The UNICORE logo is displayed in a large, blue, pixelated font. The word "UNICORE" is centered within a blue horizontal bar. The letter "O" is replaced by a grid of small blue squares. Above the bar, there is a faint, light blue graphic of a building with a dome. Below the bar, there is a large, faint, light blue grid pattern that forms a wide, shallow bowl shape.

UNICORE

## **Server Installation Tutorial**

**Michael Rambadt, Forschungszentrum Jülich - ZAM**



# Outline

- ▶ UNICORE Quickstart bundle
- ▶ UNICORE Server
  - ▶ Overview (Packages, Components, Prerequisites)
  - ▶ Gateway
    - Installation, basic configuration, maintenance
  - ▶ Network Job Supervisor (NJS)
    - Installation, basic configuration, maintenance
    - Incarnation Data Base (IDB)
    - UNICORE User Data Base (UUDB)
  - ▶ Target System Interface (Installation, Configuration)
- ▶ UNICORE Client
  - ▶ Installation



# Server Packages

- ▶ UNICORE development page:

<http://unicore.sourceforge.net/>

- ▶ UNICORE download page:

<http://unicore.sourceforge.net/>

- ▶ Latest UNICORE releases

- ▶ Quickstart bundle: 1.1.0
- ▶ Gateway: 4.1.1\_build2
- ▶ NJS: 4.6.1\_build2
- ▶ TSI: 4.1.2\_build1
- ▶ UUDB: 1.0.0.tar.gz



# Prerequisites

- ▶ Login to `zam461.zam.kfa-juelich.de`

- ▶ `ssh unitu**@zam461.zam.kfa-juelich.de`

- ▶ `cd $HOME/unitu**`

- ▶ `ls -l`

```
drwxr-xr-x 2 unitu01 users 4096 2005-10-31 14:39 bin
drwxr-xr-x 2 unitu01 users 4096 2005-10-31 14:39 Documents
drwxr-xr-x 2 unitu01 users 4096 2005-10-31 14:39 public_html
-rw-r--r-- 1 unitu01 users 3958404 2005-10-31 15:01 unicolor_demo_1_1.tar.gz
```



# UNICORE Quickstart bundle

- ▶ Unpack unicore\_demo\_1\_1.tar.gz
  - ▶ `tar xzf unicore_demo_1_1.tar.gz`
- ▶ Run the install script
  - ▶ `./install.py`
  - ▶ The config script uses Python
- ▶ Start UNICORE components as follows:



# UNICORE Quickstart bundle

- ▶ Gateway:
  - ▶ cd gateway/conf
  - ▶ ../bin/start\_gateway &
- ▶ NJS
  - ▶ cd njs/conf
  - ▶ ../bin/start\_njs &
- ▶ TSI
  - ▶ cd tsi/conf
  - ▶ ../bin/start\_tsi &



# UNICORE Quickstart bundle

- ▶ Add users to the UNICORE User Database (UADB)
  - ▶ `cd uadb`
  - ▶ `bin/add user.pem unixname`
  - ▶ Where `user.pem` stands for the public key of the user you wish to add
  - ▶ Where `unixname` stands for „unitu\*\*“
  - ▶ Select your own user certificate
  - ▶ If you don't have any certificate yet you can ask for one at <https://projects-ca.fz-juelich.de>



# UNICORE Quickstart bundle

- ▶ Add users to the UADB (cont.)
  - ▶ For testing you might use  
`$HOME/unit**/cert/fake_usercert.pem`
  - ▶ `cd $HOME/unitu**/uadb`
  - ▶ `bin/add ../cert/fake_usercert.pem unitu**`
  - ▶ Mapping from user certificate to user's login on the target system (zam461)

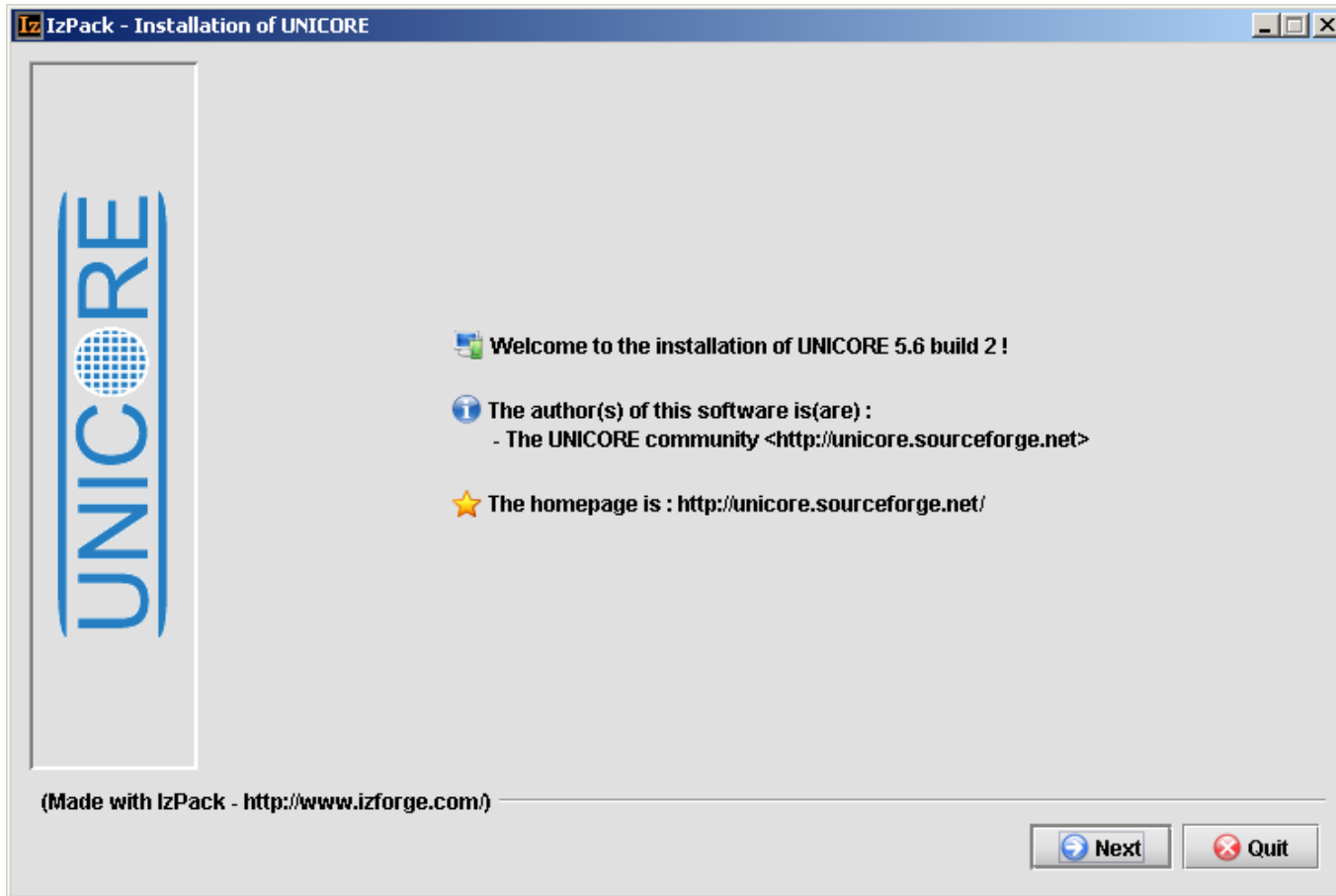


# UNICORE Client installation

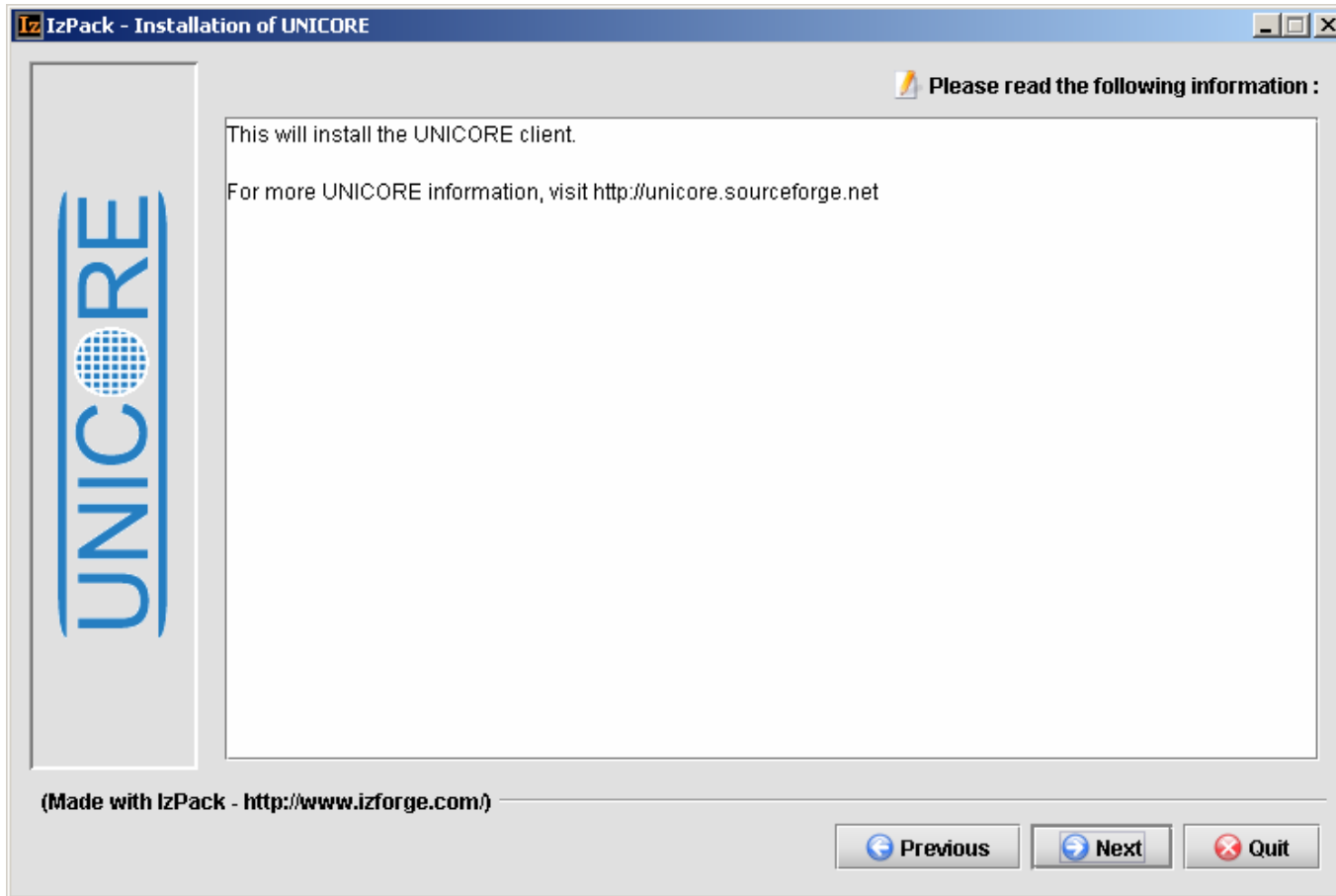
- ▶ Download `client_5.6_2_installer.jar` (~9 MB) from <http://unicore.sourceforge.net/> to your laptop
- ▶ Windows: double click on `client_5.6_2_installer.jar`
- ▶ Linux: `java -jar client_5.6_2_installer.jar`
- ▶ Prerequisites: SUN Java JRE >1.4.x



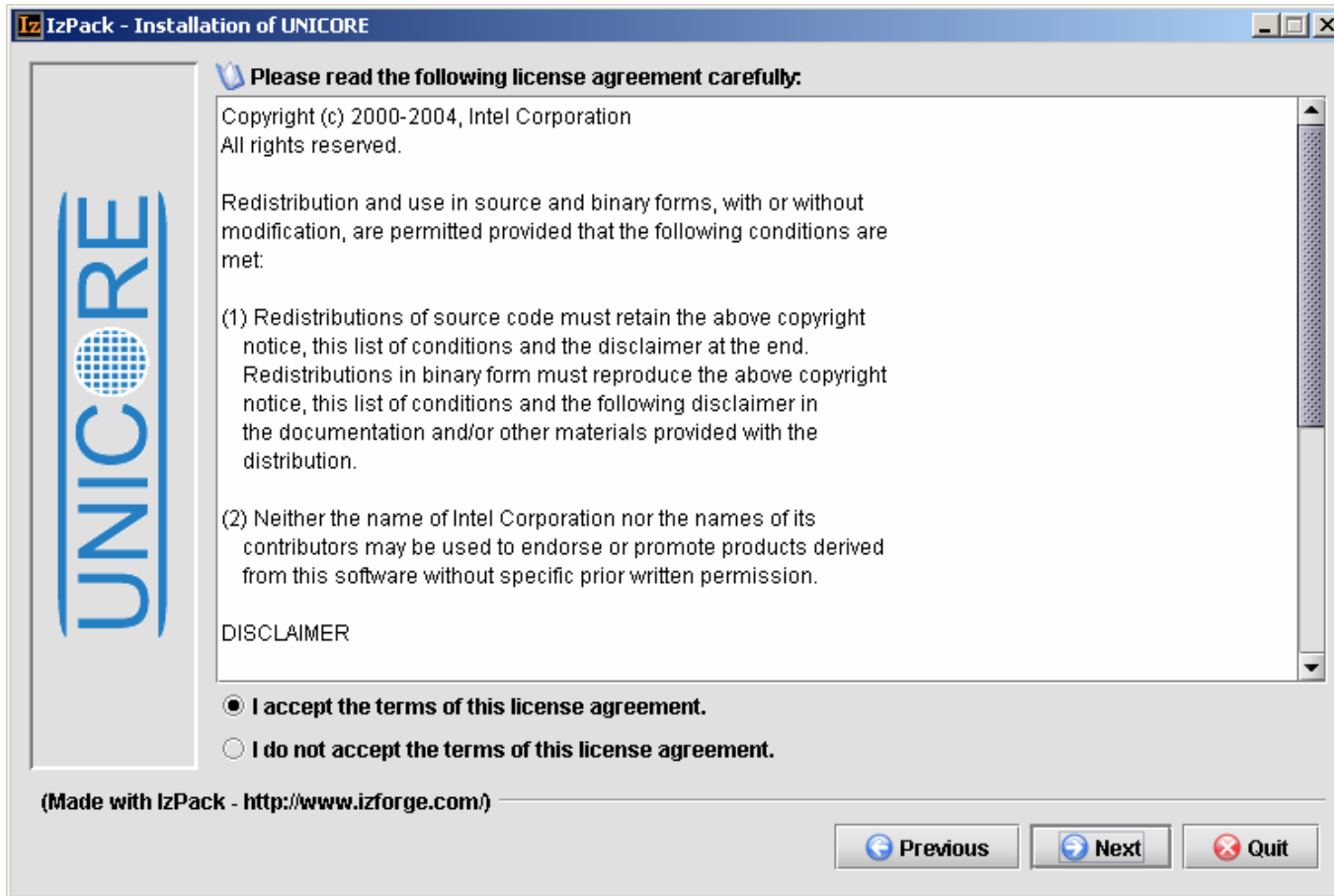
# UNICORE Client installation<sub>(cont.)</sub>



