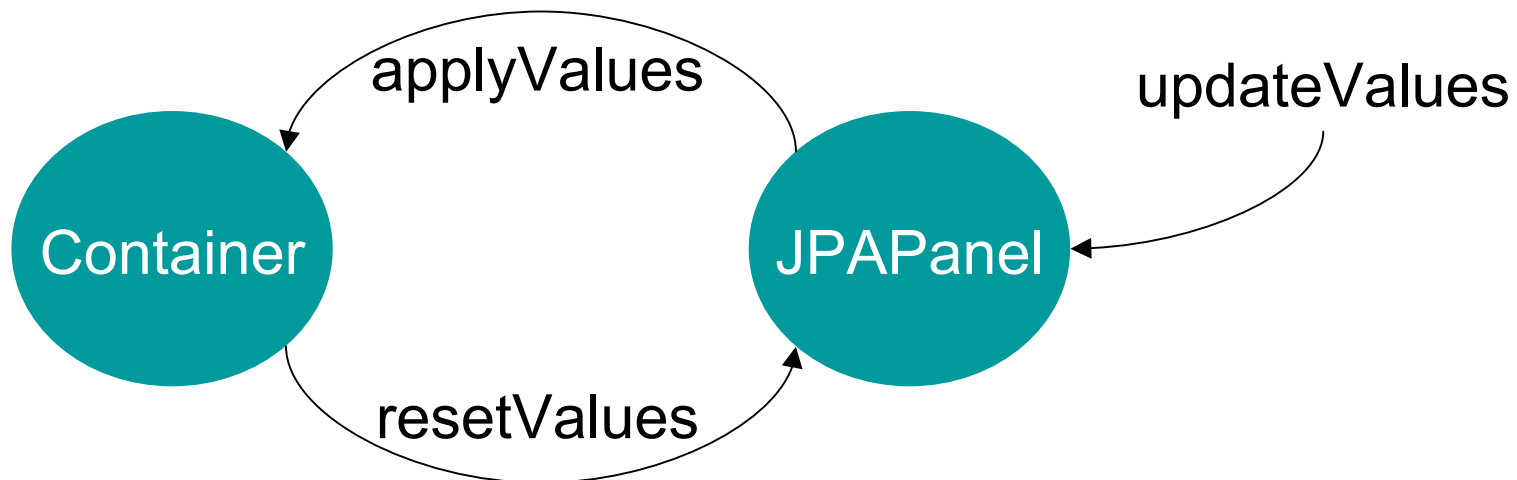
**You implement**  
`buildExecuteGroup()`

# Using Application Resources





- Set parameters in container
- Document/View paradigm
- Sub class of javax.swing.JPanel
- Implements interface *Applicable*
- Follow *Java Look and Feel Design Guidelines*



# Import and Export Panels



- Specify file imports and exports from the GUI
- Use out of the box

Remove Import

Browse file systems

New Import

**File Imports**

Source	File at Source	File in Job Directory	Overwrite File(s)	Binary
Root	usr/share/info/find.info.gz	find.info.gz	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Home	localOutputfile	localOutputfile	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Local	C:\tmp\dateLoop.sh	dateLoop.sh	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

**File Exports**

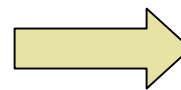
File in Job Directory	Destination	File at Destination	Overwrite File(s)	Binary
output.gif	Home	Documents/	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
outputfile	Local	C:\tmp\outputfile	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

## Step 2: Writing the Boltzmann Plugin



- Execute the application with an AJO UserTask
- Specify input file in import panel
- Specify output file in export panel

1. Unpack Code
2. Compile Code
3. Build Jar
4. Sign Jar
5. Deploy Jar
6. Run Client



**GO!**





- Load, edit and save files from remote and local file spaces
- Use out of the box

