UNICORE
Introduction to the Intel Client
and a look behind the scenes…

Grid Summer School, July 28, 2004

Ralf Ratering

Intel

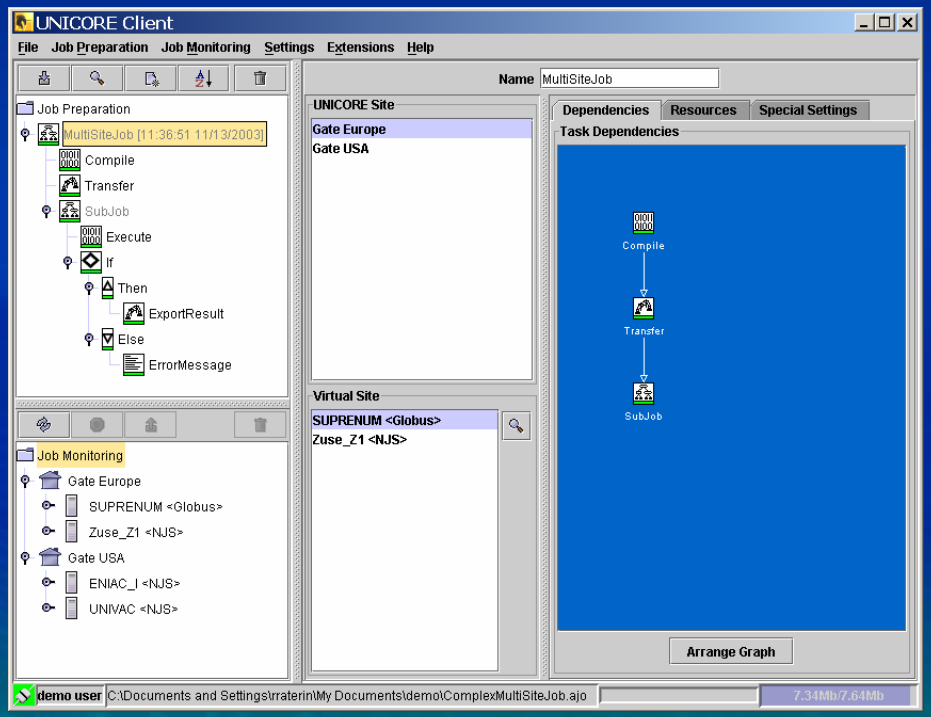
Parallel and Distributed Solutions Division (PDSD)

Outline

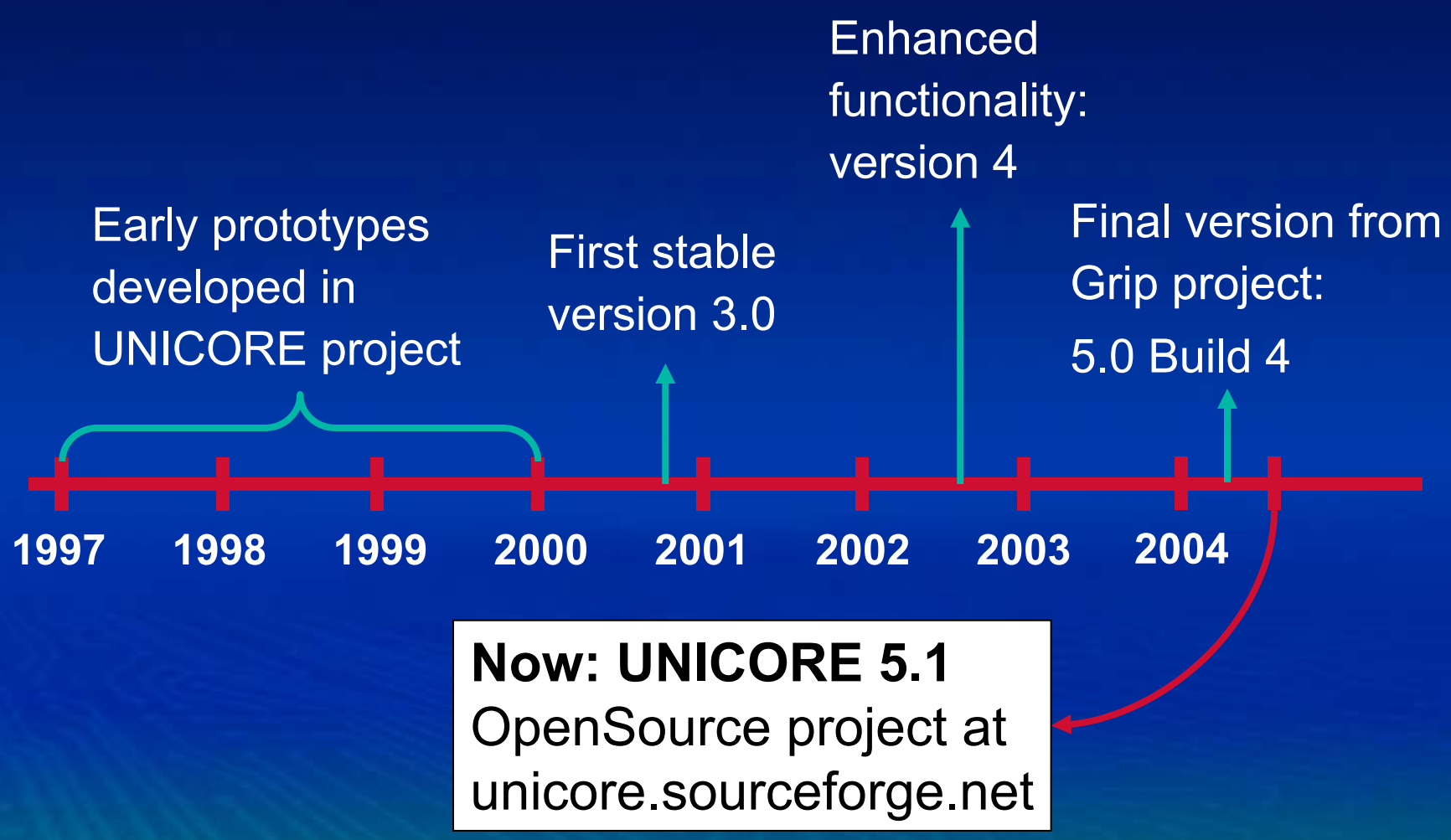
- Getting started with the UNICORE client
- Constructing jobs in the client
- Integrated application support
- A real-world application

The Intel UNICORE Client

- Graphical interface to UNICORE Grids
- Platform-independent Java application
- Open Source available from UNICORE Forum
- Functionality:
  - Job preparation, monitoring and control
  - Complex workflows
  - File management
  - Certificate handling
  - Integrated application support

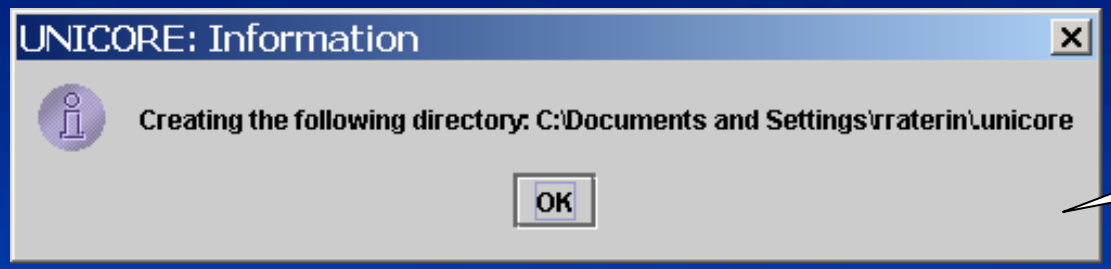


History of UNICORE Client Versions



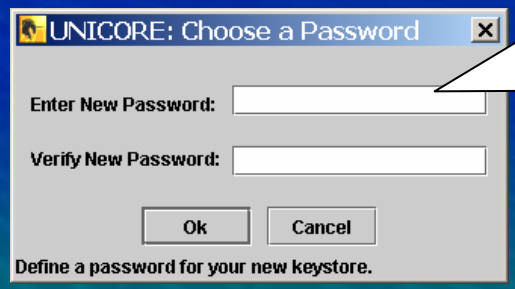
Starting the Client

- Prerequisites: Java ≥ 1.4.2



UNICORE configuration directory <.unicore> in your HOME directory

- Automatically creates an empty keystore and imports trusted certificates from „cert“ directory



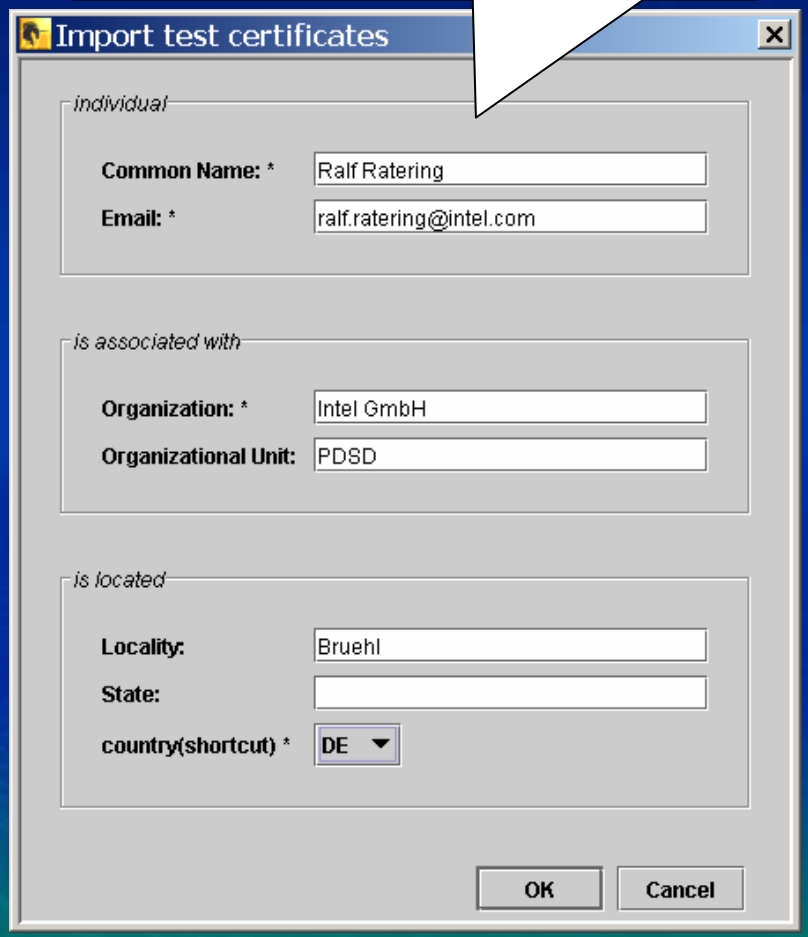
Define password for your unicore keystore file (.unicore/keystore)

Getting a Test Certificate

- „Import test certificates“ from „Settings->Keystore Editor“

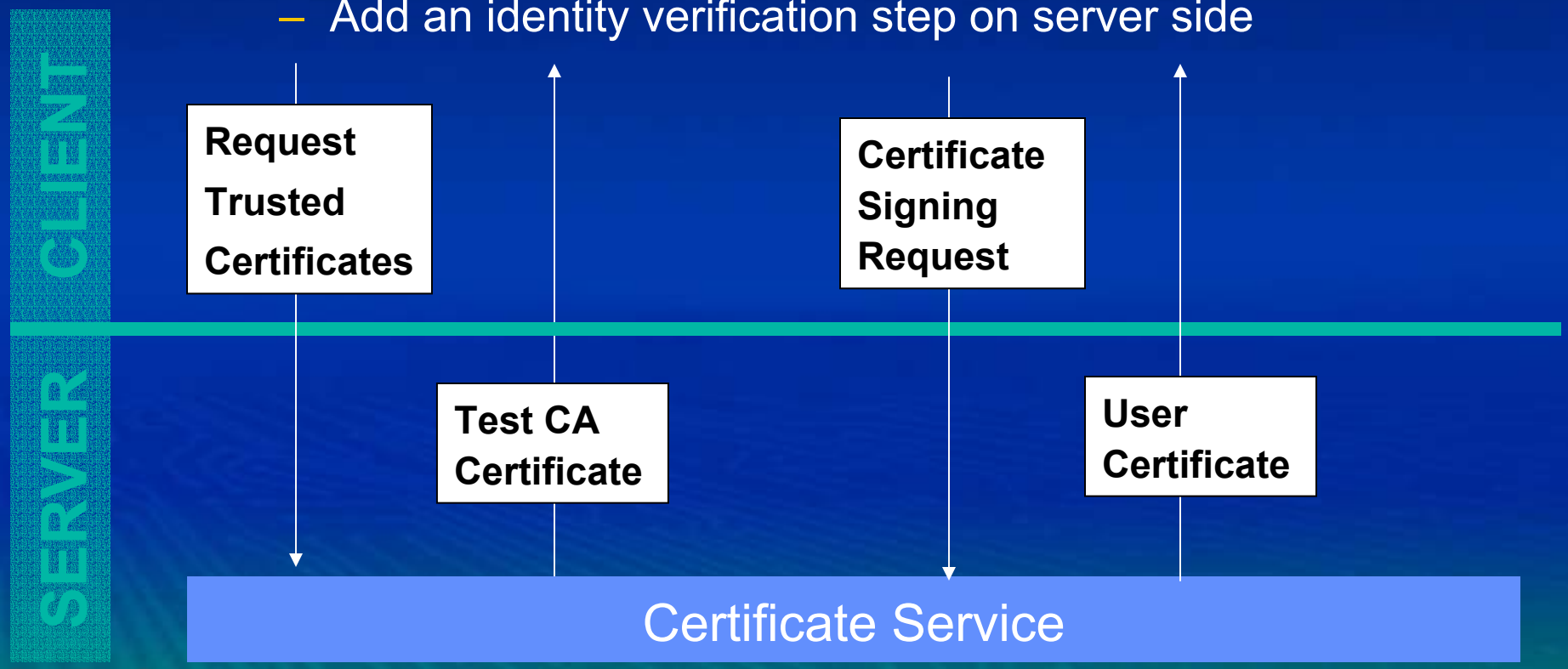
CA web service endpoint

Certificate signing request (CSR) Information will be used to generate a test certificate for you.