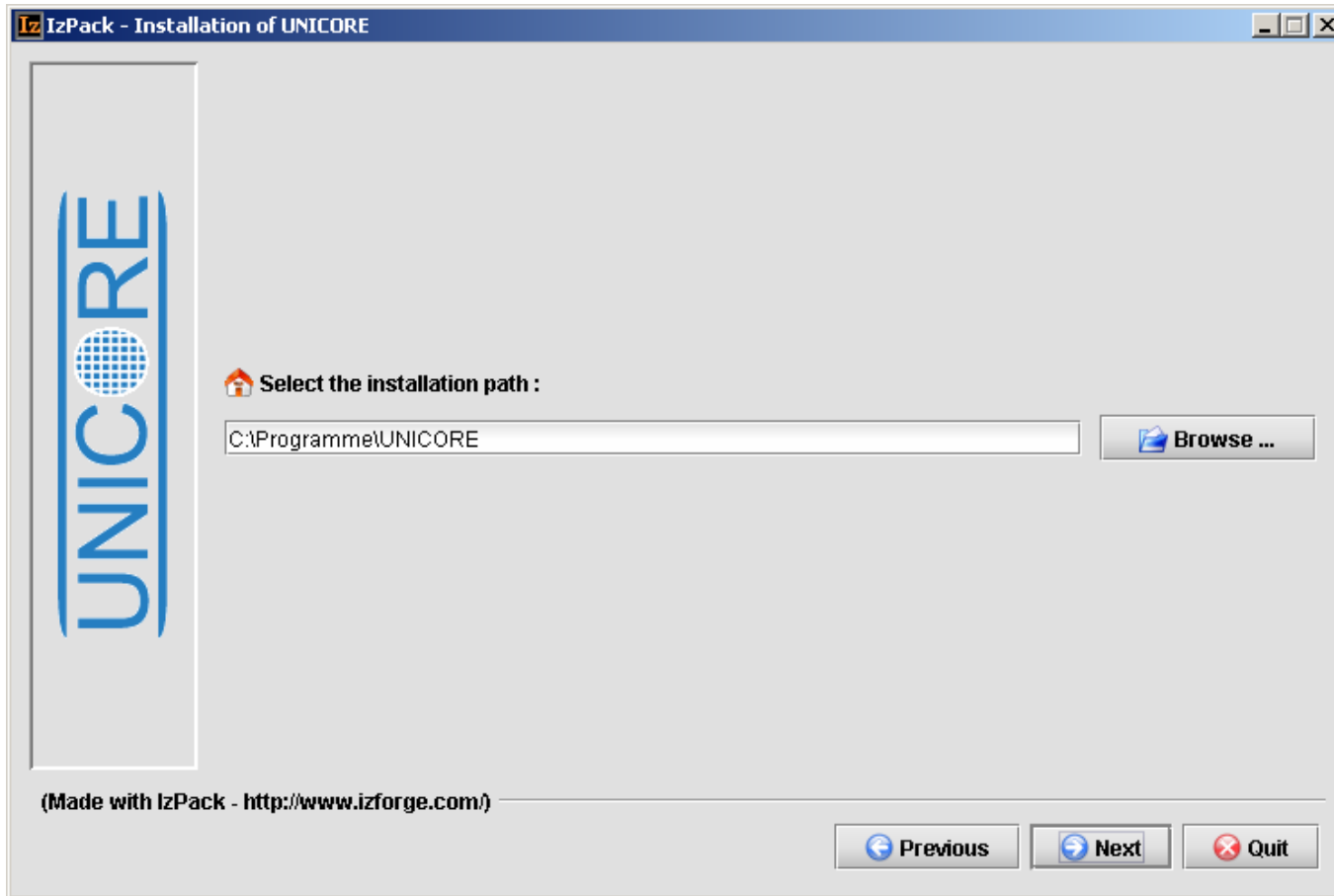
# UNICORE Client installation<sub>(cont.)</sub>



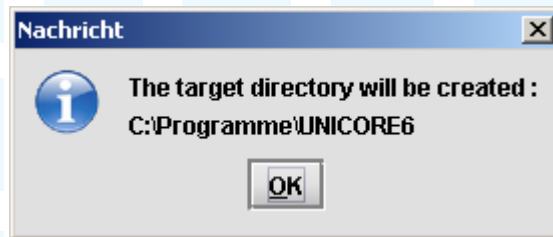
# UNICORE Client installation(cont.)



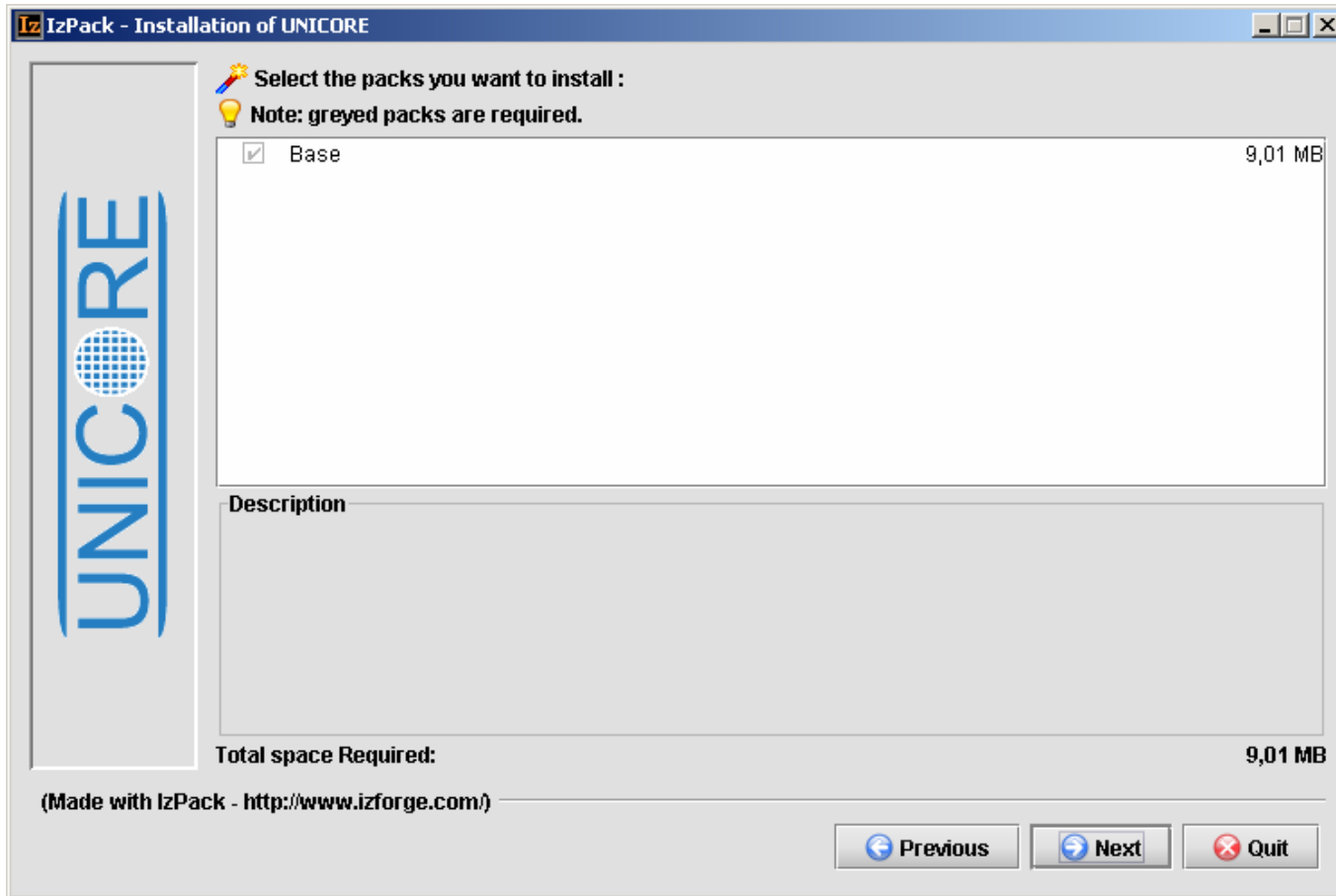
# UNICORE Client installation (cont.)



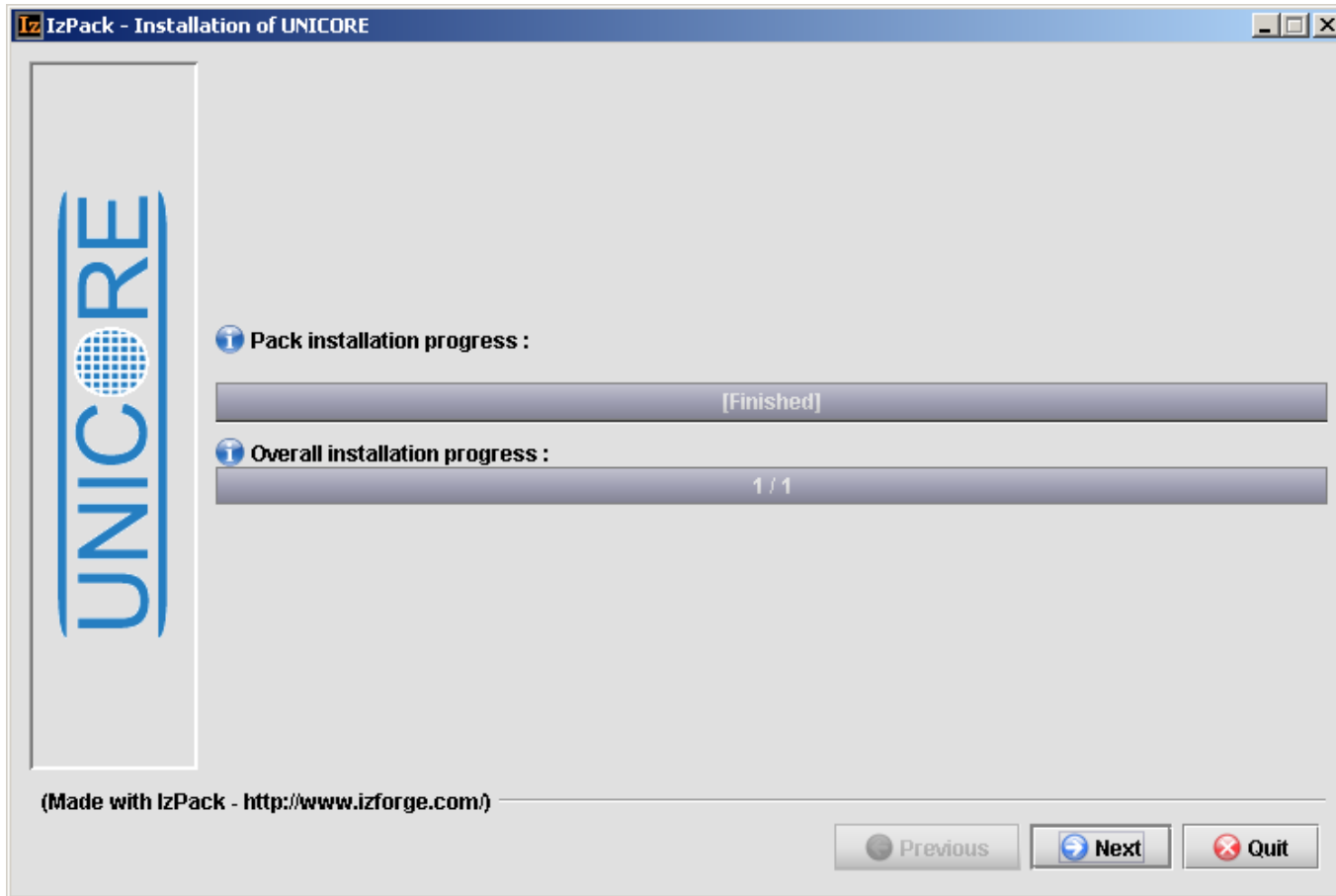
# UNICORE Client installation<sub>(cont.)</sub>