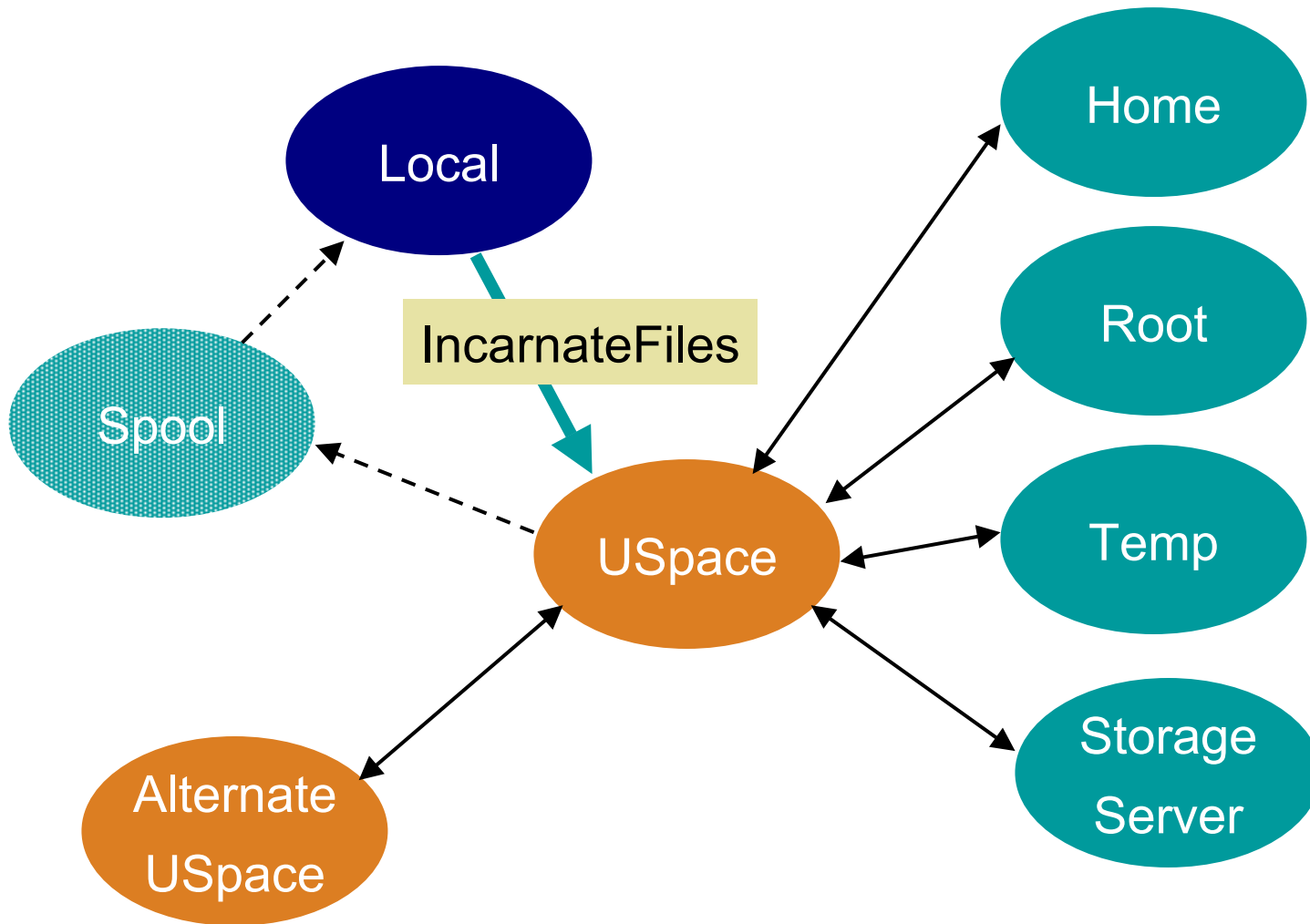
```
public class PluginJPAPanel extends JPAPanel {
    private PluginContainer container;
    private RemoteTextEditor textEditor;

    private buildComponents() {
        textEditor = new RemoteTextEditor();
        JScrollPane editorScrollPane =
            new JScrollPane(textEditor);
    }

    public void applyValues() {
        container.setText(textEditor.getText());
    }

    public void updateValues(boolean vsiteChanged) {
        if(vsiteChanged) {
            textEditor.setVsite(container.getVsite());
        }
    }
}
```

# File Transfers in the AJO

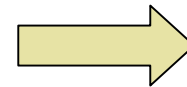


## Step 3: Transferring input and output files



- Edit the input file with the RemoteTextEditor
- Plugin sends the input file with IncarnateFiles
- User adds output file in export panel

