# Grid Summer School 2004

Client Plug-in Programming Exercises

Ralf Ratering (ralf.ratering@intel.com)

## Getting started

Before you start the exercises, please make sure that you have the following software installed:

1. Java SDK 1.4.2 (Sun or BEA JRockit)
2. Eclipse 3.0
3. Apache Ant 1.6.1

You should also have the UNICORE Client 5.0 installed and have access to a virtual site. You will find all necessary source code, scripts and libraries in the GridSchool2004.zip archive.

1. The first thing we will do now is to unzip the archive GridSchool2004.zip into a directory of your choice
2. Start Eclipse
3. Select "New->Project" and then "Java Project"
4. Project Name: "GridSchool" and IMPORTANT: "Create Project at external location". Enter the directory name where you unzipped the GridSchool2004.zip archive (The parent directory of the "GridSchool" directory).
5. Press "Next"
6. "Source folders on build path" should point to "GridSchool/src" and "Default output folder" to "GridSchool/bin"
7. Press "Finish"

To build, compile and run our project we will use the Ant script "build.xml" in the "GridSchool" folder. This script can be invoked from the within Eclipse through the following steps:

1. Go to "Window->Show View->Ant"
2. In the Ant window select "Add buildfiles" and choose the GridSchool/build.xml file
3. Double-click "runClient" in the Ant Window The Client should show up. Run a simple script job to verify your installation. Check that there are no plug-ins loaded except for the script and command plug-ins.

## Building and running the example plug-ins

In the Eclipse package explorer view you will see the "src" folder containing 5 plug-in packages:

1. AJO Plug-in: A simple example on how to create AJOs in a plug-in
2. PovRay Plug-in: The ray tracing plug-in from the standard distribution. This plug-in demonstrates the use of additional outcome panels
3. Boltzmann Plug-in: A task plug-in that demonstrates steering functionality
4. SmallService Plug-in: An example of an extension plug-in executing a software resource

5. Hello Plug-in: This is the template for the plug-in that we will develop during the exercises.

To build the plug-ins double-click on the “all” target in the Ant window. The Ant script will build the plug-in jar files in the release directory and will use the Keystore file in the “GridSchool” directory to sign the jar files. To load the plug-ins in the Client you have to set the Plugin directory in “Settings->User Defaults->Paths” to the “GridSchool/release” folder that contains the new plug-ins.
When you “Reload Plug-ins” you will be asked if you want to add the signing certificate permanently to the keystore. After adding the certificate to the keystore you will see the plug-ins in the Client menus.
We are now ready to create our first simple plug-in.

## Exercise 1: Building and running the “Hello” Plug-in

To build the Hello plug-in you have to modify the Ant build.xml script edit the “list of plug-ins”, “build plug-ins” and “sign plug-ins” sections. Then rebuild the plug-ins with the “all” target. If re-start the Client now or press “Reload Plug-ins” you should see a new task plug-in called “Hello” that will execute the HelloWorld SoftwareResource at a virtual site. What is the output of this application?

## Exercise 2: Importing and Exporting Files

Add an import panel and an export panel to your plug-in. Import and export panels can be used as “out-of-the-box” components and will create the necessary AJO tasks automatically. Please, take a look at the POVRay-Plug-in source code on how to add these panels to the JPAPanel in your plug-in.
To verify your installation, create a file called “myFile.txt” on your local file system, and import it into the USpace with the import panel in your plug-in. The target name in the USpace should be “outputFile.txt”. Then add an export in the export panel to transfer the file “output.txt” from the USpace back to your local file system.
Submit the job and see if the file output.txt appears in your local directory after pressing “Fetch Outcome” in the Job Monitor.

## Exercise 3: Executing the “tac” application

So far we have been executing the “HelloWorld” application on the server side using an AJO UserTask that will be created in the class “HelloContainer”. Now we want to execute the “tac” application instead. Change the “APPLICATION_NAME” of the executed resource to “tac”.
To verify your implementation, import a “myFile.txt” to the USpace and rename it to “inputFile.txt”. The “tac” application will look for this file and create an output file with the name “tacOutput.txt”. Export this result file to your local file system.
What are the contents of “tacOutput.txt?”

## Exercise 4: Use the RemoteTextEditor to specify the input for “tac”

A convenient way to edit input files for your application is to use the RemoteTextEditor component in your plug-in. With this editor it is possible to edit files on your local file system as well as on remote file systems. Take a look at the Boltzmann-Plug-in to learn how you can add this component to your plug-in JPAPanel.
The text that is displayed in the editor should now automatically be transferred to the USpace in a file called “inputFile.txt”, before the “tac” application will be executed. A similar mechanism is used in the Boltzmann plug-in, so you should take a look at the BoltzmannContainer to understand how the “IncarnateFiles” task can be used for the import.
To verify your implementation enter a text  in your plug-ins editor and export the “tacOutput.txt” file from the Uspace to your local file system. The “tacOutput.txt” file should now contain the editor contents in reverse order.

## Exercise 5: Display “tac” output in an additional outcome panel

The final component of our plug-in will be an additional visualization panel in the outcome area that displays the file “tacOutput.txt”. The same mechanism is used in the POVRay plug-in to display the output images, so this should give you an example on how to use additional outcome panels. Take a closer look at the interface IPanelProvider and at the method getAdditionalExports() in the PovRayContainer.
If your implementation is correct it should no longer be necessary to manually specify any imports or exports. Just enter the text in the plug-in text editor, submit the job, fetch the outcome in the job monitor and view the result in your additional outcome panel. This is the way that most applications work, so you may use your simple plug-in soon as a template for more complex applications running on UNICORE Grids.
But for now you’re done, so go to the beach or have a drink….