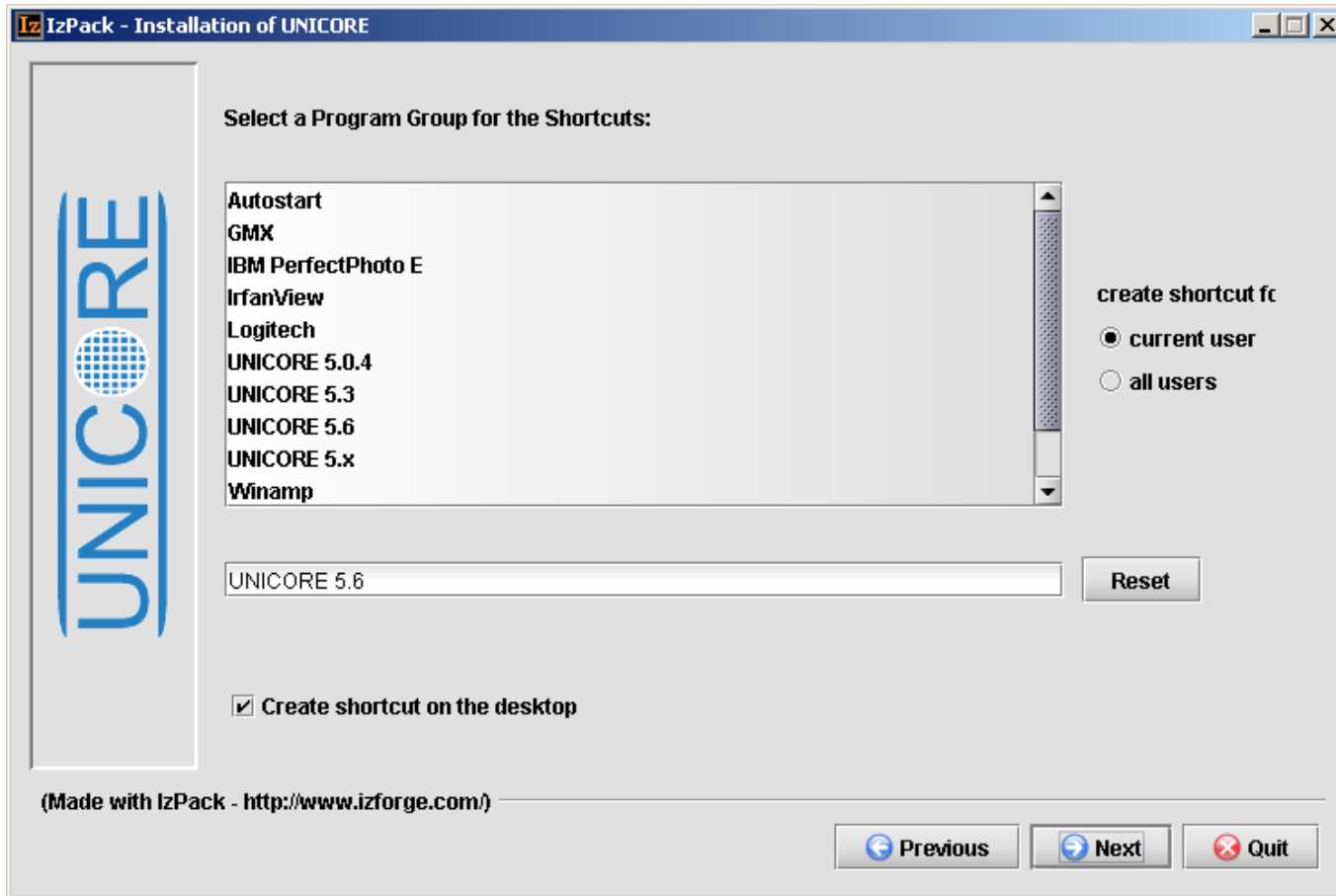
# UNICORE Client installation<sub>(cont.)</sub>



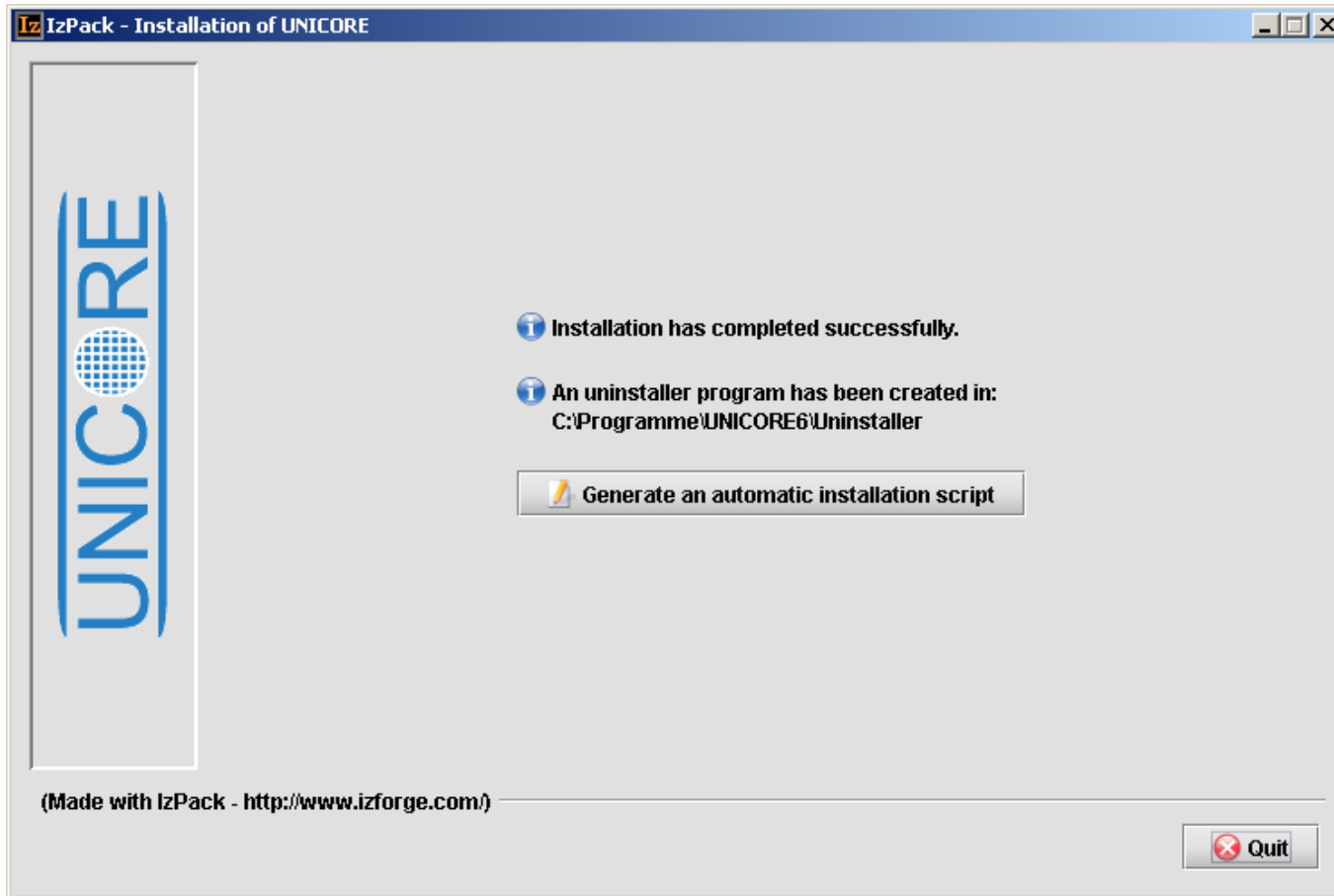
# UNICORE Client installation<sub>(cont.)</sub>



# UNICORE Client installation<sub>(cont.)</sub>



# UNICORE Client installation (cont.)

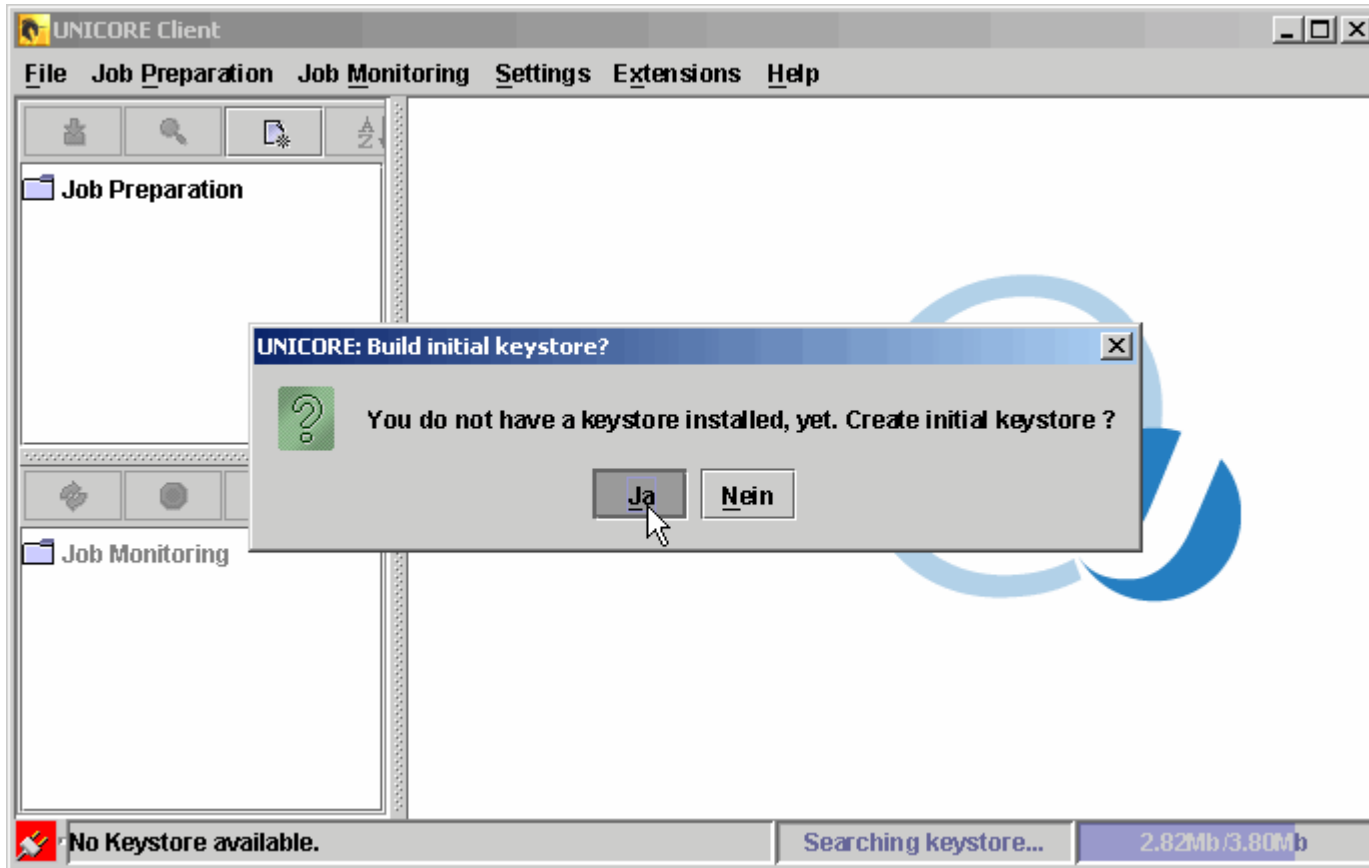


# Starting UNICORE client

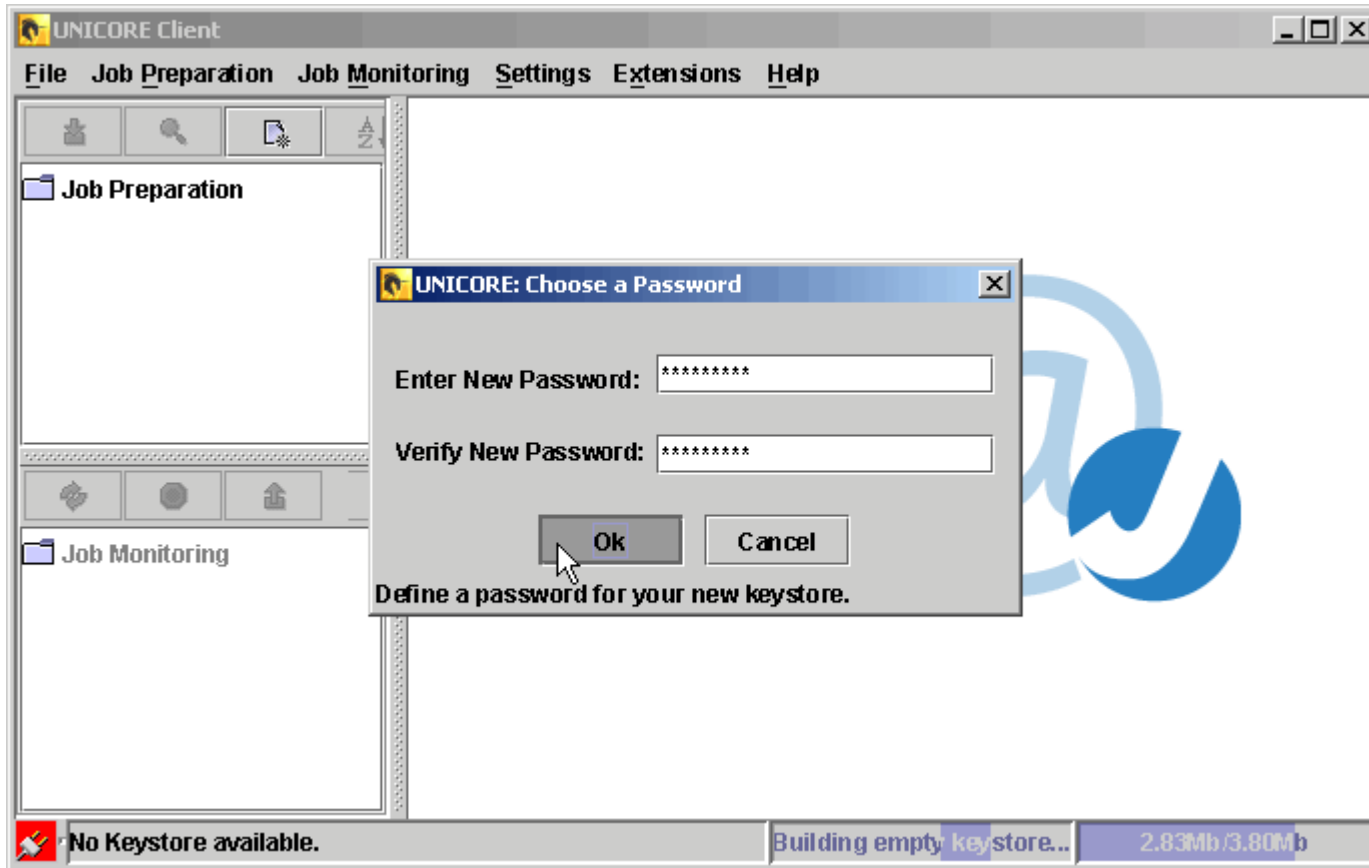
- ▶ Windows:
  - ▶ Double click on UNICORE symbol on the Desktop
- ▶ Linux:
  - ▶ Change to installation directory
  - ▶ Type `unicore`