Certificate Web Services

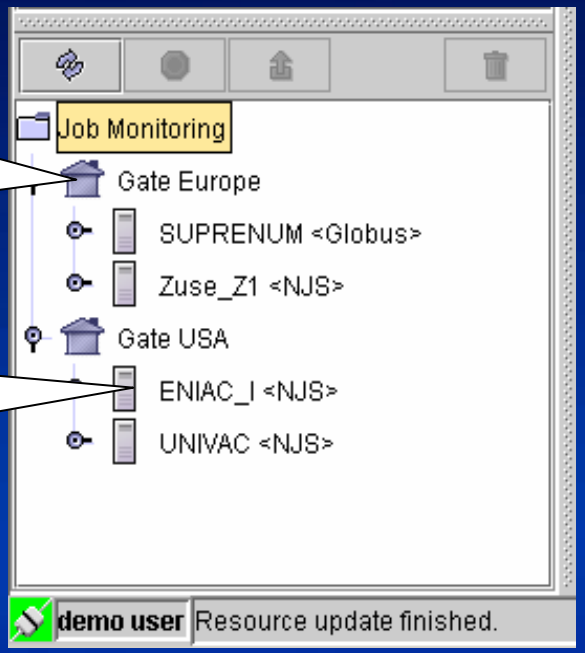
- Low Security Model for Test Grid Access
- Certificates are imported automatically into Client
- Currently implemented at Research Center Jülich:
  - Add an identity verification step on server side



Ready to go? „Hello Grid World!“

UNICORE Site == Gateway
Typically represents a computing center

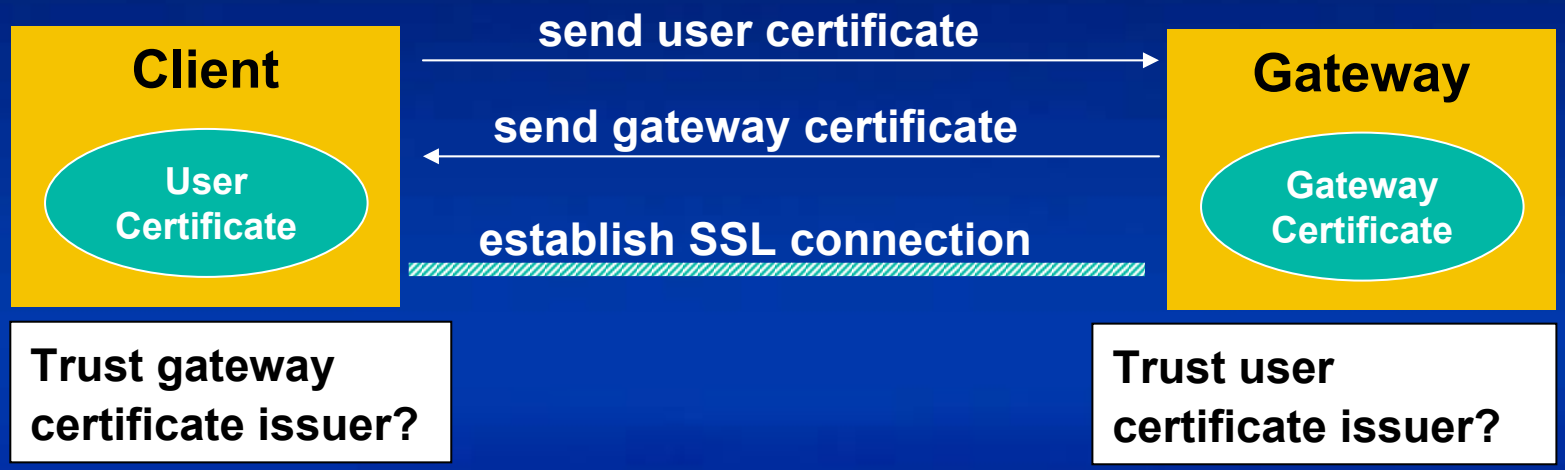
Virtual Site == Network Job Supervisor
Typically represents target system



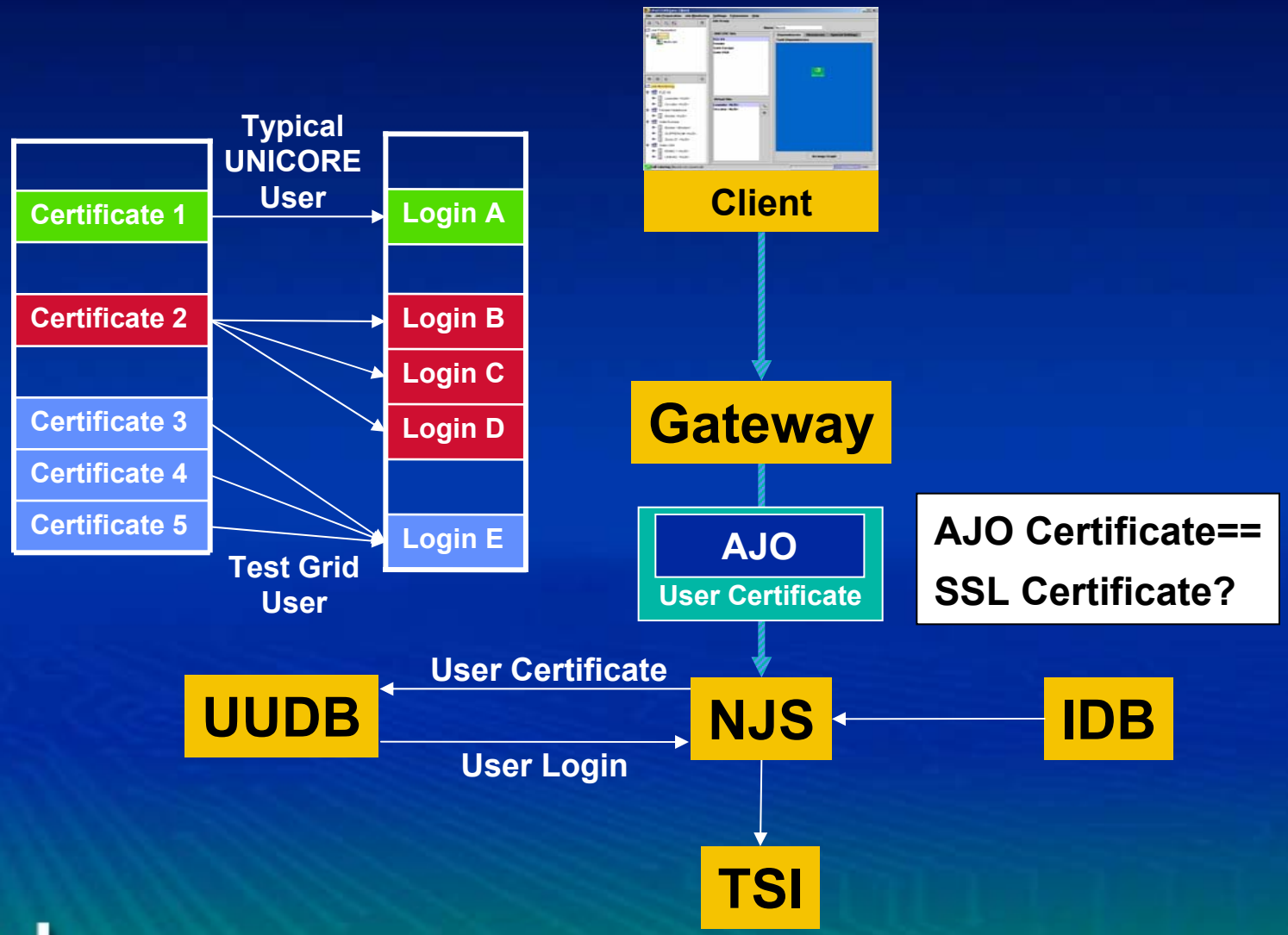
1. **Execute a simple script on the UNICORE Test Grid**
2. **Get back standard output and standard error**

DEMO

Behind the Scenes: Authentication

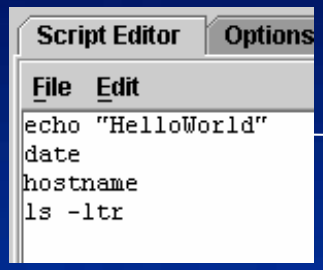


Behind the Scenes: Authorization



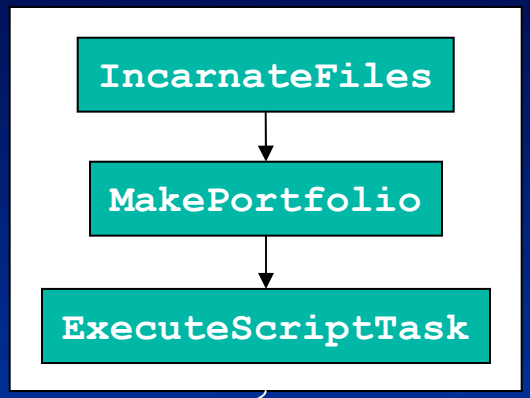
Behind the Scenes: Creation & Submission

CLIENT



Script Container

Abstract Job Object



SERVER

1. **Create file with script contents**
2. **Wrap file in portfolio**
3. **Execute portfolio as script**

Script_HelloWorld1234...

Job Directory (USpace)

A temporary directory at the target system where the job will be executed

Monitoring the Job Status

- Successful: job has finished succesfully
- Not successful: job has finished, but a task failed
- Executing: Parts of a job are running or queued
- Running: Task is running
- Queued: Task is queued at a batch sub system
- Pending: Task is waiting for a predecessor to finish
- Killed: Task has been killed manually
- Held: Task has been held manually
- Ready: Task is ready to be processed by NJS
- Never run: Task was never executed