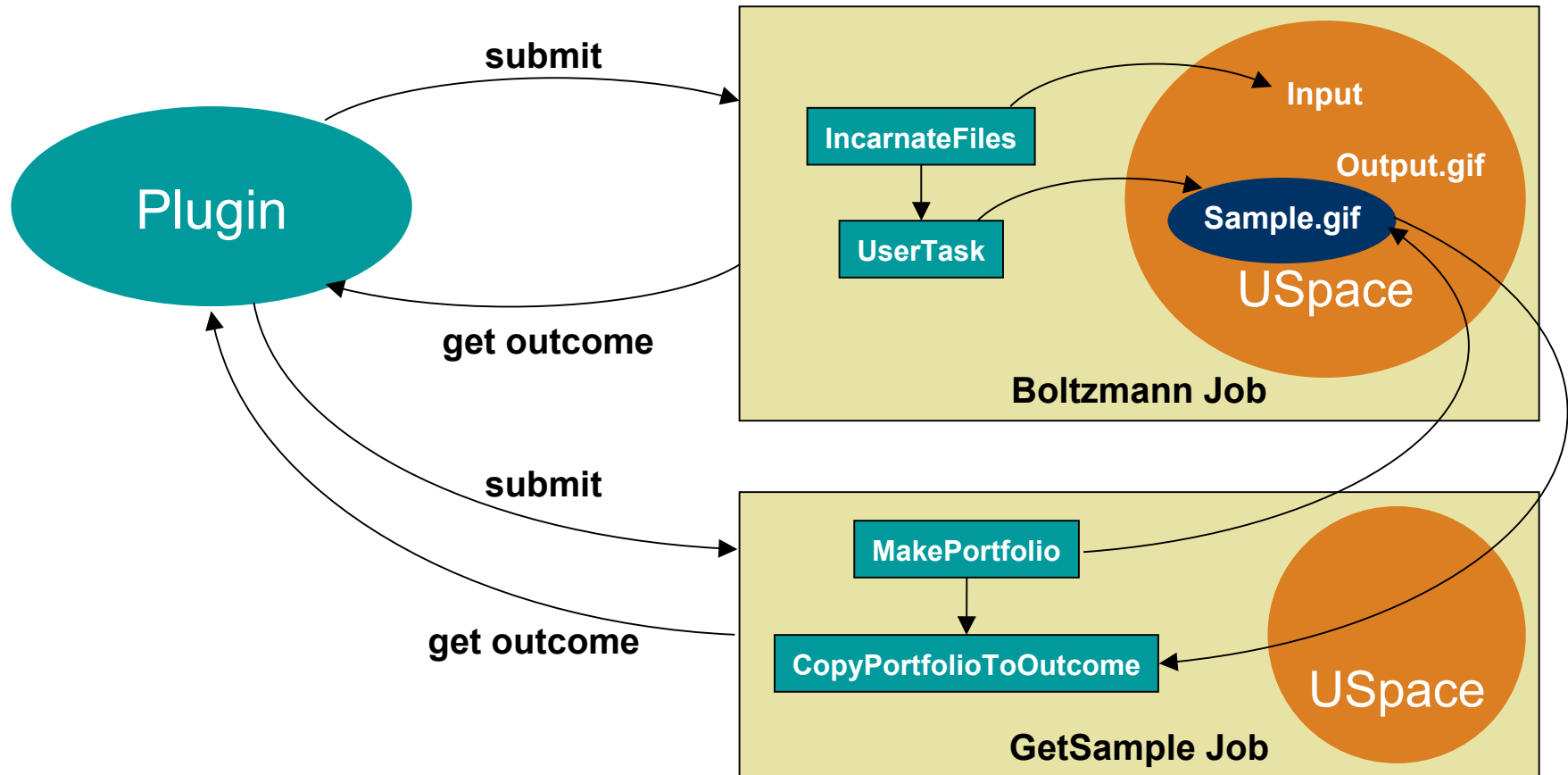
1. Unpack Code
2. **Add IncarnateFiles task to execution ActionGroup**
3. Compile Code
4. Run Client



**GO!**



- Wrap files in USpace in portfolios
- Pass portfolios between tasks





- Make your outcome panel a sub class of JPanel
- Implement interface IPanelProvider in Container
- Implement interface IApplicable in outcome panel

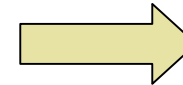
Use `com.pallas.unicore.client.panels.ImagePanel`  
to display `sample.gif`

## Step 4: Getting intermediate results



- Get sample files with MakePortfolio and CopyPortfolioToOutcome (use GetFilesFromUSpace!)
- Visualize sample files in additional outcome panel

1. Unpack Code
2. Add GetFilesFromUospace request
3. Compile Code
4. Run Client



**GO!**



- Write a Boltzmann Wizard
  - Add GUI elements to specify input parameters
  - Generate input file from GUI entries
- Automatically export output.gif
  - Add a FileExport object in Plugin Code
- Send Control files to running application
  - Add a control panel to outcome area
  - Write a SendFilesToUospace request
- ...???