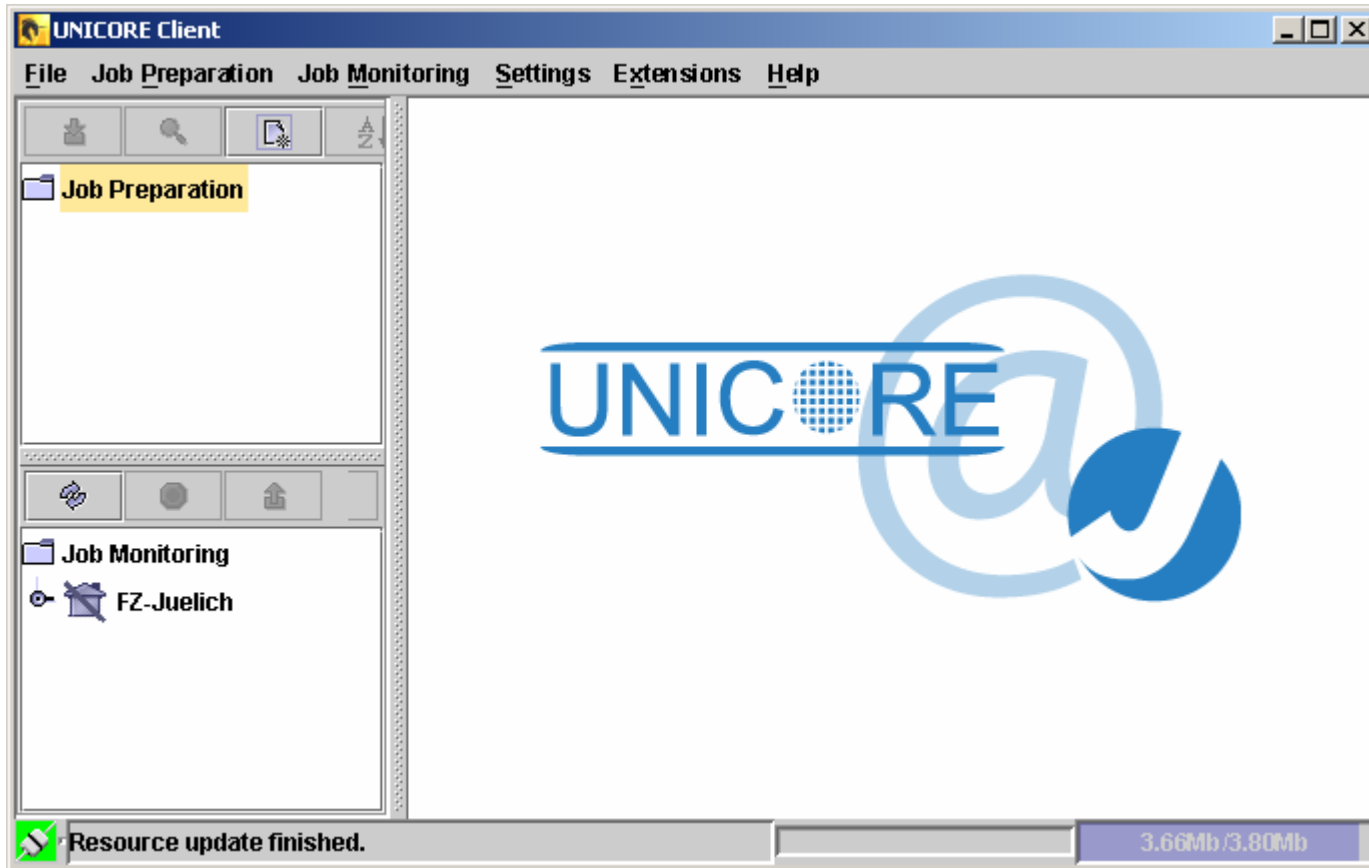
# Starting the UNICORE Client



# Starting the UNICORE Client



# Starting the UNICORE Client

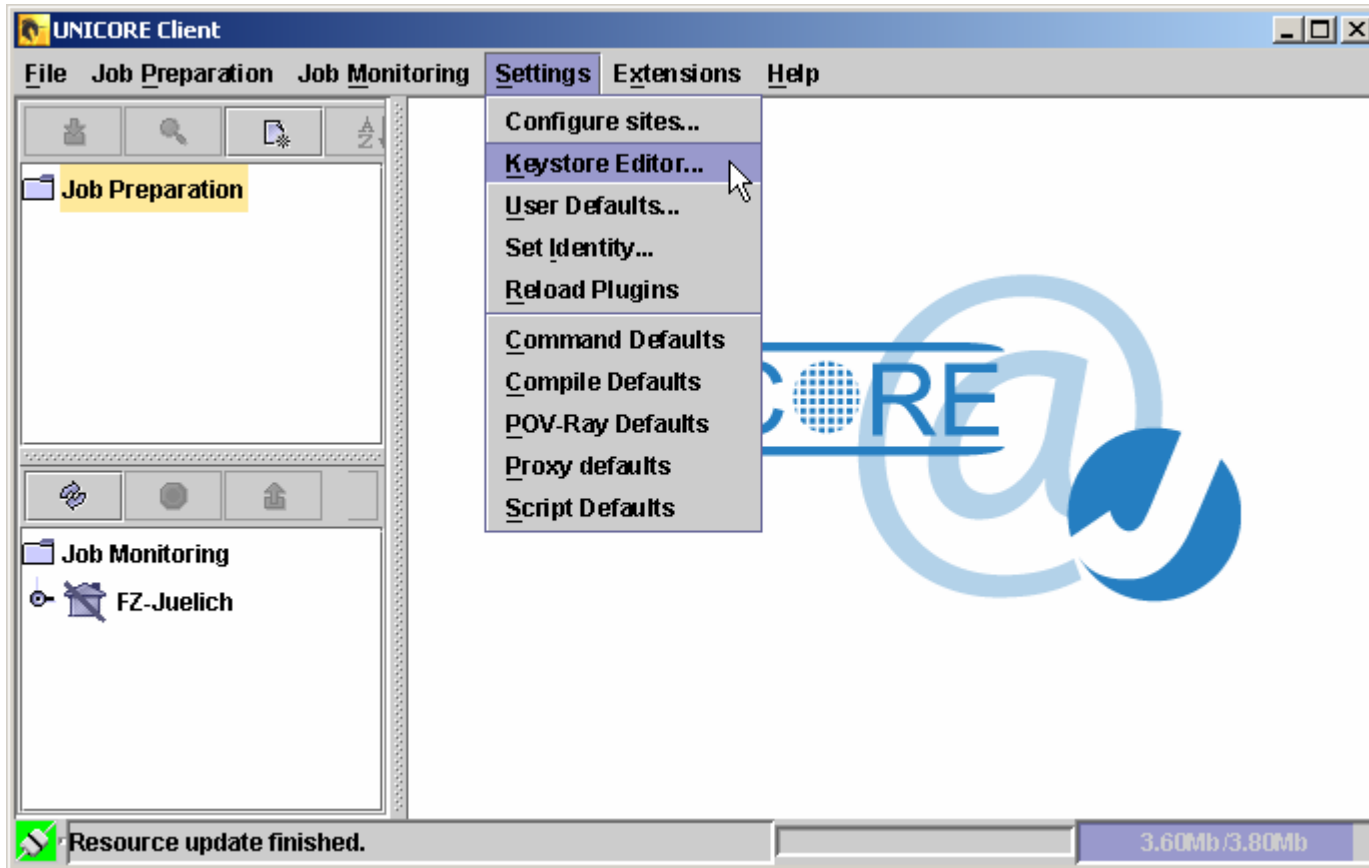


# Importing Quickstart Bundle certificate into UNICORE

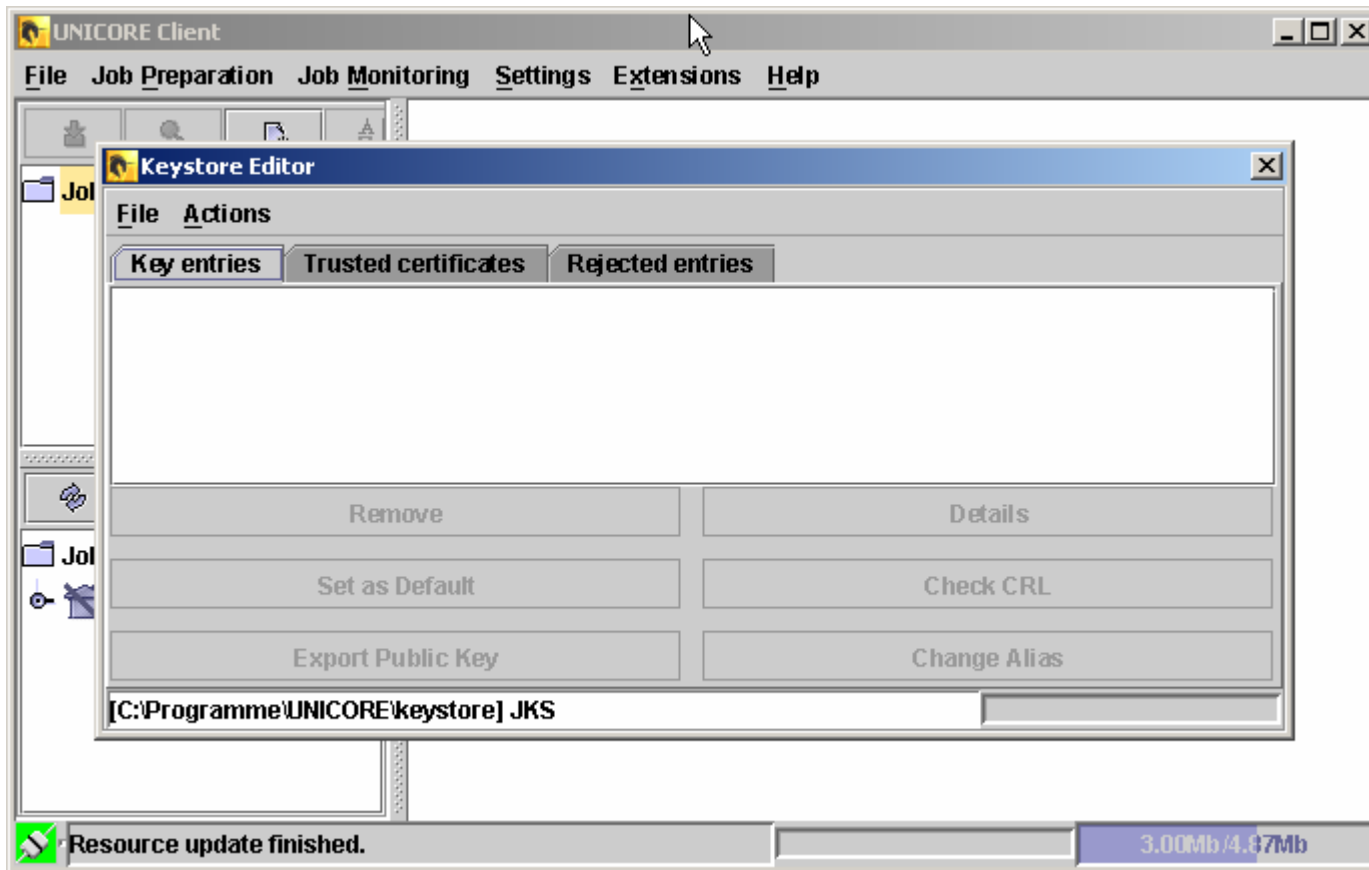
- ▶ Just for testing!
- ▶ Copy the file `$HOME/unitu/cert/njs_identity.p12` to your laptop
- ▶ In the UNICORE client:  
Select Settings -> Keystore Editor
  - ▶ Select Actions -> Import Keystore
  - ▶ Select `njs_identity.p12` , password is „the!njs“
  - ▶ Choose „yes“ when asked wether this should be set as default