The Primes Example

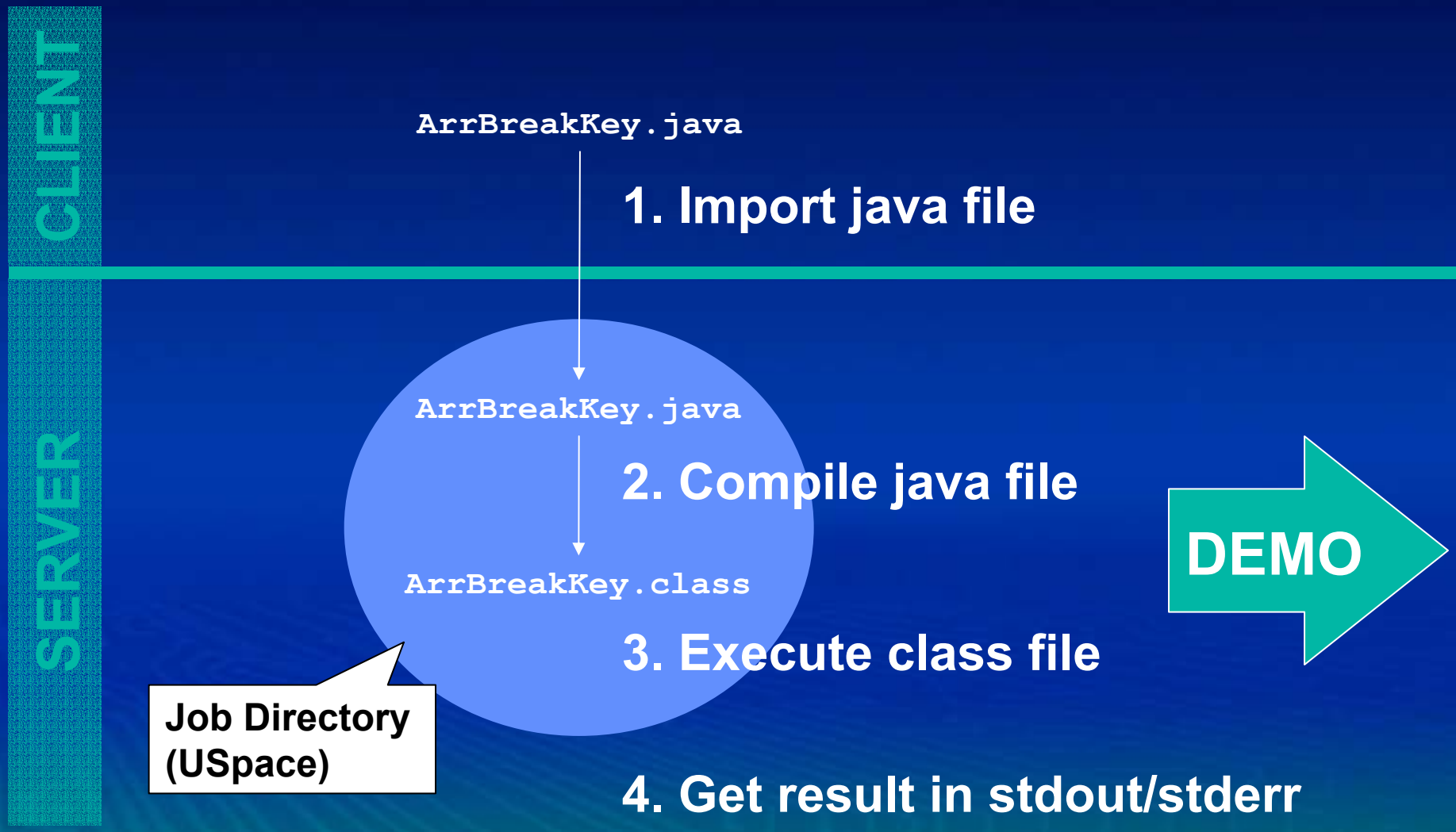
```
public void breakKey() {
  try {
    BufferedReader br = new BufferedReader(new FileReader("primes.txt"));
    while (true) {
      inputLine = br.readLine();
      st = new StringTokenizer(inputLine," ");
      val = new BigInteger(st.nextToken());
      if ( (N.mod(val).compareTo(BigInteger.ZERO)) == 0) {
        p = val;
        q = N.divide(val);
        return;
      }
    }
  } catch (NullPointerException e) {
    System.out.println("Done!");
  } catch (IOException e)  {
    System.err.println("IO Error:" + e);
  }
  p = BigInteger.ZERO;
  q = BigInteger.ZERO;
}
```

ArrBreakKey.java

```
2
3
5
7
11
13
17
19
23
29
31
37
41
43
47
53
59
61
67
71
73
79
...
```

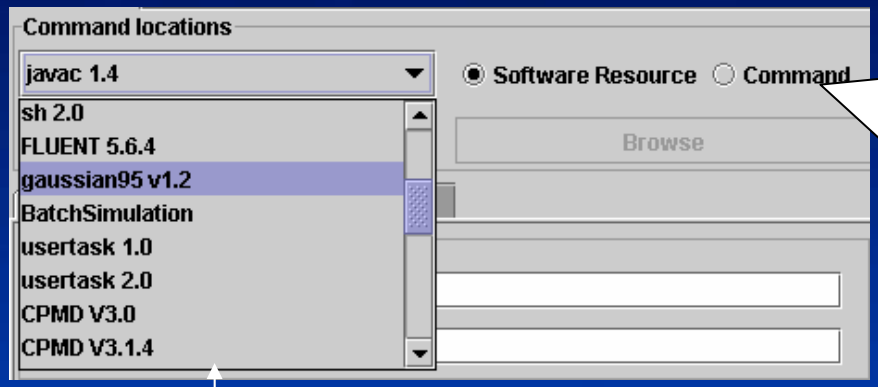
Primes.txt

Demo 1: "Gridify" the Primes Example



Behind the Scenes: Software Resources

CLIENT



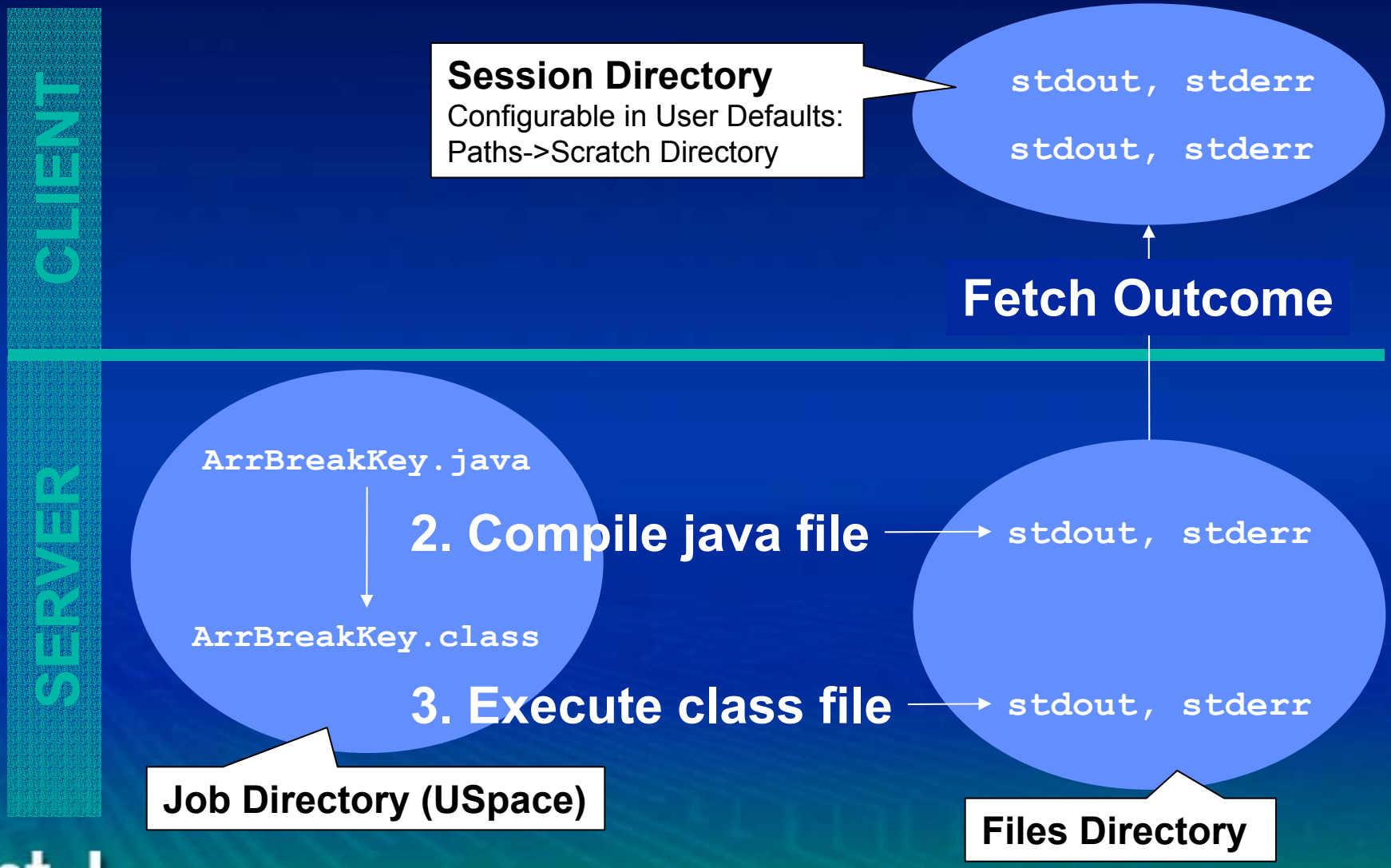
Command Task
Executes a software resource, or command (a binary that will be imported into the job directory)

SERVER

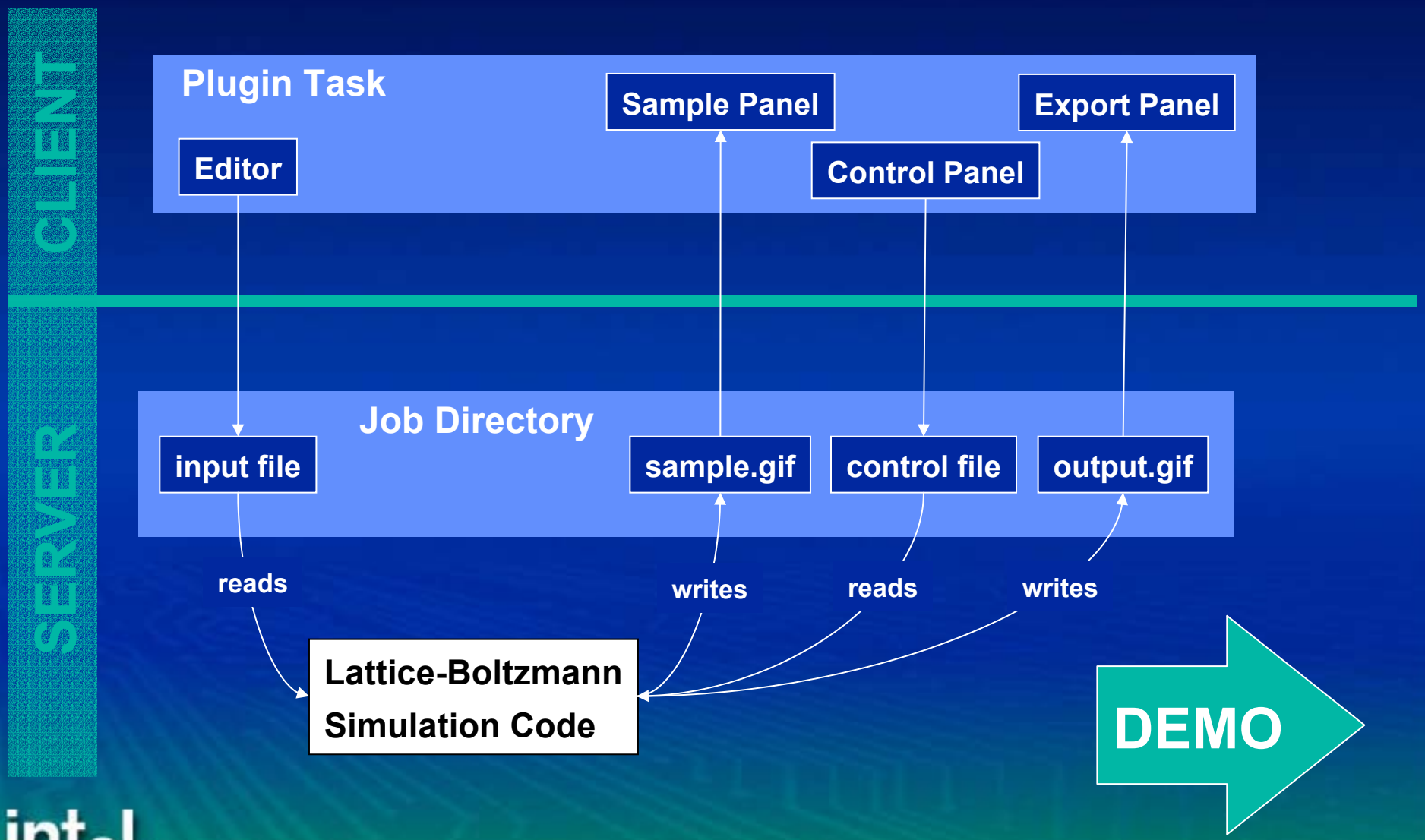
```
APPLICATION javac 1.4
Description „Java Compiler"
INVOCATION [
  /usr/local/java/bin/javac
]
END
```

Incarnation Database (IDB)
Application Resources **contain system specific information, absolute paths, libraries, environment variables, etc.**