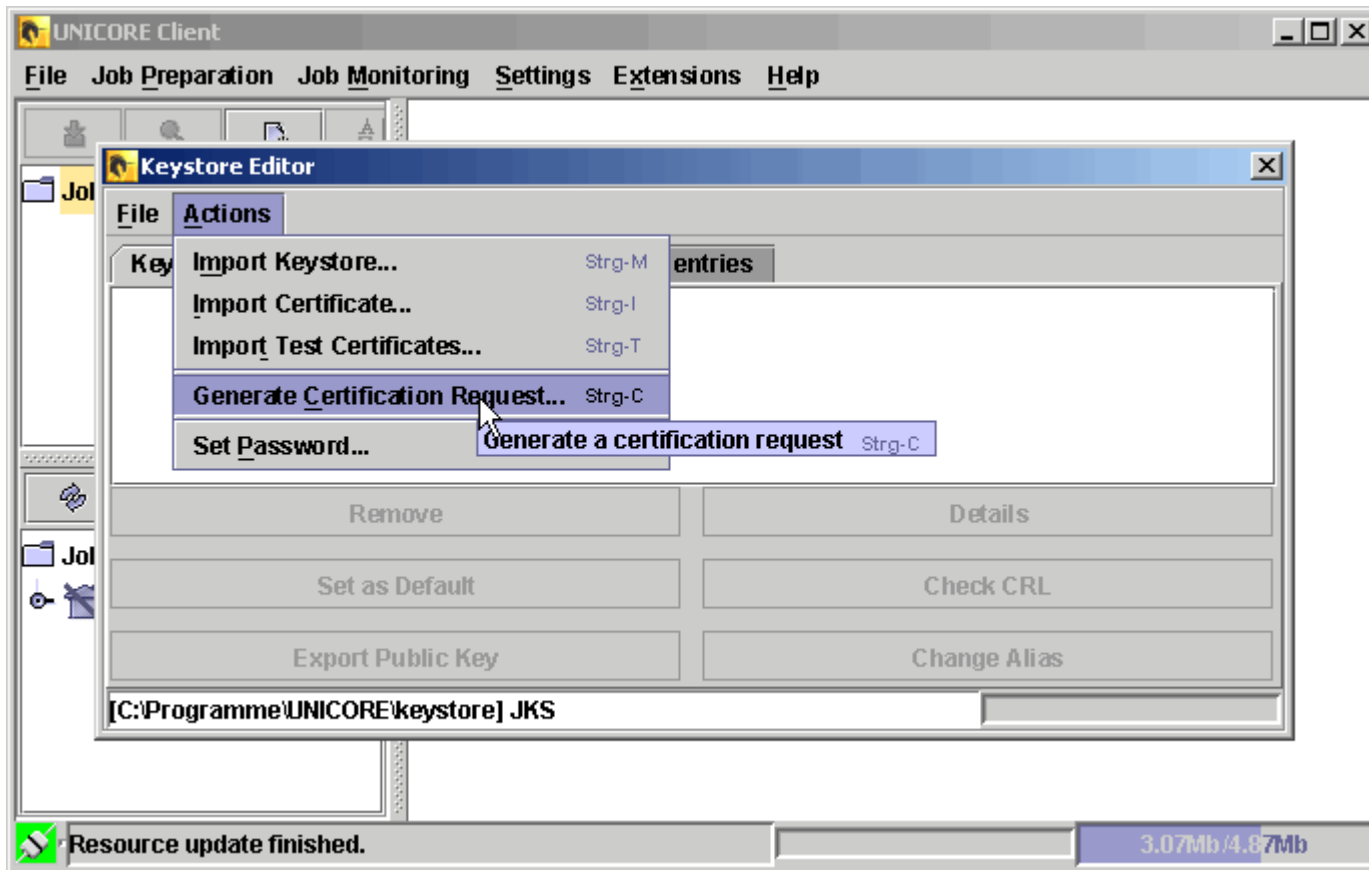
# Generate user certificate request



# Generate user certificate request



# Generate user certificate request



# Generate user certificate request

**Generate certification request**

*individual*

**Common Name: \***

**Email: \***

*is associated with*

**Organization: \***

**Organizational Unit:**

*is located*

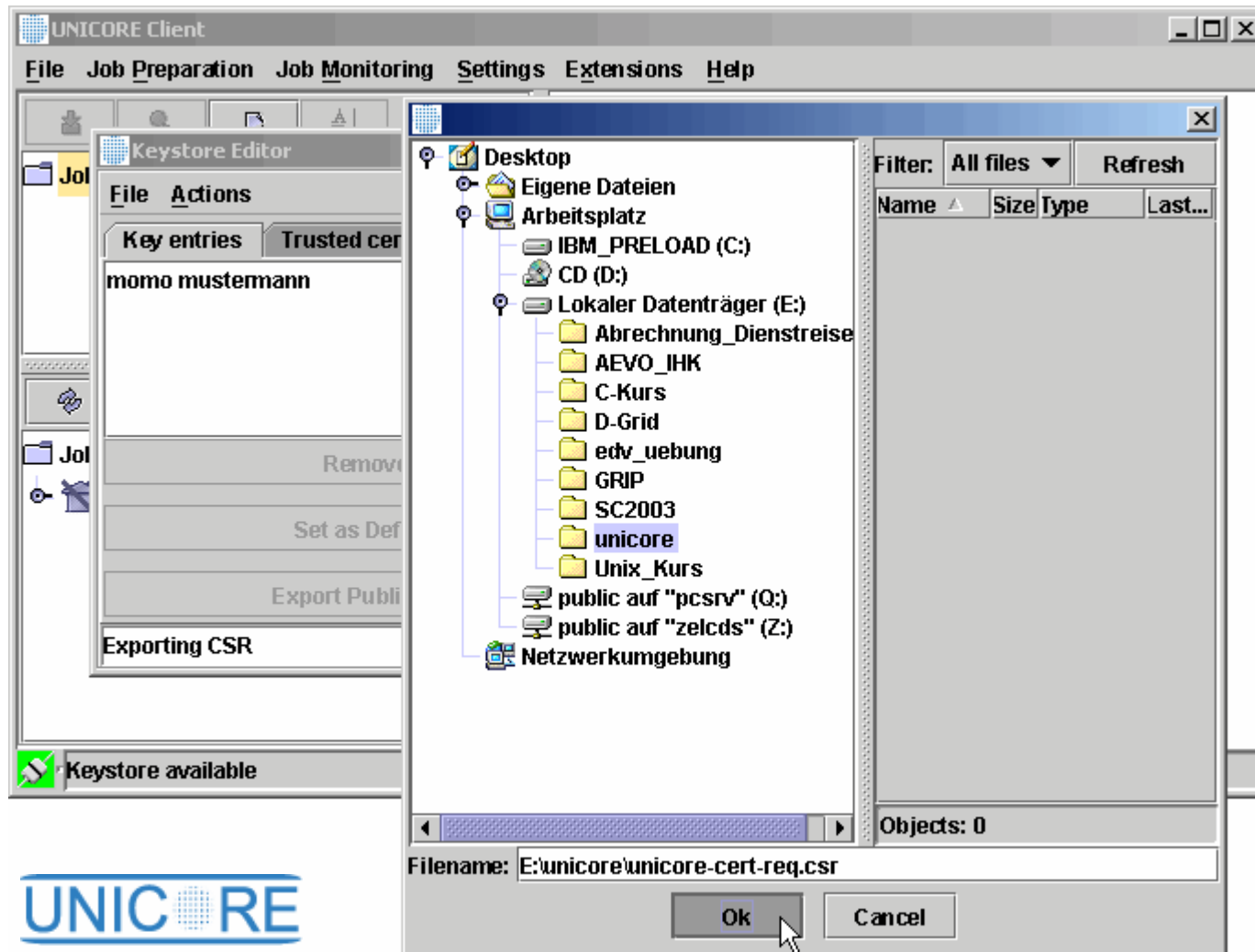
**Locality:**

**State:**

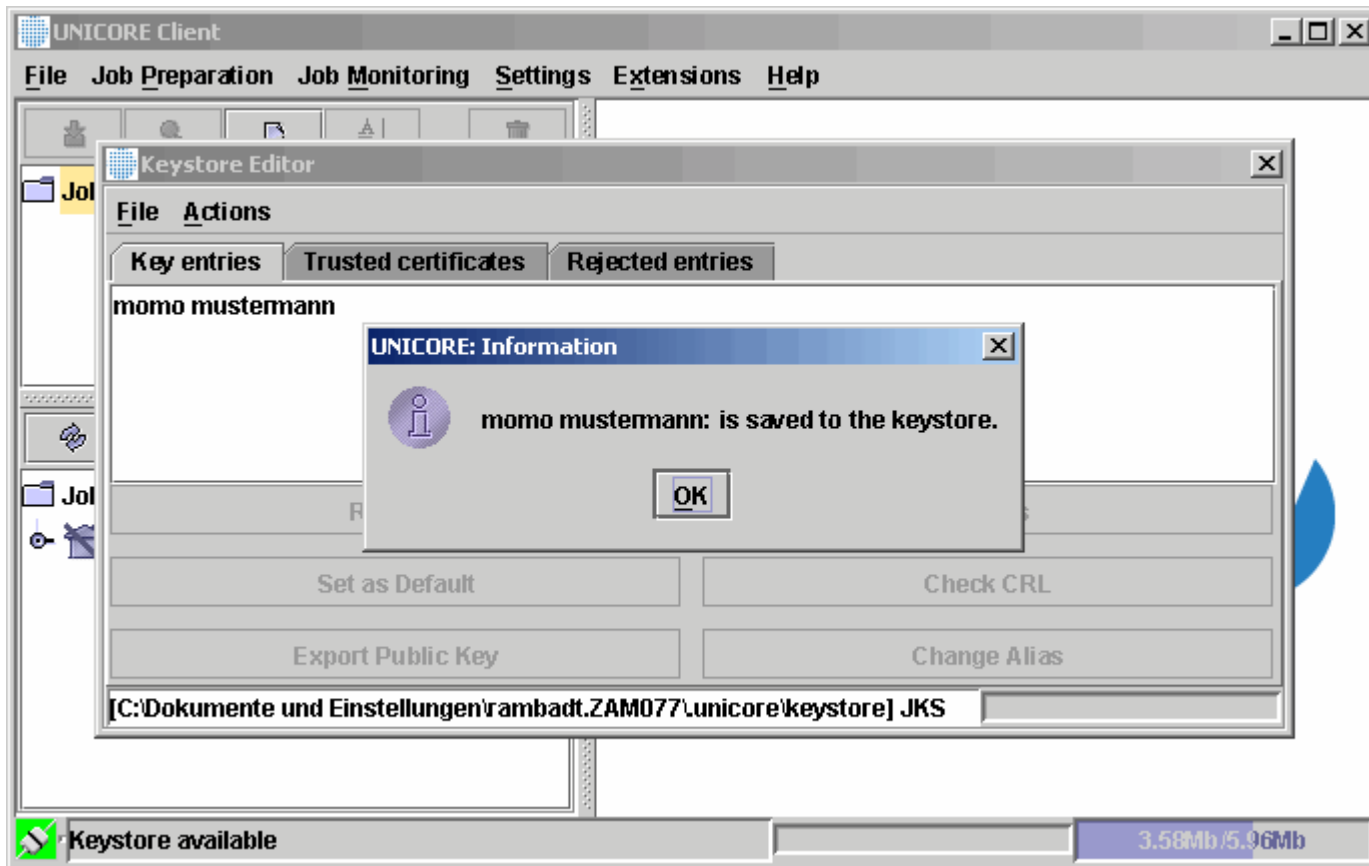
**country(shortcut) \***

**OK** **Cancel**

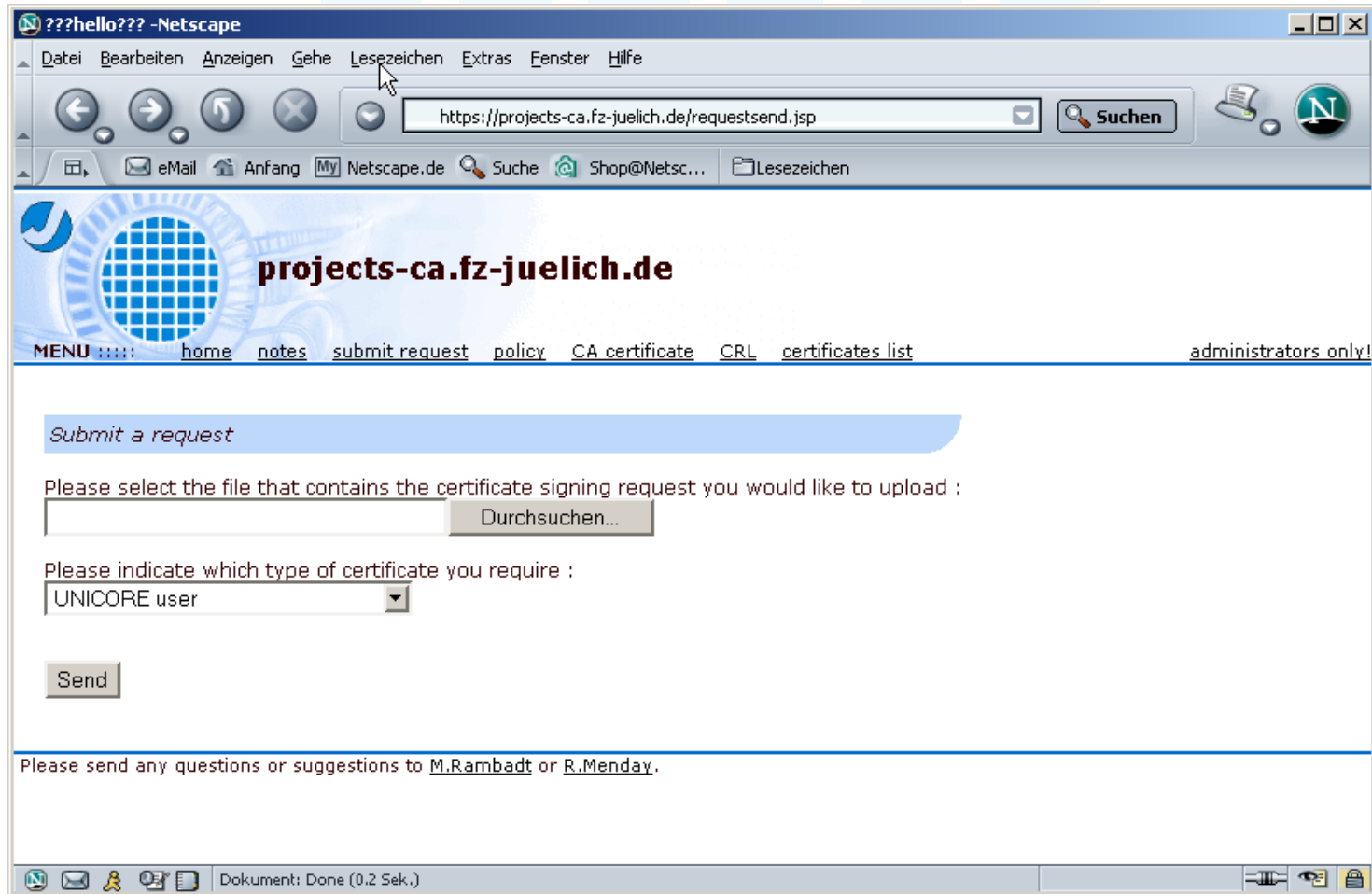
# Generate user certificate request



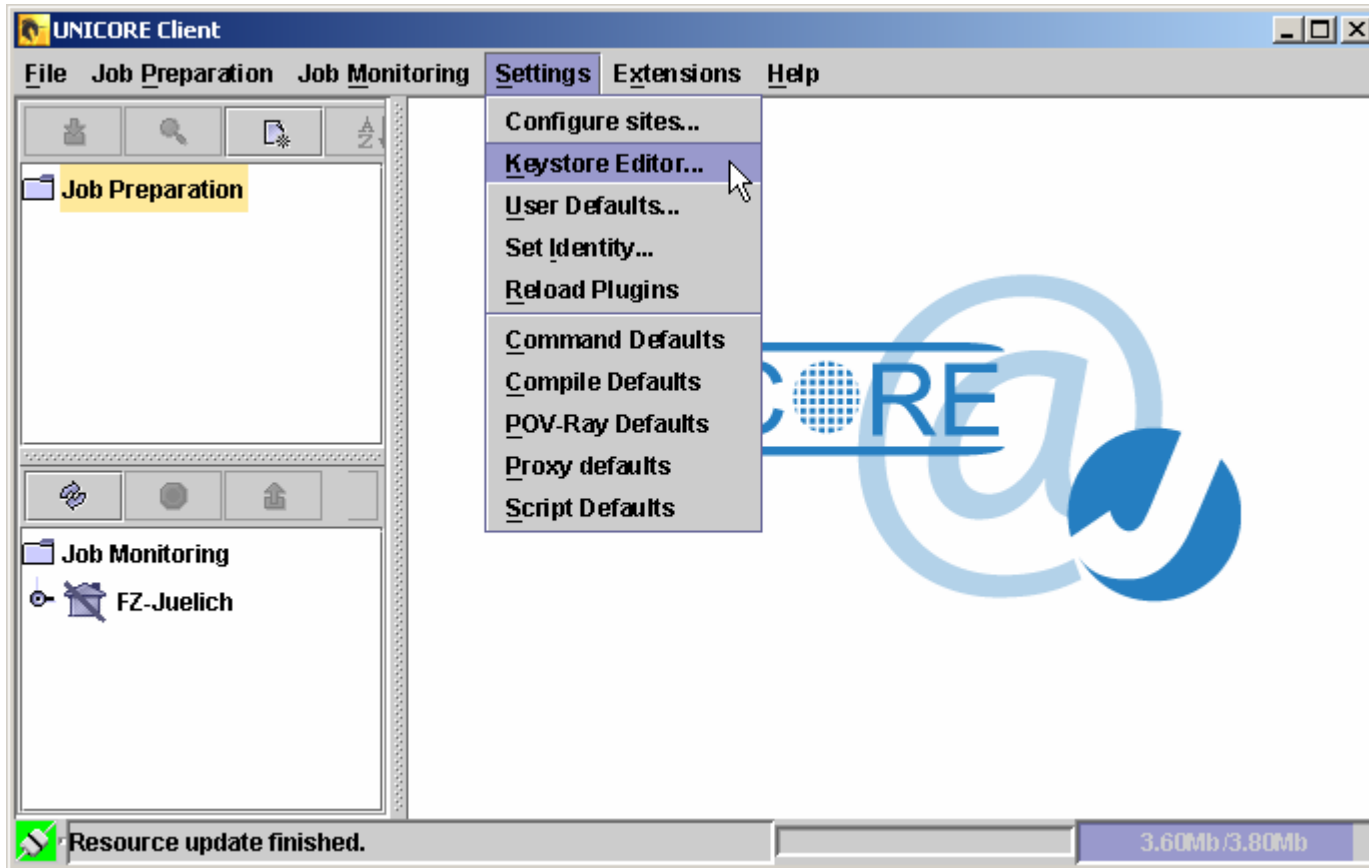
# Generate user certificate request



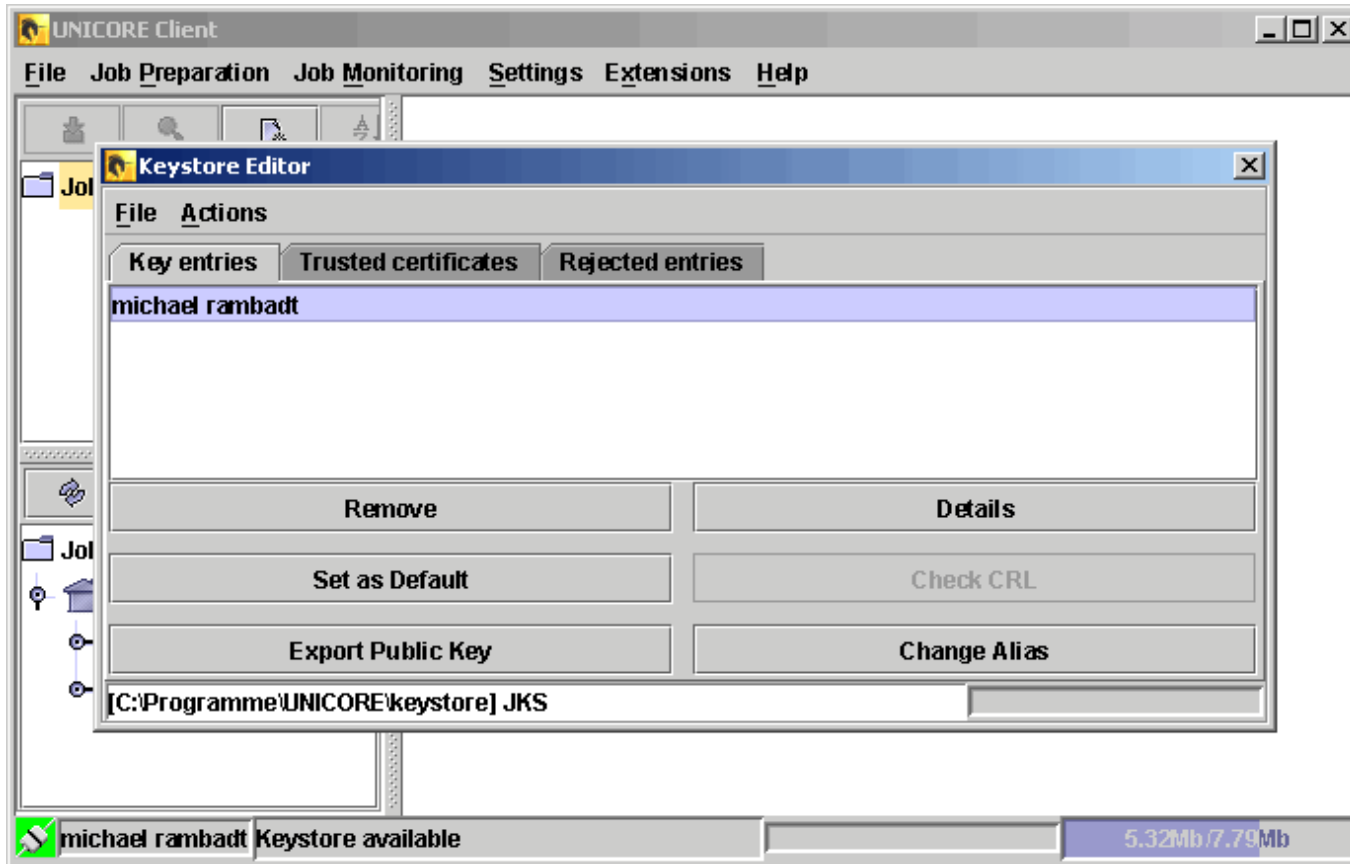
# https://projects-ca.fz-juelich.de



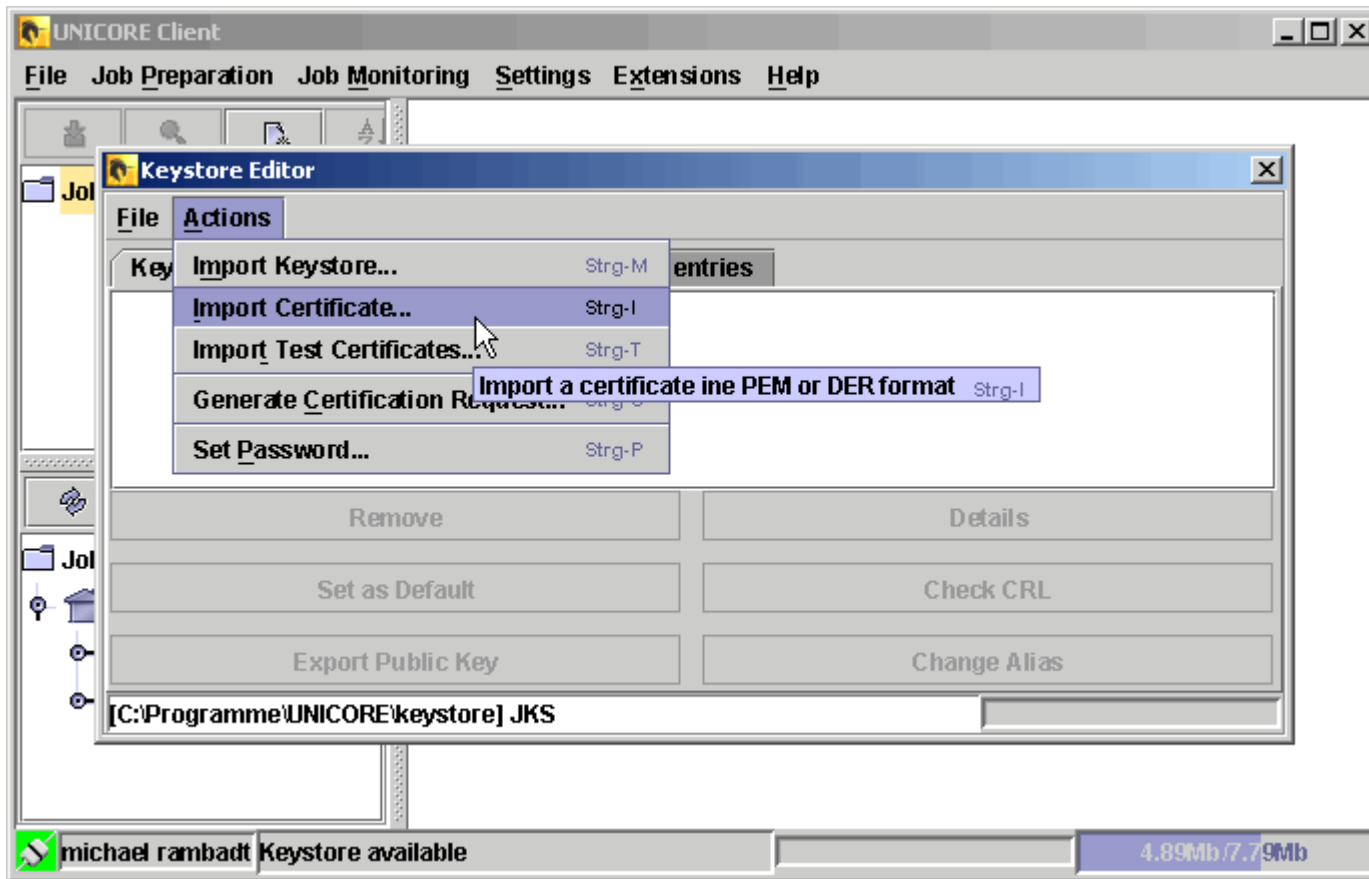
# Import certificate into UNICORE client



# Import certificate into UNICORE client



# Import certificate into UNICORE client



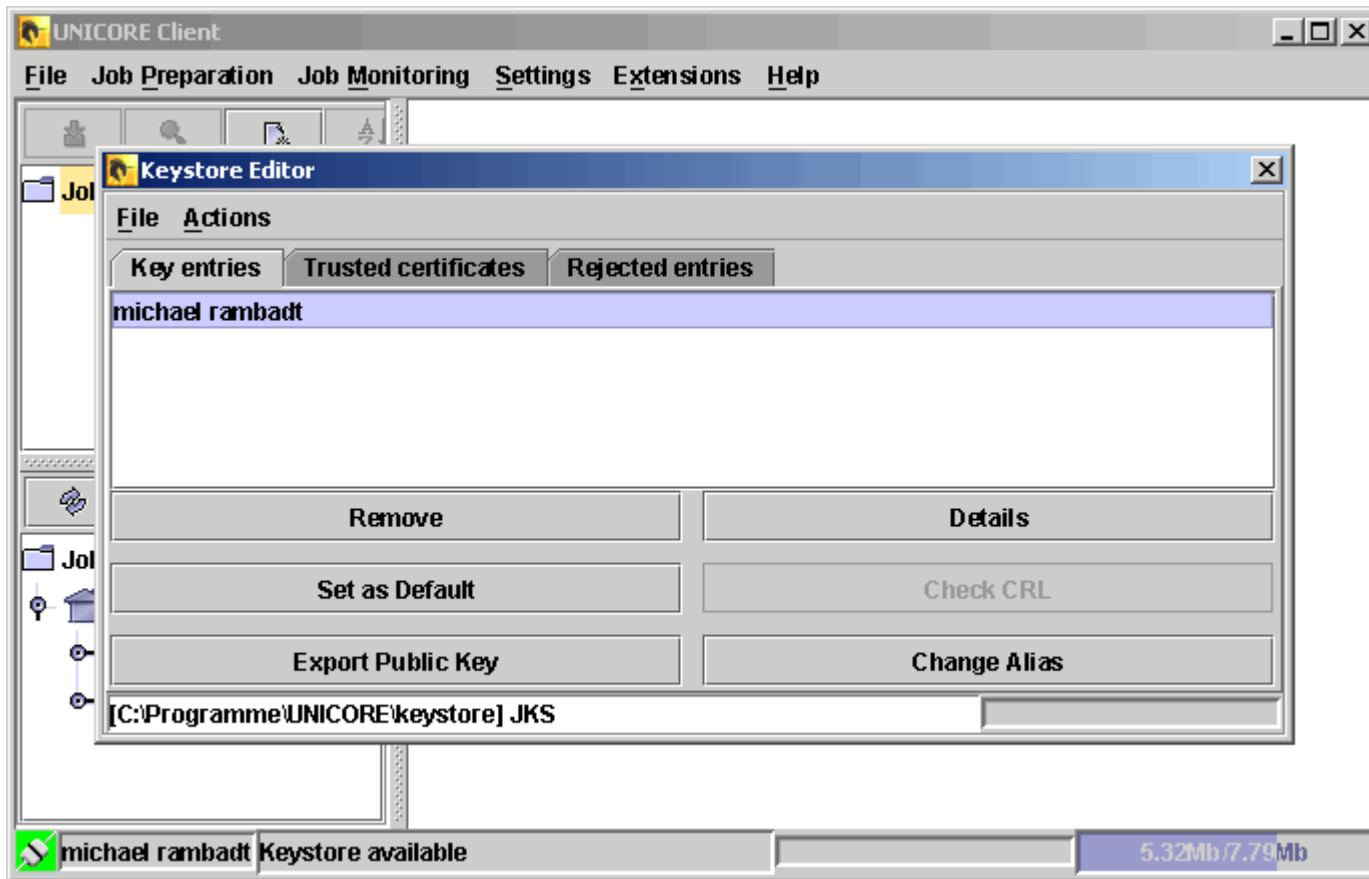
# Import certificate into UNICORE client