Behind the Scenes: Fetching Outcome



Demo 2: Steer the Lattice Boltzmann Simulation

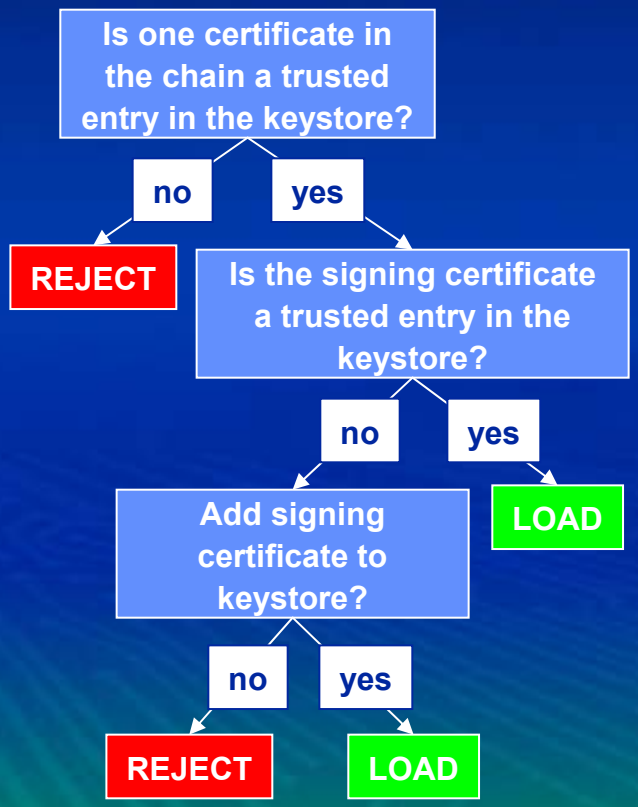


Behind the Scenes: Plug-in Concept

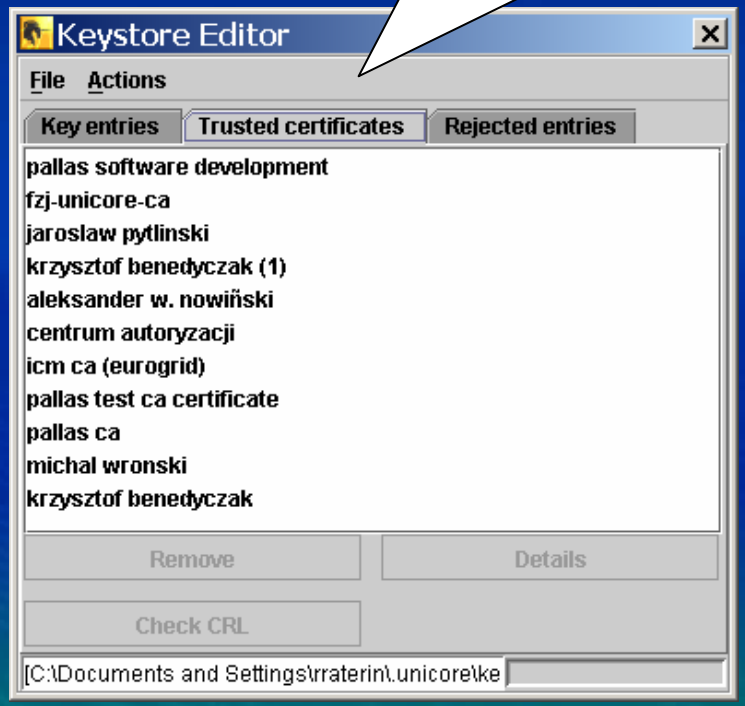
- Add your own functionality to the client!
  - Heavily used in research projects all over the world
  - More than 20 plug-ins already exist
- No changes to basic client software needed
- Plug-ins are written in Java
- Distribution as signed jar archives

Using 3rd Party Plug-ins

- Get plug-in jar file from web-site, email, CD-ROM, etc.
- Store it in client's plug-in directory
- Client will check plug-in signature



Import plug-in certificates from the actions menu in the keystore editor



Task Plug-ins

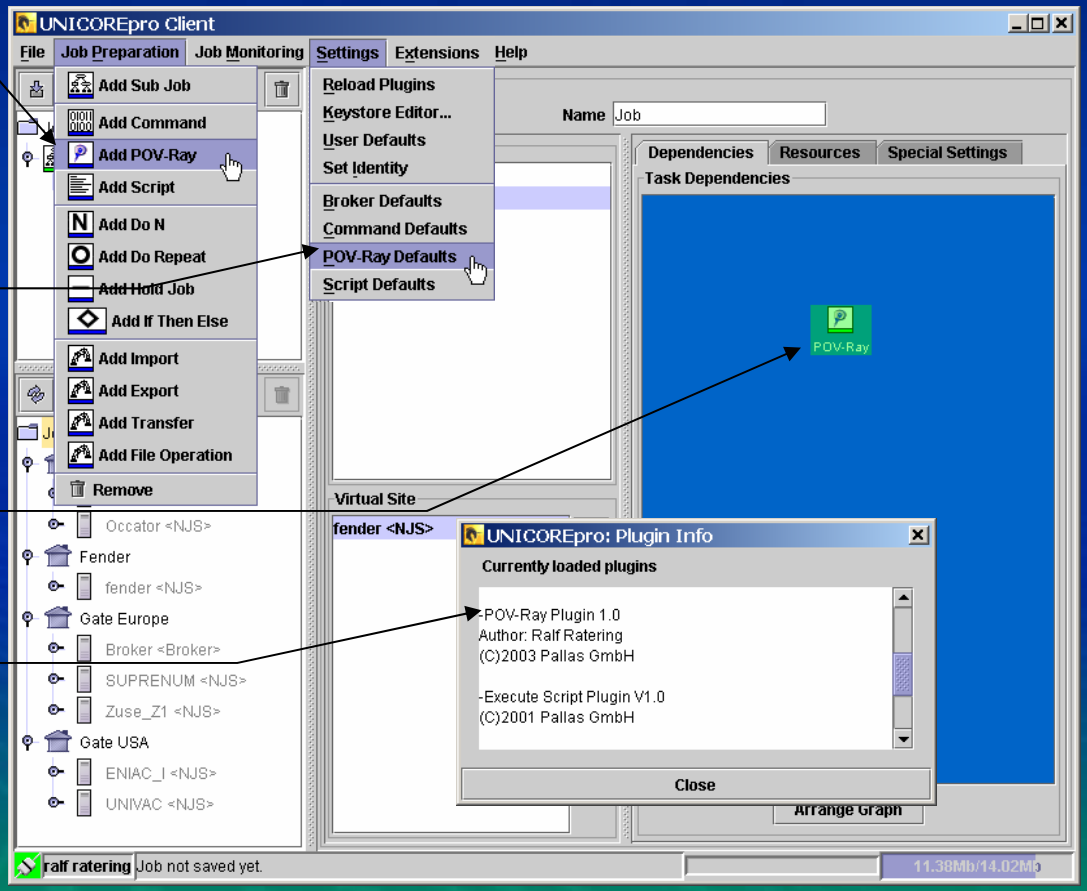
- Add a new type of task to the client GUI
- New task can be integrated into complex jobs
- Application support: CPMD, Fluent, Gaussian, etc.

Add task item

Settings item

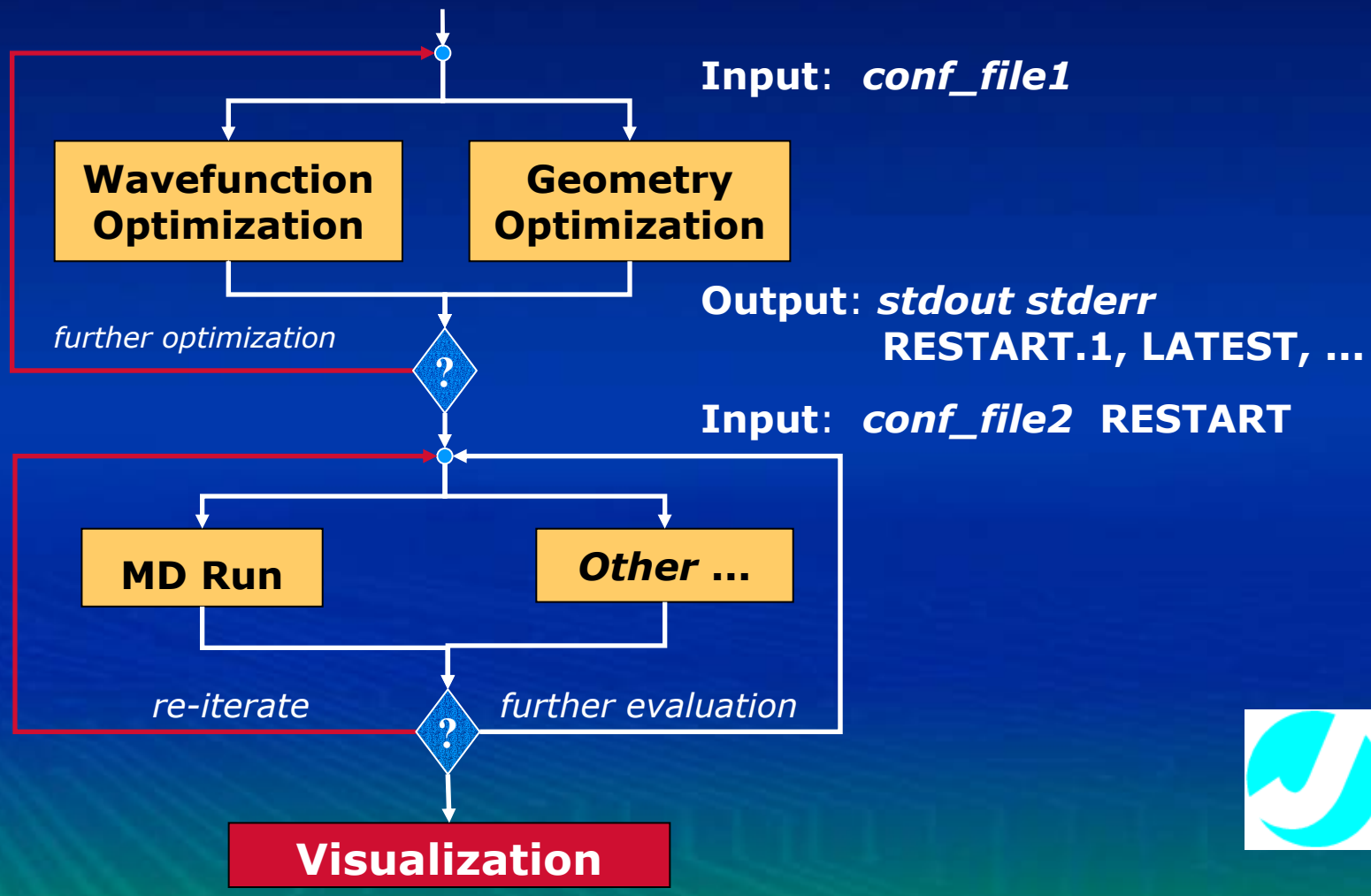
Icon

Plugin info



A Task Plug-in: CPMD

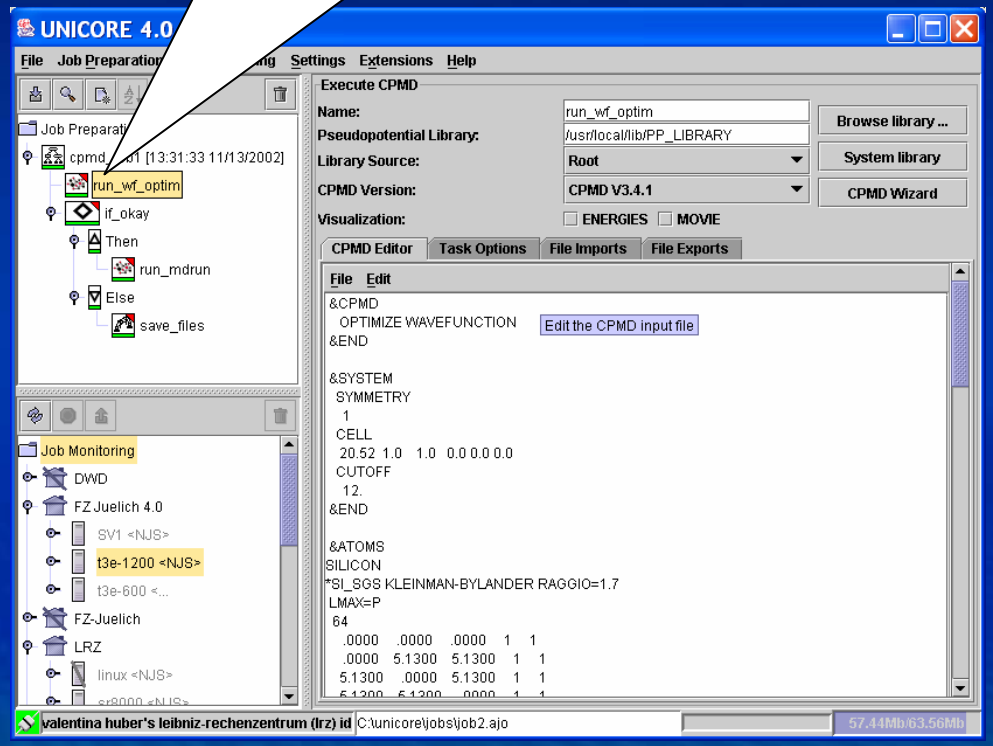
- **Workflow for Car–Parrinello molecular dynamics code**