The screenshot shows the UNICORE Client interface with the 'Import a certificate' dialog box open. The dialog displays a file explorer view of the local drive with the 'Certificates' folder selected. A table lists files in the folder:

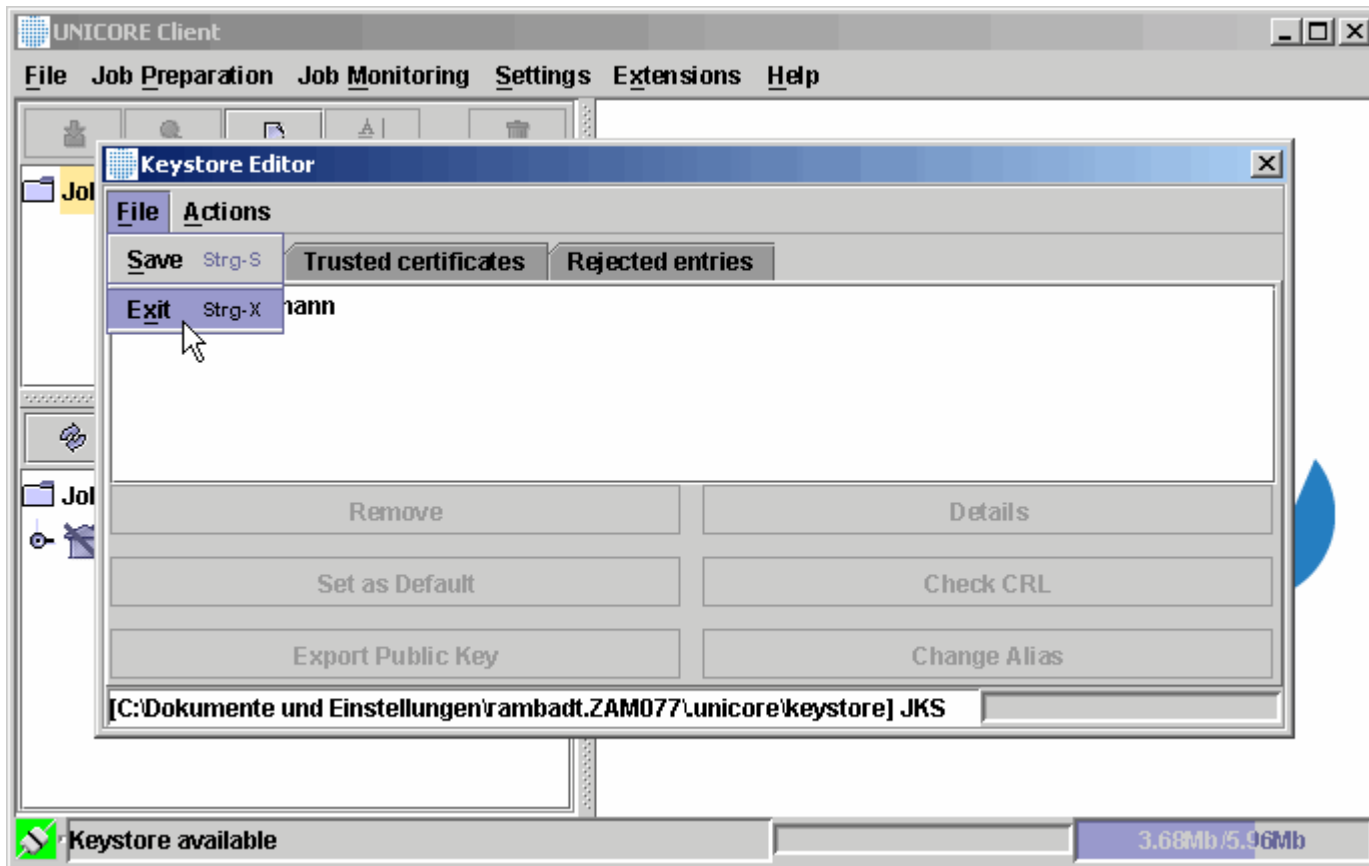
Name	Size	Type	Last M...
MeinZertifikat.pem	2 KB	PEM-Datei	11:31:...
request		Dateiordner	11:09:...
unicore-ca-fz-juelich.pem	2 KB	PEM-Datei	09:39:...

The 'Filename' field at the bottom of the dialog shows 'C:\UNICORE\Certificates\MeinZertifikat.pem'. The 'Filter' is set to 'Certificates in PEM format (\*.pem)' and 'Objects: 6' are displayed. The 'Ok' button is highlighted.

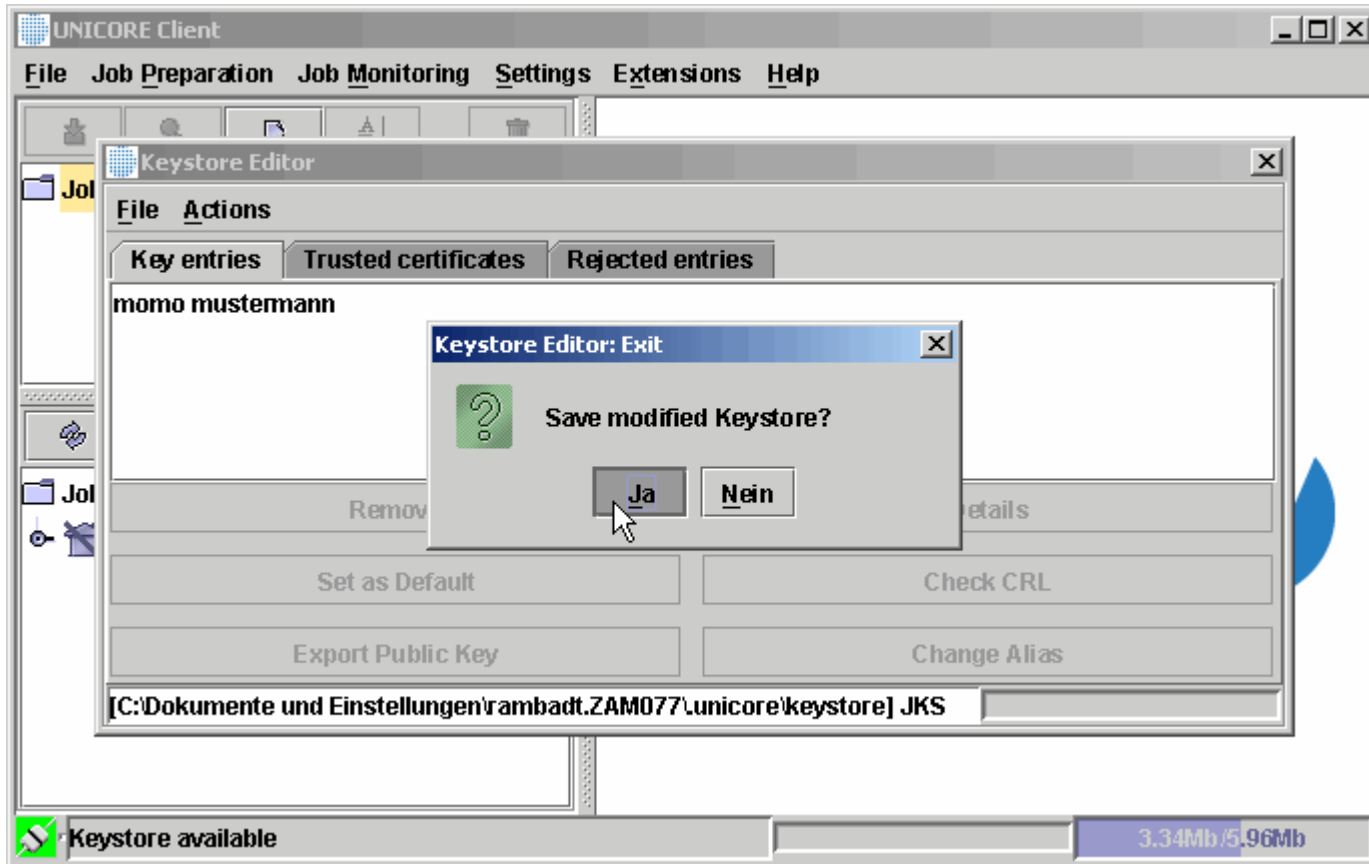
# Import certificate into UNICORE client




# Import certificate into UNICORE client



# Import certificate into UNICORE client



# Import UNICORE sites

- ▶ Copy gateways.xml from zam461:  
\$HOME/unitu\*\* to your laptop
- ▶ Select Settings -> User Defaults...in the UNICORE client
- ▶ Click on  to load the gateways.xml
- ▶ Restart the Client



# Ports used in this installation

- ▶ Client – GW : 911(0-9)
  - ▶ \$HOME/unitu\*\*/gateway/conf/gateway.properties: **gw.port= ~~4004~~**
  - ▶ \$HOME/unitu\*\*/gateway/conf/gateways.xml: ~~4004~~
- ▶ GW – NJS: 92(1-9)1 and 9301
  - ▶ \$HOME/unitu\*\*/gateway/conf/conections: ~~4444~~
  - ▶ \$HOME/unitu\*\*/njs/conf/njs.properties: **njs.gateway.port= ~~4444~~**
- ▶ NJS Admin: 92(1-9)2 and 9302
  - ▶ \$HOME/unitu\*\*/njs/conf/njs.properties: **njs.admin\_port= ~~4555~~**
  - ▶ \$HOME/unitu\*\*/njs/bin/njs.admin: **\$njs\_port = ~~4555~~**
- ▶ TSI (shepard): 92(1-9)3 and 9303
  - ▶ \$HOME/unitu\*\*/njs/conf/njs.idb: **EXECUTION\_TSI: SOURCE ... ~~4666~~**
  - ▶ \$HOME/unitu\*\*/njs/tsi/conf/tsi.properties: = **tsi.njs\_port= ~~4666~~**
- ▶ TSI (worker): 92(1-9)4 and 9304
  - ▶ \$HOME/unitu\*\*/njs/conf/njs.idb: **EXECUTION\_TSI: SOURCE ... ~~4433~~**
  - ▶ \$HOME/unitu\*\*/njs/tsi/conf/tsi.properties: = **tsi.my\_port= ~~4433~~**



# Quickstart Bundle

- ▶ Installation of UNICORE servers and Client is finalized
- ▶ All server components are on one machine (zam461)



# Notes

- ▶ Gateway normally runs on a separate machine and services more than one virtual site
- ▶ The TSI is usually interfacing to a batch system such as PBS
  - ▶ In our demo installation the „NOBATCH“ TSI is used that simply executes everything directly
- ▶ Also the NJS and the TSI don't need to run on the same machine



# Notes (cont.)

- ▶ In a multiuser environment the TSI runs as root
  - ▶ All commands are executed after the TSI does a `setuid` to the „unixname“ from the UUDB
- ▶ All certificates in the D-Grid initiative have to be signed by DFN



# Server Prerequisites

- ▶ Gateway and NJS:
  - ▶ SUN JRE > 1.4.x
  - ▶ X509 certificates for Gateway and NJS
    - E.g. <https://projects-ca.fz-juelich.de>
  - ▶ Signer certificates
    - E.g. [projects-ca-fz-juelich.pem](#)
- ▶ TSI
  - ▶ Perl >= 5.004 running on the target system machine



# Requesting a server certificate

- ▶ Generate a certificate request:
  - ▶ `openssl req -new -nodes -keyout user1key.pem -out user1.csr`
  - ▶ Or in the UNICORE client: Settings -> Keystore Editor: Actions -> Generate Certification Request
- ▶ Submit the request to your local CA.
- ▶ Then, merge the signed certificate and the private key into a .p12 file, as follows
  - ▶ `openssl pkcs12 -export -in user1.pem -out user1.p12 -inkey user1key.pem -name "user1" -certfile signer-ca.pem (e.g. projects-ca-fz-juelich.pem)`
  - ▶ Or in the UNICORE client: Settings -> Keystore Editor – Actions -> Import Keystore



# Gateway

- ▶ Entry point of a UNICORE Site
- ▶ Accepts SSL connections from Clients and NJSs
- ▶ Accepts valid certificates from all signers known to it (authentication)
- ▶ Talks UNICORE Protocol Layer (UPL) on connections to the outside world
- ▶ Sends/receives AJOs to/from the local NJSs



# Gateway Certificate Checking

- ▶ Following checks are performed on certificates presented by a client
  - ▶ Certificate is issued by one of the trusted CAs (Idris-CA, Cineca-CA, FZJ-CA...)
  - ▶ Certificate is within its validity period
  - ▶ Certificate has not been revoked (if check for Certification Revocation Lists (CRL) is activated)