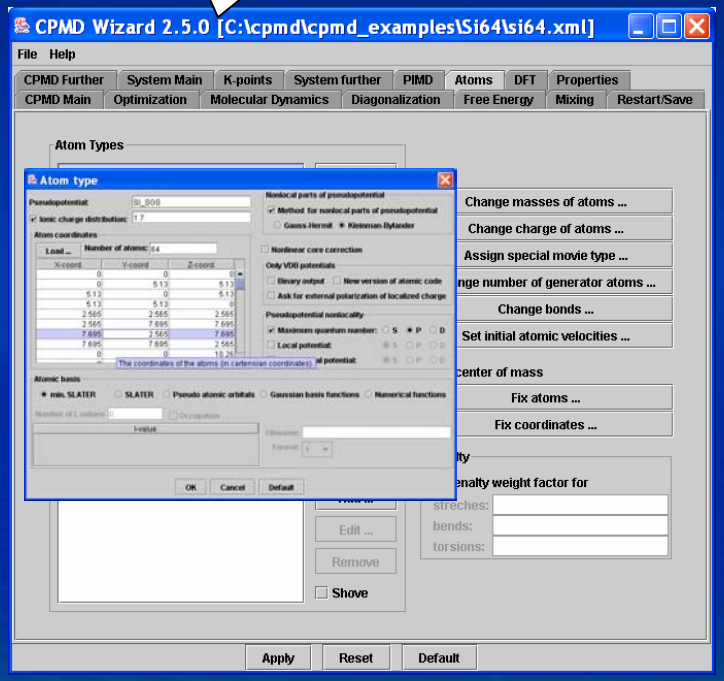


A Task Plug-in: CPMD

CPMD Plug-In Task used in UNICORE workflows

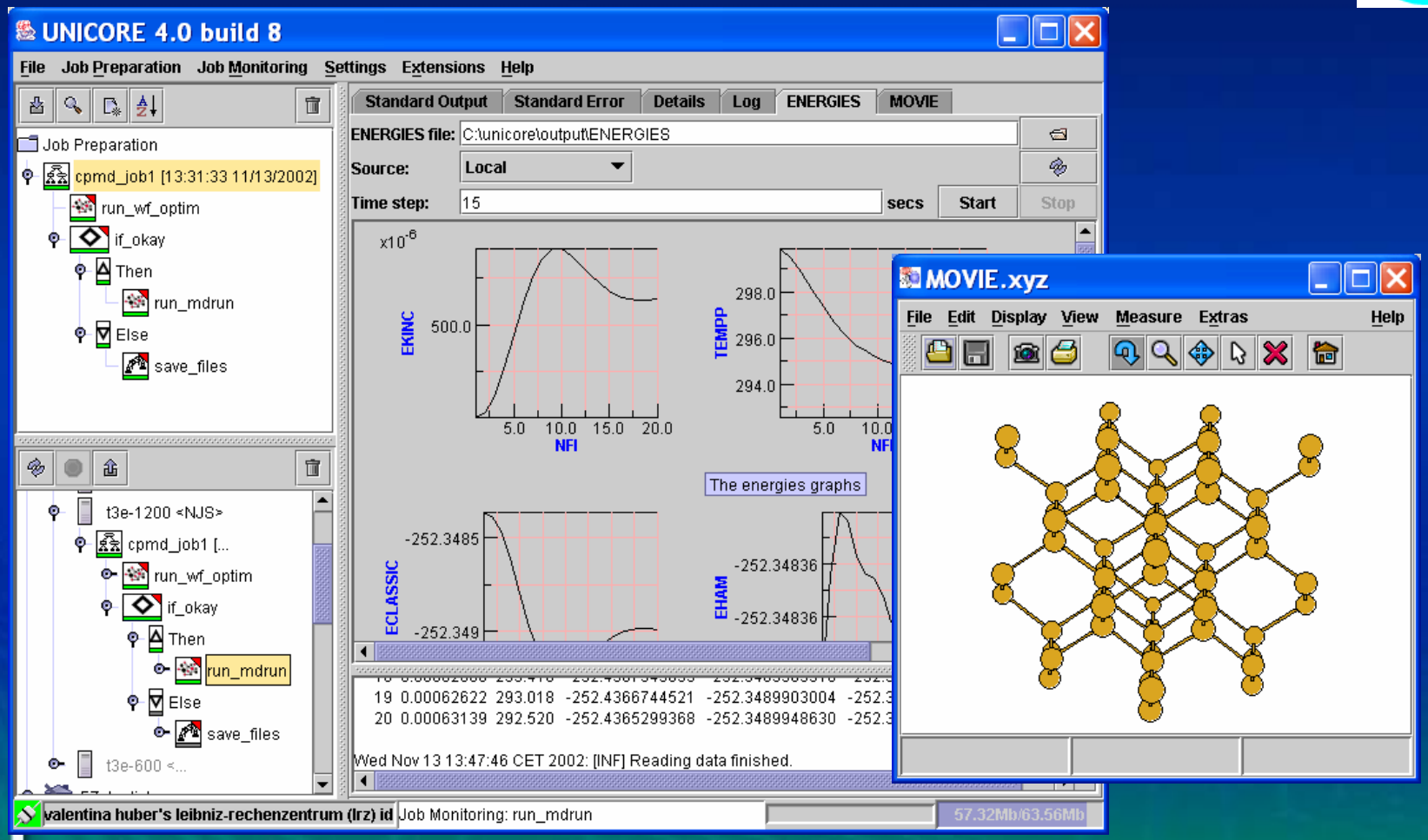
CPMD wizard assists in setting up the input parameters





A Task Plug-in: CPMD

- Visualize results



Supporting an application at a site

- Install the application itself
- Add entry to the Incarnation Database (IDB)

```
APPLICATION CPMD 3.4.1
Description „Car Parrinello Molecular Dynamics Code"
INVOCATION [
    export JOBTYPE=8E8;
    /usr/mpi/bin/mpiexec -p IAPAR  -n $UC_PROCESSORS
    /usr/local/bin/cpmd.x    $CPMD_FILE  $PP_LIBRARY
]
```

Extension Plug-ins

- Add any other functionality
- Resource Broker, Interactive Access, etc.

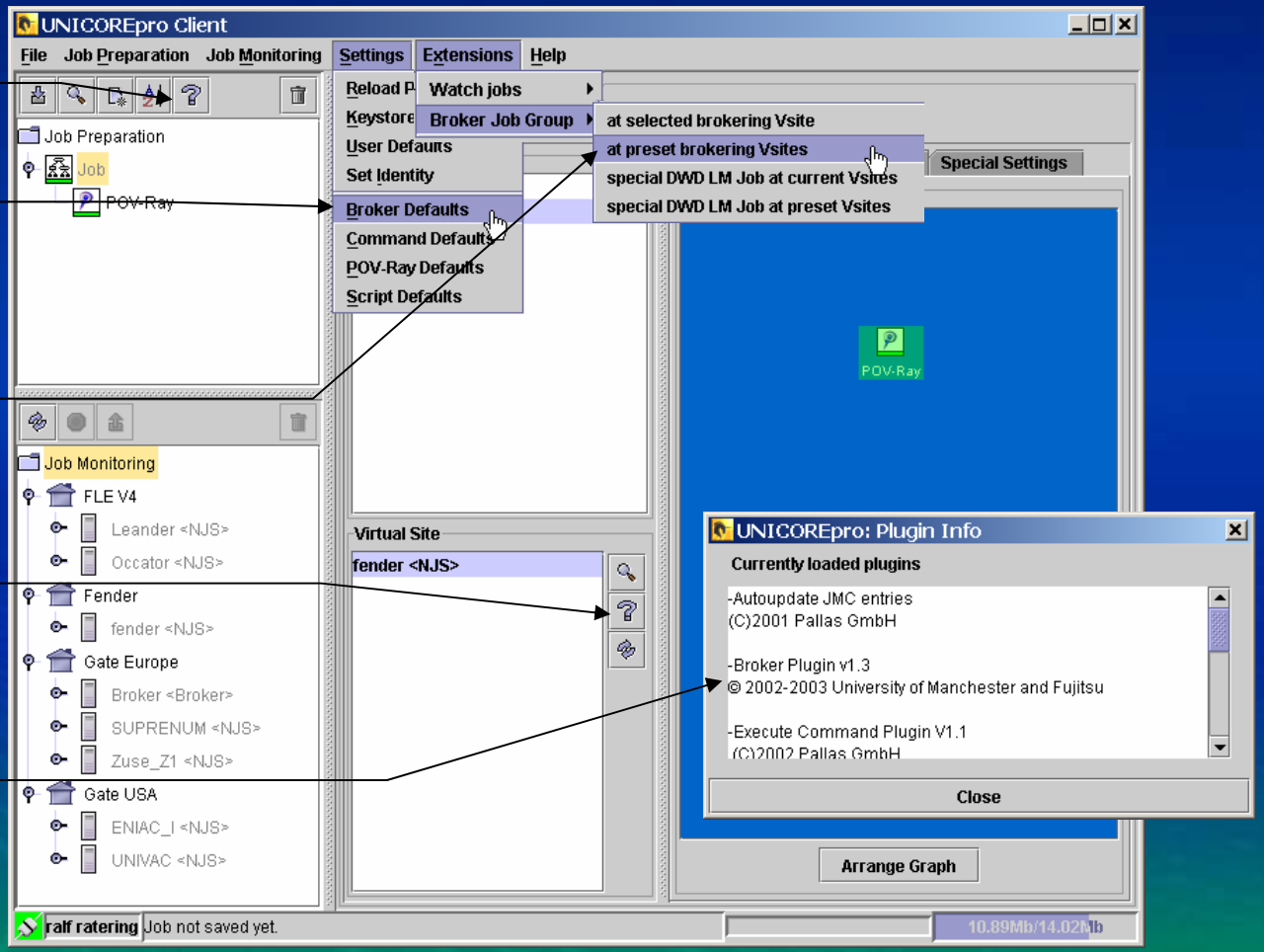
JPA toolbar

Settings item

Extensions menu

Virtual site toolbar

Plugin info



An Extension Plug-in: Resource Broker

- Specify resource requests in your job
- Submit it to a broker site
- Get back offers from broker

UNICORE 4.1

File Job Preparation Job Monitoring Settings Extensions Help

Job Group

Name SubJob_Job

Unicore Site

Resources Special Settings

Offers from Broker

Please select a single offer, and click OK

Virtual Site	Unicore Site	Start Time	End Time	Cost
T3E_turing	Manchester Univer...	1:49 PM to 2:19 PM	1:49 PM to 2:19 PM	1.00741E-4 CSAR ...
T3E_turing	Manchester Univer...	1:49 PM to 2:19 PM	1:50 PM to 2:20 PM	1.00741E-4 CSAR ...
O2000_fermat	Manchester Univer...	1:49 PM to 2:49 PM	1:49 PM to 2:49 PM	1.61875E-4 CSAR ...
O2000_fermat	Manchester Univer...	1:49 PM to 2:49 PM	1:49 PM to 2:49 PM	1.61875E-4 CSAR ...
O300_wren	Manchester Univer...	1:49 PM to 1:49 PM	1:49 PM to 1:49 PM	2.06475E-4 CSAR ...
O300_wren	Manchester Univer...	1:49 PM to 1:49 PM	1:49 PM to 1:50 PM	2.06475E-4 CSAR ...

Tickets in detail

The tickets contain the following:

Ticket for Task Script1
Ticket ID: org.unicore.AAIdentifier@2bbe9ee4
Ticket valid until: Tue Jul 08 14:49:23 BST 2003

ResourceSet:
Node (Number of Nodes) = 1.0 Nodes
Processor (Number of PEs per Node) = 1.0 Processors per node
Memory (Total Amount of Memory) = 20.0 Megabytes per node
RunTime (Time per Job) = 15.0 Seconds
org.unicore.resources.QoSCheck: QoSCheck (QoSCheck)

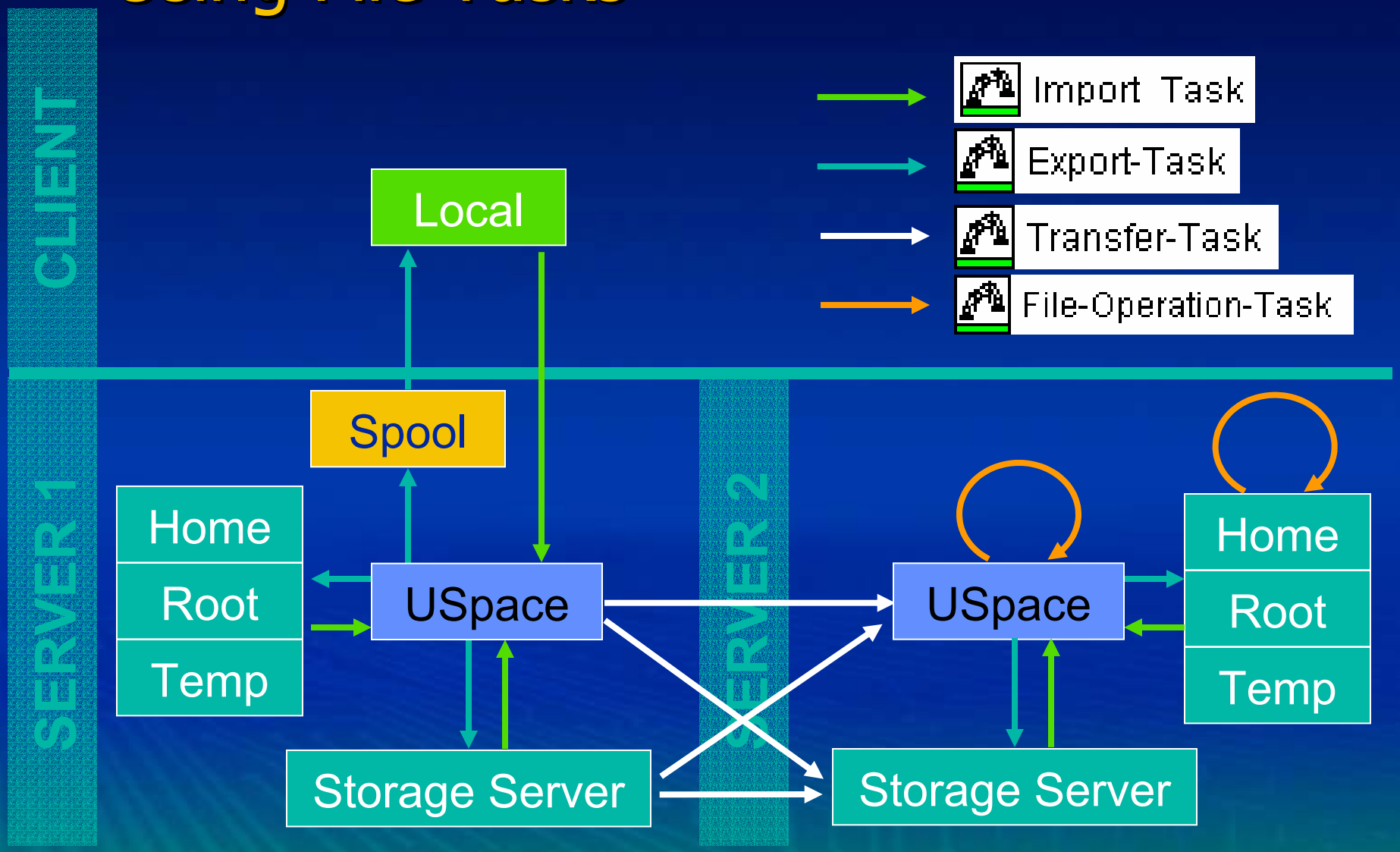
Existing Plug-Ins (incomplete)

- **CPMD (FZ Jülich)**
- **Gaussian (ICM Warsaw)**
- **Amber (ICM Warsaw)**
- **Visualizer (ICM Warsaw)**
- **SQL Database Access (ICM Warsaw)**
- **PDB Search (ICM Warsaw)**
- **Nastran (University of Karlsruhe)**
- **Fluent (University of Karlsruhe)**
- **Star-CD (University of Karlsruhe)**
- **Dyna 3D (T-Systems Germany)**
- **Local Weather Model (DWD)**
- **POV-Ray (Pallas GmbH)**
- **...**

- Resource Broker (University of Manchester)
- Interactive Access (Parallab Norway)
- Billing (T-Systems Germany)
- Application Coupling (IDRIS France)
- Plugin Installer (ICM Warsaw)
- Auto Update (Pallas GmbH)
- ...

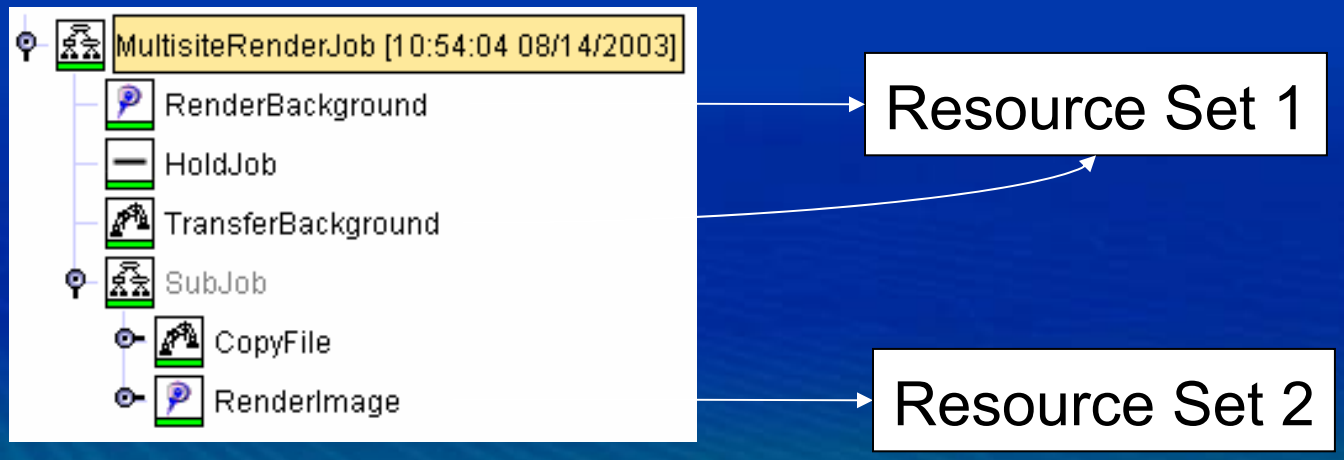
intel®

Using File Tasks



How to specify resource requests?

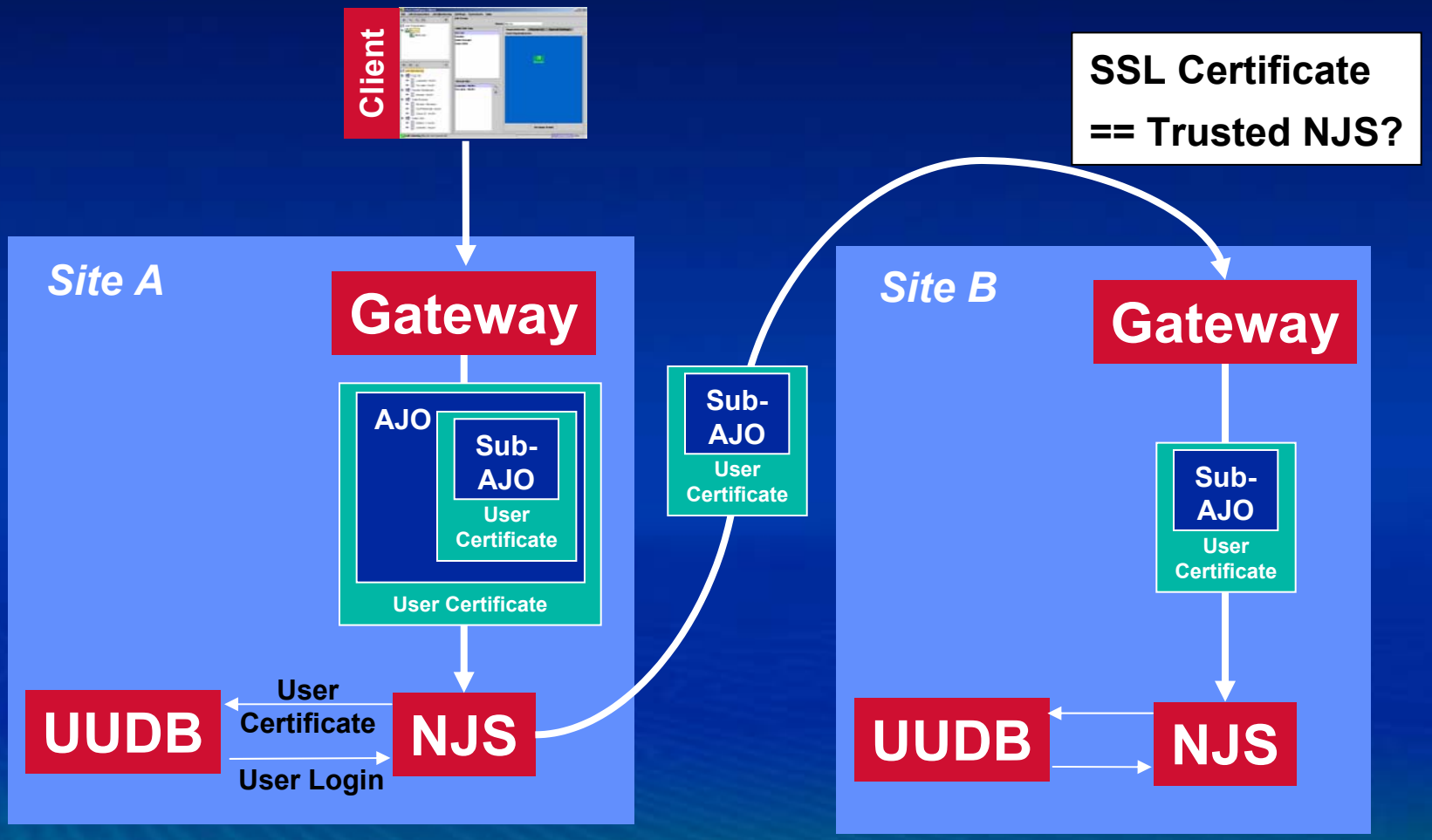
- Tasks can have resource sets containing requests
- If not resource set is attached, default resources are used
- Resource sets can be edited, loaded and saved
- If a resource request does not match resources available at a site, the client displays an error



Demo 3: Run a multi site job

1. Use the primes example
2. Compile the source file on one virtual site
3. Transfer the resulting class file to a sub job running at a different virtual site
4. Execute the class file in the sub job

Behind the Scenes: Authorization

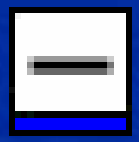


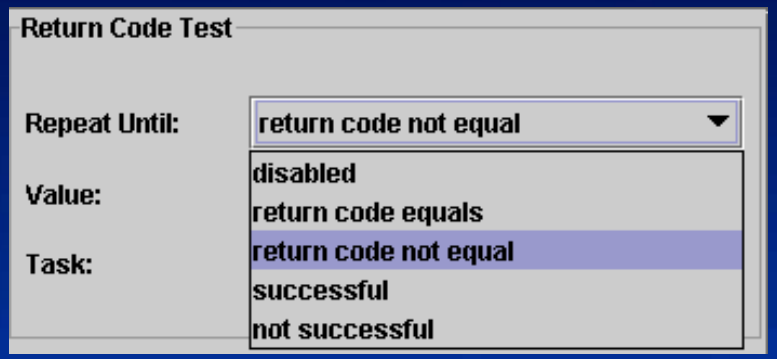
Complex Workflow: Control Tasks

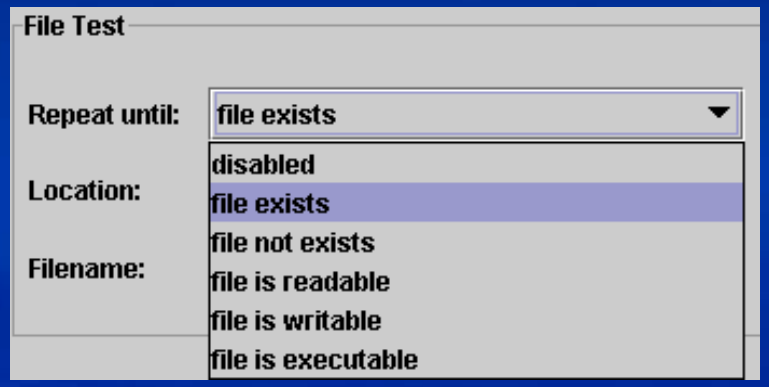

Do Repeat Loop

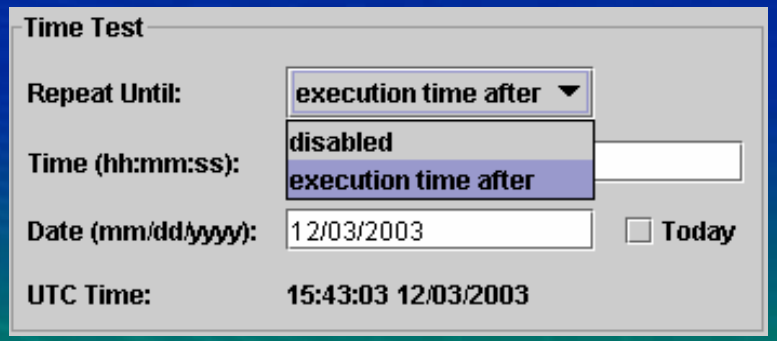

If Then Else


Do N Loop


Hold Task







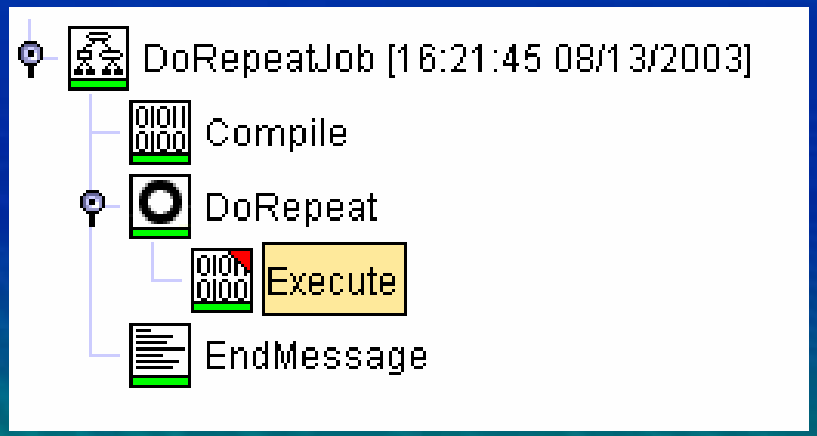
Demo 4: Test the return code in a loop

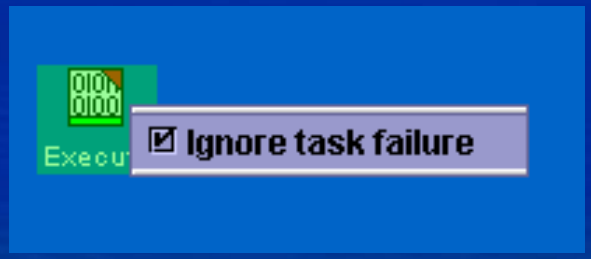
```
import java.util.Random;

public class Application {

  public static void main(String[] args) {

    Random rnd = new Random(System.currentTimeMillis());

    double random = rnd.nextDouble();

    System.out.println("RANDOM: " + random);

    int exitCode = (int)(5*random);

    System.out.println("EXIT CODE: " + exitCode);

    System.exit(exitCode);

  }

}
```

Repeat execution until it fails with a exit code 2!

Behind the Scenes: Ignore Failure

- UNICORE jobs stop execution when a task fails
- Sometimes Task failure is acceptable
  - If and DoRepeat conditions
  - Tasks that try to use restart files
  - Whenever you do not care about task success
- Set „Ignore Failure“ flag on Task





Right Mouse Click in Dependency Editor

Loops: Accessing the iteration counter

- Iteration variable: `$UC_ITERATION_COUNTS`
- Lives on server side
- Supported in
  - Script Tasks
  - File Tasks
  - Re-direction of stdout/stderr
- Nested loops: iteration numbers are separated by „_“, e.g. „`2_3`“
- Caution: counter will not be propagated to sub jobs

Integrated Application Example: POV-Ray

CLIENT

SERVER

Scene Description

```
#include "colors.inc"
#include "shapes.inc"
camera {
   location <50.0, 55.0, -75.0>
   direction z
}
plane {y, 0.0 texture {pigment {RichBlue }}}
object { WineGlass translate -x*12.15}
light_source { <10.0,50.0,35.0> colour White }
...
```

Command Line Parameters

Include Files

Display

Input Files

Output Image

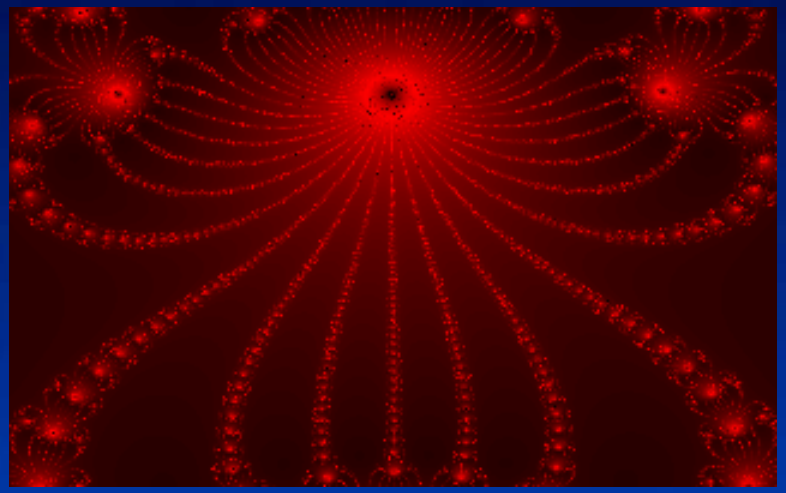
Libraries

POV-Ray Application

Job Directory (USpace)

Remote File System (XSpace)

Demo 5: Hold and release a job



Demo Images from Pov-Ray Distribution

1. Render Background Image
2. Hold Job to check Image
3. Manually Resume Job Execution
4. Render Final Image

DEMO

Job Monitor Actions

Get new status for a site, job or task

Remove job from server. Deletes local and remote temporary directories

Kill job

Hold job execution

Resume a job that was held by a „Hold Job“ action or a Hold task

Show dependencies of job

Show resources for task

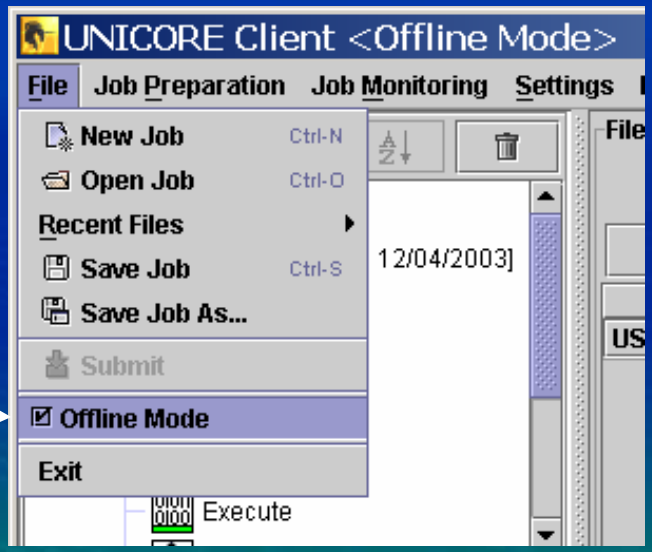
- Refresh
- Stop Refresh
- Fetch Output
- Remove
- Abort Job
- Hold Job
- Resume Job
- Copy
- Collapse All
- Show Dependencies
- Show Resources

Get stdout, stderr and exported files of a job

Copy a job from the job monitor. The job can be pasted into the job preparation tree and re-run e.g. with different parameters

Caching Resource Information

- Client works on cached resource information
  - UNICORE Sites, Virtual Sites, available resources
- Resource cache will be updated on...
  - ... startup
  - ... refresh on „Job Monitoring“ tree node
- Client uses cached information in offline mode

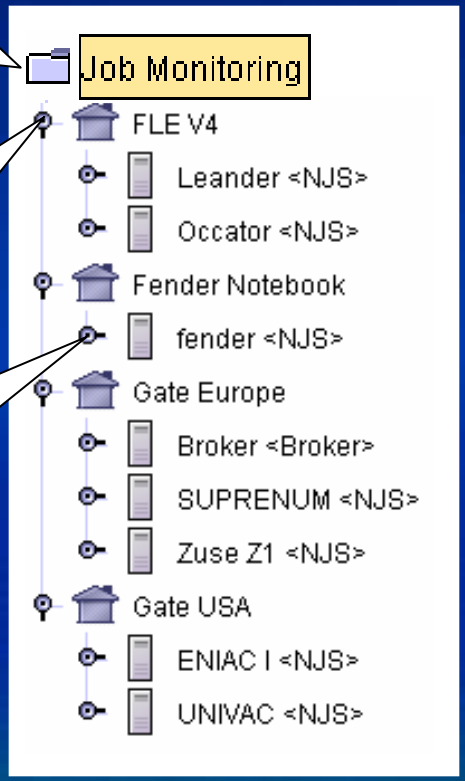


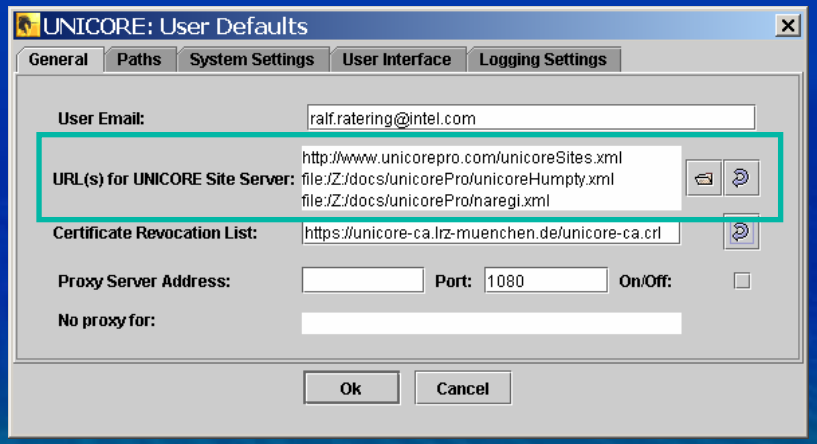
Accessing other UNICORE Sites

Job Monitor Root
Performing a „**Refresh**“ on this node will reload UNICORE Sites

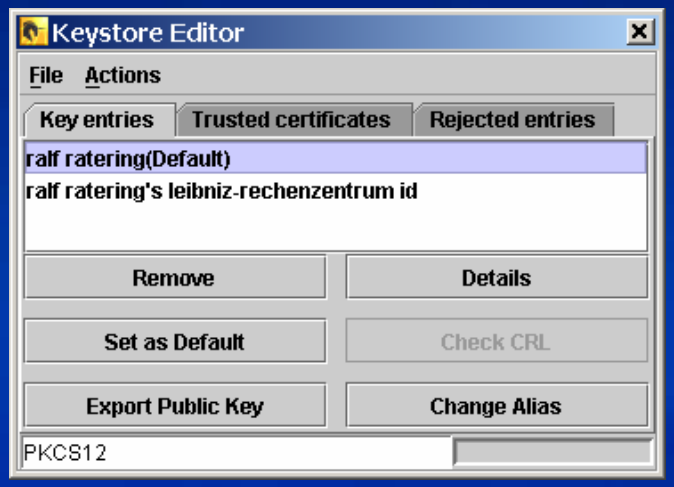
UNICORE Sites
will be read from an XML file
Can be a URL on the web

Virtual Sites
are configured at the UNICORE Site

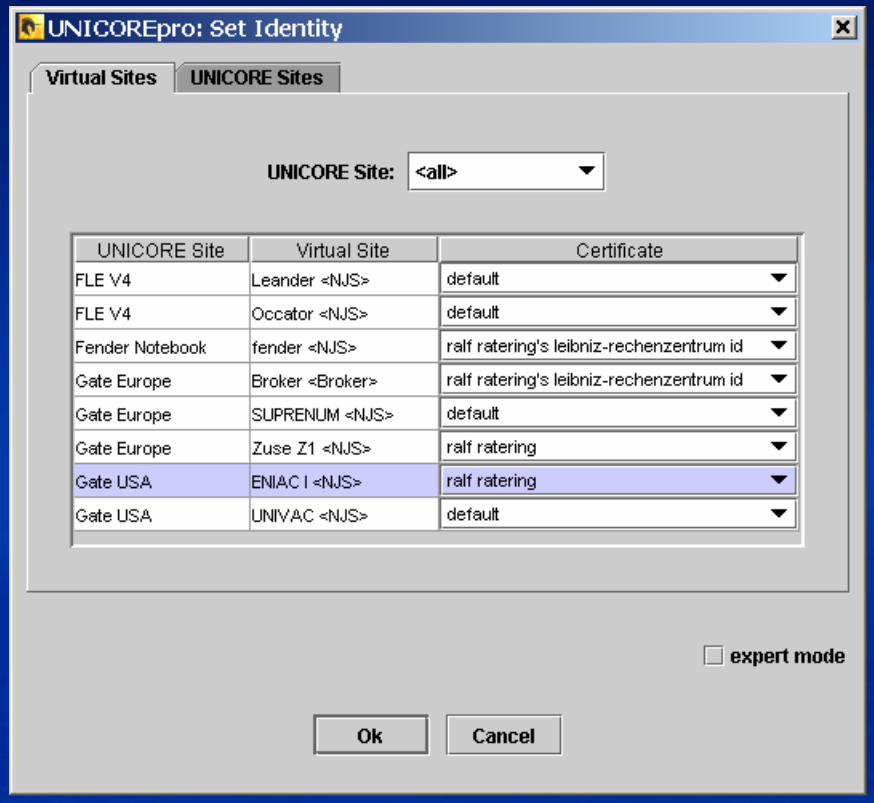




Configuration: Using Different Identities



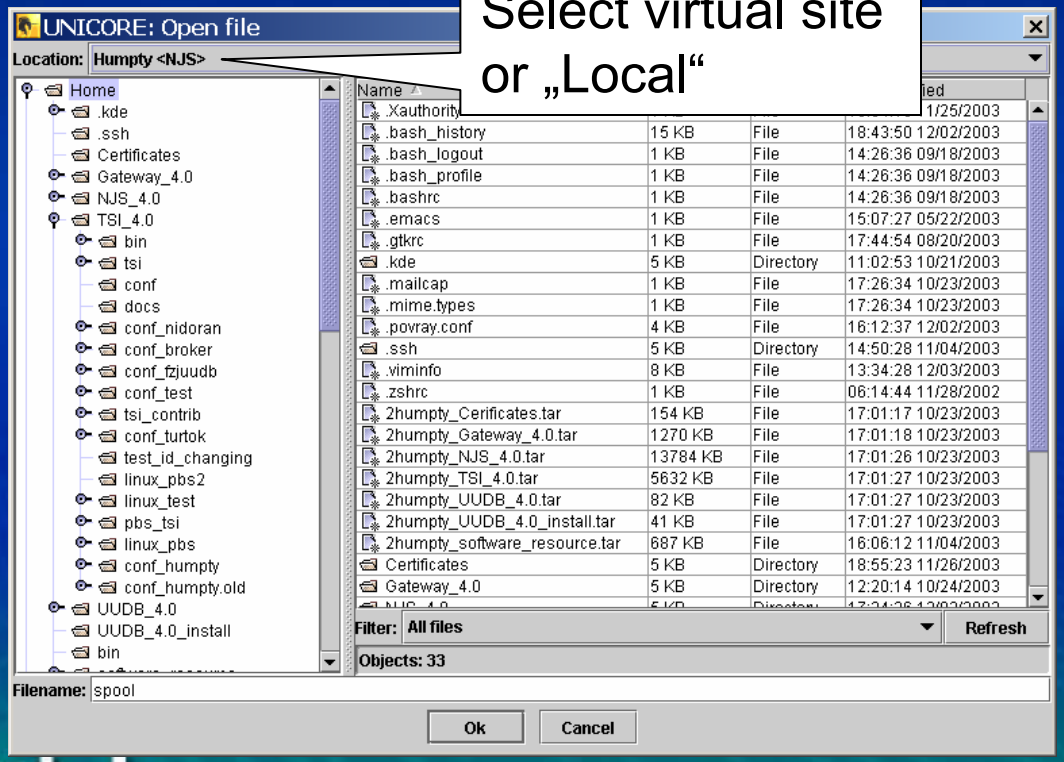
Key entries: Who am I?



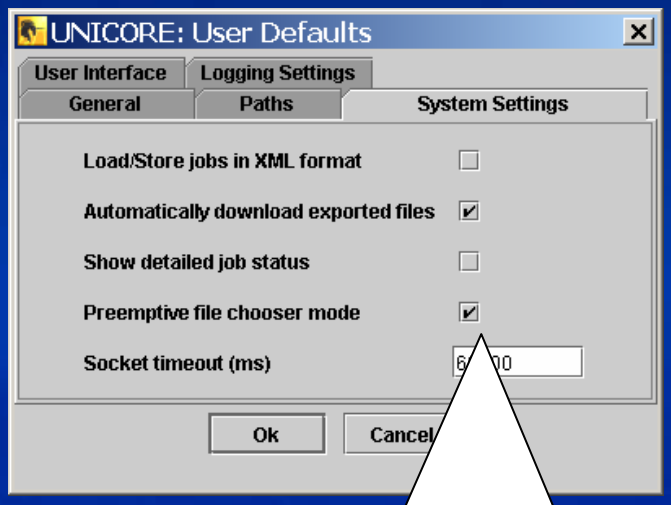
Using different identities

Browsing Remote File Systems

- **Remote File Chooser**
  - Used in Script Task, Command Task, for File Imports, Exports, etc.



Select virtual site or „Local“



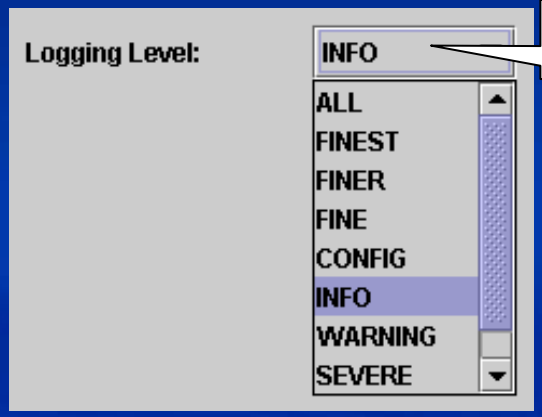
Preemptive file chooser mode will enhance performance on fast file systems

The Client Log

- „clientlog.txt“ or „clientlog.xml“
- Used by developers to figure out problems

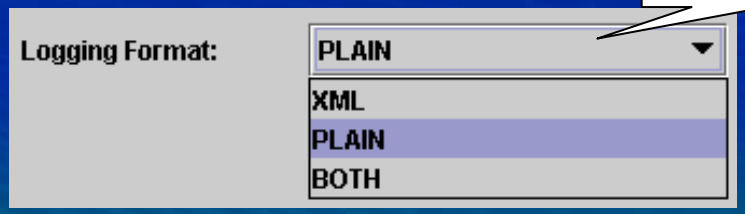
User Defaults->Paths:

User Defaults->Logging Settings:



INFO should be fine

Use PLAIN



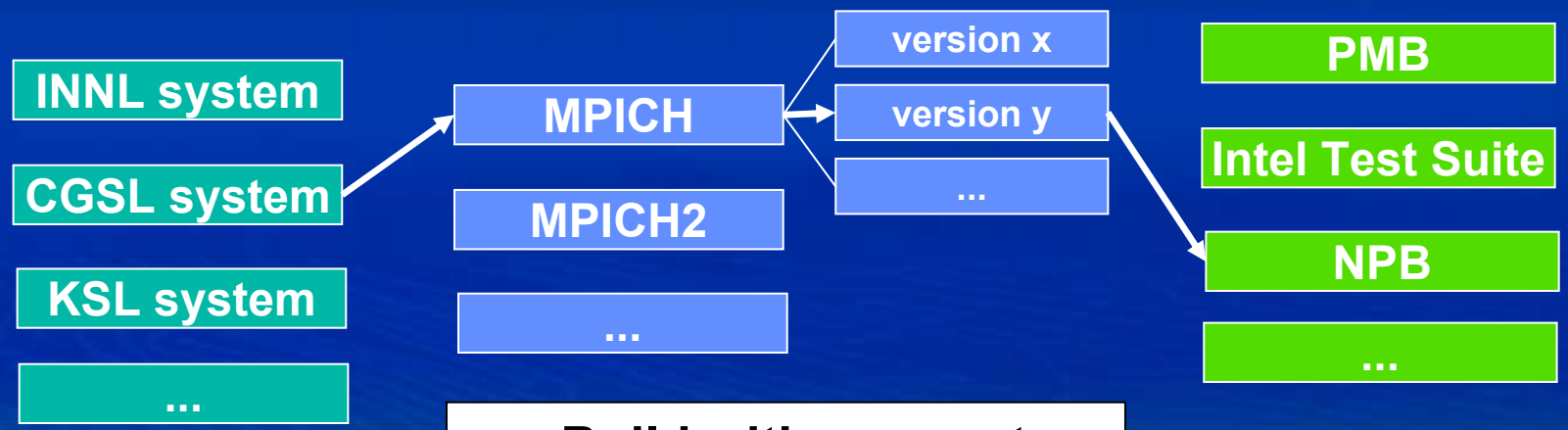
Enable under Windows, when no console is used

Starting the client re-visited

- client.jar in lib directory
  - start with .exe (Windows) or run script (Unix/Linux)
  - or: „`java -jar client.jar`“
- Command line options
  - Choose an alternative configuration directory:
    - `-Dcom.pallas.unicore.configpath=<mypath>`
  - Enable the security manager:
    - `-Dcom.pallas.unicore.security.manager`
  - Enable SOCKS proxy:
    - `-DsocksProxyHost="socks-proxy.isw.intel.com"`
    - `-DsocksProxyPort="1080"`

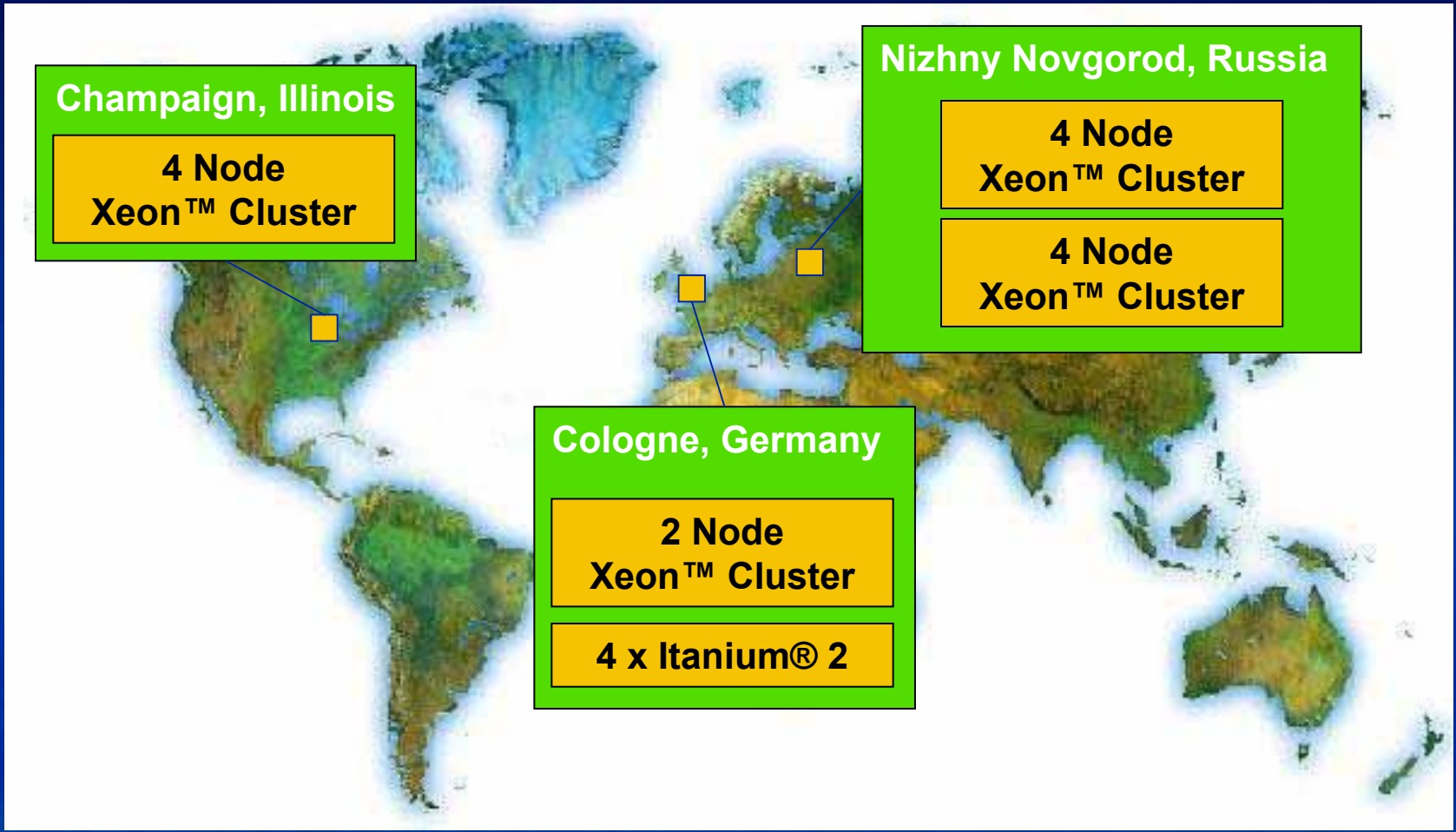
A real world Enterprise application: UNICORE inside Intel

- Software testing at Parallel and Distributed Solutions Division (PDSD)
  - Windows TSI port on server side
  - Complex existing testing environment



1. **Build with parameters**
2. **Run with parameters**
3. **Get result files**

Intel PDSD Grid



- UNICORE makes testing different versions on distributed systems a lot easier

Lessons learned…

- Security is negligible within intranet
  - Systems are protected by firewall
- Firewalls in the Intranet are a problem
  - Administrators have to open ports for every new NJS to the Gateways
- Users come and go
  - Managing user database and logins too complex
- Solutions
  - Open port range in firewalls
  - All testers use the same user certificate!!!

intel®

Summary

- Intel UNICORE Client offers an intuitive user interface to UNICORE Grids
- Client can be downloaded as Open Source at unicore.sourceforge.net
- Client functionality can be extended through plug-in interface