# Gateway Installation

- ▶ `tar xvzf gateway_4.1.1_build2.tar`
- ▶ It is recommended to keep the directory structure



# Gateway directory structure

gateway

docs/

*Documentation and example code*

conf/

gateway.properties

connections

logs

lib/

ajo.jar

gateway.jar

bin/

*Scripts to control the gateway*



# gateway.properties

- ▶ Machine name advertised by the Gateway
  - ▶ gw.gateway\_host\_name= *zam461.zam.kfa-juelich.de*
- ▶ Port to listen for connections from clients
  - ▶ gw.port= *911\**
- ▶ Initialise the list of supported Vsite
  - ▶ Dynamic registration is possible
  - ▶ Static registration is recommended for D-Grid
    - gw.connections = *connections*
    - gw.named\_njs = *no\_njs\_registration\_allowed*



# gateway.properties (cont.)

- ▶ Where to find the gateway's private key
  - ▶ `gw.identity= $HOME/unitu**/gateway/conf/gateway_identity.p12`
- ▶ The password for the private key
  - ▶ `gw.password= the!gateway`
    - *Or path of a file containing the password*
- ▶ List of all trusted signer certificates
  - ▶ devided by ,:'
  - ▶ `gw_trusted_cas= $HOME/unitu**/gateway/conf/fakeca-cert.pem`



# gateway.properties (cont.)

- ▶ Number of allowed Gateway threads
  - ▶ gw.max\_threads= 25 (optional)
- ▶ Gateway logging level
  - ▶ gw.logging\_level= *T(S, W, I, C, D)*
- ▶ Change interval of the gateway log file
  - ▶ gw.change\_log\_files= *d(aily), [h(our), 4, 5, 6...]*



# Gateway Logging

- ▶ 6 levels of logging
  - ▶ **Severe**: serious problem: Execution cannot continued
  - ▶ **Warning**: problem but processing can continue usually correctly
  - ▶ **Information**: useful information, not necessarily related to any problems
  - ▶ **Configuration**: messages relating to the Gateway configuration
  - ▶ **Talk**: verbose output
  - ▶ **Debug**: very verbose output



# connections

## ► List of supported VSITES

```
# Arcon Connections definition file version 3.6 (do not change this line)
# This file defines the Vsites under the control of this Gateway.
# Simple definition format is:
# <Vsite name> <NJS machine> <NJS port>
# example:
# fecit meiousei.fecit.co.uk 8186
# Sven van den Berghe, fecit
DEMO_NJS zam461.zam.kfa-juelich.de 92**
```



# Gateway Scripts

- ▶ Gateway distribution includes scripts to control the execution of the gateway (in gateway's bin/)
  - ▶ `start_gateway` [`<conf_dir>`] *starts the Gateway*
  - ▶ `start_gateway_p` [`<conf_dir>`] *starts the Gateway and prompts for the gateways password*
  - ▶ `stop_gateway` [`<conf_dir>`] *checks for LAST\_PID and stops that process*
  - ▶ `list_log_files` `type` [`<conf_dir>`] *list log file names and pathes*
  - ▶ `change_log_level` {S|W|I|C|T|D} [`<conf_dir>`]
  - ▶ `change_log_file` [`<conf_dir>`] *starts new log file*



# Start Gateway

- ▶ `cd gateway_dir/bin`
- ▶ Check the Java Path in the `start_gateway` script (`JAVA=`)
- ▶ `start_gateway ../conf`
- ▶ `cd ../conf/logs`
- ▶ Look for log entry is „Initialisation complete“ -  
> Installation successful



# Network Job Supervisor (NJS)

- ▶ UNICORE scheduler
- ▶ Receives/sends AJOs from/to local Gateway
- ▶ Translates AJO into batch job for target
- ▶ Maps the user's Ulogin to Xlogin
- ▶ Sends sub-AJOs to corresponding Gateway according to dependencies
- ▶ Polls for status and output of sub-AJOs
- ▶ Sends batch jobs and requests to TSI
- ▶ Polls TSI for job status and output



# NJS Installation

- ▶ NJS uses a number of files and directories during start up and processing:
  - ▶ A configuration file (*njs.properties*)
  - ▶ A file to define resource and incarnation information (the IDB)
  - ▶ UUDB (mappings from public certificates to Xlogins)
  - ▶ Log directory
  - ▶ Directory in which to save restart information



# NJS Installation (cont)

- ▶ `gunzip njs_4.6.2_build_2.tar.gz`
- ▶ `tar xvf njs_4.6.2_build_2.tar`
- ▶ It is recommended to keep the directory structure



# NJS Directory Structure

njs\_4.6.2\_build\_2/

docs/

*NJS and TSI documentation*

conf/

njs.properties

IDB file

logs/

lib/

ajo.jar

njs.jar

bin/

*Scripts to control the NJS*



# njs.properties

- ▶ Operate as a full NJS
  - ▶ `njs.my_adress= zam461.zam.kfa-juelich.de`
  - ▶ **`njs.operation_mode = full`**
    - *all Abstract Actions are processed*
- ▶ The name of the Vsite that this NJS controls
  - ▶ `njs.vsite_name= DEMO_NJS`
- ▶ Where the NJS will save state if stopped
  - ▶ `njs.save_dir= $HOME/unitu**/SAVE_STATE`
- ▶ The location of the UADB directory (UNICORE User Database)
  - ▶ `uadb.directory= $HOME/unitu**/uadb`



# njs.properties

- ▶ Where to find the Incarnation database
  - ▶ `njs.incarnationdb= njs.idb`
- ▶ Level of NJS log Files
  - ▶ `njs.logging_level= T (S, W, I, C, D)`
- ▶ NJS logging interval
  - ▶ `njs.log_file_change_interval= Daily (Hourly, 4,5,6...)`
- ▶ On this port the NJS listens for administration commands
  - ▶ `njs_admin_port= 92(1-9)2 and 9302`
    - *Port number is arbitrary*



# njs.properties

- ▶ Port on which the NJS listens for connections from Gateways
  - ▶ `njs.gateway_port= 92(1-9)1 and 9301`
- ▶ The Gateway machine name
  - ▶ `njs.gateway= zam461.zam.kfa-juelich.de`
- ▶ Registration with the gateway above is not wanted
  - ▶ `njs.gw_registration= false`



# njs.properties

- ▶ Whether or not use https to connect to gateways
  - ▶ `njs.use_ssl=true`
    - *protocol for connections to other Vsites*
- ▶ Password to decrypt the NJS private key
  - ▶ `njs.ssl_password= the!njs`
    - Or file in which the password is stored
- ▶ Where to find the NJS's private key
  - ▶ `njs.njs_cert_loc= ./njs_identity.p12`
- ▶ Signing authority certificates
  - ▶ `njs.unicore_ca_loc= ./fakeca-cert.pem`
  - ▶ *list of accepted signing authority certificates separated by :*



# NJS Scripts

- ▶ NJS distribution includes scripts to control the execution of the NJS and TSI (in bin/)
  - ▶ `list_log_files` type <configuration\_directory>
    - <type>:

a	all files
l	all since latest start
L	all before latest start
n	latest n files
-n	all except latest n
  - ▶ `start_njs` <configuration\_directory>
    - Starts NJS
  - ▶ `njs_admin`
    - TSI/NJS administration interface



# NJS Startup / Shutdown

- ▶ Startup: `start_njs <conf_dir>`
  - ▶ starts NJS using the configuration given; creates `LAST_PID` in `<conf_dir>`; starts new log file in `<conf_dir>/logs`
  - ▶ Set JAVA path
  - ▶ `MEMORY=Xmx256m`
  - ▶ `cd conf/logs`
  - ▶ Look for log entry „Initialisation complete“ -> installation successful
- ▶ Shutdown: `njs_admin stop (now)`



# NJS Administration Interface (njs\_admin)

- ▶ Perl script implementing administrator interface
  - ▶ Set Perl path
- ▶ Connects to NJS on specified port
- ▶ Connection specification in script:
  - `$njs_machine = "<host name>";`
  - `$njs_port = "<port>"; [ 92(1-9)2 and 9302 ]`
  - Can be overwritten by command line options `-m`, `-p`*
- ▶ Session and command mode



# njs\_admin commands

- ▶ help
  - ▶ Print help about the commands recognised by NJS
- ▶ status
  - ▶ returns the NJS status
- ▶ pause\_njs [number]
  - ▶ Pause the execution of NJS
- ▶ quit
  - ▶ Stops execution of the njs\_admin command



# njs\_admin commands (cont.)

- ▶ list [short|detailed|long] [<selection>]

- ▶ <selection>

- ajos     *all AJOs corresponding to batch jobs*

- all       *all AJOs*

- <expression> *terms using & and | and ( ) with  
type, status, user, ulogin, bssid,  
and rootajo selections*



# njs\_admin commands (cont.)

- ▶ abort <selection>
  - ▶ Aborts execution of the selected actions on NJS
- ▶ cancel <selection>
  - ▶ Cancel the execution of the selected actions on NJS
- ▶ hold <selection>
  - ▶ Hold execution of the selected actions on the NJS
- ▶ resume <selection>
  - ▶ Resumes the held selection



# njs\_admin commands (cont.)

- ▶ `tsi [up|down|status|stop|refresh]`
  - ▶ Controls the TSI and the NJS view of TSI status



# njs\_admin commands

logging [new\_file | level | interval | info]

- ▶ Close current log file and open new one
- ▶ Set new logging level:
  - ▶ logging level [S | W | I | C | T | D]
- ▶ Set new interval for changing the log file
  - ▶ logging interval [daily | hourly | *n*]
- ▶ Gives current logging status (level, interval, log-file)



# njs\_admin commands

ls [outcomes | uspaces] [<Xlogin>]

remove [outcome | uspace] <ajo\_id><Xlogin>

njs\_admin 'ls uspaces zdv038'

```
drwx----- 2 rambadt root    4096 Oct 18 14:47
uspace_7d86d3a1 New_Job1 (7d86d3a1 in 7d86d3a1)
AJO    14:47:51 18/10 EXECUTING  rambadt ()
Ulogin: ...
```



# UNICORE User Database (UUDB)

- ▶ Management of Ulogin – Xlogin mapping information
- ▶ NJS accesses this information
- ▶ Basic version allows to map one certificate to exactly one Xlogin
- ▶ NJS to UUDB interface defined to adapt to site specific user data bases (i.e. ldap)



# UUDB Installation

- ▶ `gunzip uudb_1.0.0.tar.gz`
- ▶ `tar xvf uudb_1.0.0.tar`
- ▶ `cd uudb_1.0.0/src`
- ▶ `chmod 700 installer`
- ▶ `./installer <UUDB root path> <NJS root path>`
  - ▶ UUDB root path: `$HOME/unitu**/uudb_1.0.0`
  - ▶ NJS root path: `$HOME/unitu**/njs/`



# UUDB Administration

- ▶ **add** <certificate file><xlogin>
  - ▶ Add user certificate (x509 public key) to UUDB mapped to xlogin
  - ▶ Example: add \$HOME/viola/unicore/certs/rambadt.pem zdv017
- ▶ **add\_njs** <certificate file><njs\_name>
  - ▶ Add NJS certificate to UUDB (necessary if NJS certificates don't have the UNICORE server extension)
- ▶ **list**
  - ▶ List current entries in UUDB
- ▶ **delete** <xlogin>
  - ▶ Delete xlogin entry from UUDB
- ▶ **delete\_njs** <njs\_name>
  - ▶ Delete NJS from UUDB



# Incarnation Database

- ▶ Information service and translation table
- ▶ Contains definitions for each site and each target system
- ▶ NJS converts abstract job description into site and vendor specific commands and sends this to the TSI
- ▶ The IDB contains information for the NJS about
  - ▶ Uspace folder
  - ▶ Stdout / stderr folder
  - ▶ How to retrieve status information



# Incarnation Database (cont.)

- ▶ Devided into sections:
  - ▶ GENERAL Section (file spaces, descriptions, ...)
  - ▶ EXECUTION\_TSI section (host + ports, resources, batch queues, ...)
  - ▶ RUN Section (*Translation of application into executable commands*)
  - ▶ FILE\_COPY CLEANUP LIST\_DIRECTORY RENAME\_FILE SYMBOLIC\_LINK DELETE\_FILE CHANGE\_PERMISSION
  - ▶ FORTRAN section
  - ▶ LINK section
  - ▶ COPY\_PF section
  - ▶ MAKE\_FIFO section



# Incarnation Database (cont.)

## ▶ GENERAL section

### ▶ DEFINE NJS\_FILE\_SPACE (directory on the production machine) \$HOME/deisa/unicore/ospace

- USPACE\_ROOT                      NJS\_FILE\_SPACE
  - *path for job's temporary working dir*
- OUTCOME\_ROOT                      NJS\_FILE\_SPACE
  - *Outcome directory*
- TextInfoResource
  - *site defined information to be given to the user*



# Incarnation Database (cont.)

- ▶ EXECUTION\_TSI section
  - ▶ NAME DEISA
  - ▶ SOURCE **your machines IP adress 9755 9765**
  - ▶ NODE / PROCESSOR / MEMORY /CPUTIME
    - Must be set to the production system values and must agree with the values in the QUEUE section few lines below)
  - ▶ DEFINE BIN\_DIR **/bin** (review it!)
  - ▶ DEFINE USR\_DIR **/usr/bin** (review it!)
- ▶ RUN section
  - ▶ DEFINE TSI\_LS (directory on the production machine)  
**\$HOME/deisa/unicore/tsi/build/tsi\_.../tsi\_ls** (review tsi path!)
- ▶ **#END OF MUST REVIEW SECTION**



# Target System Interface (TSI)

- ▶ Requires production system specific modifications!!!
- ▶ Runs on the production machine
- ▶ Interface to target operating and batch system
- ▶ Perl scripts and modules (needs Perl 5.004)
- ▶ Needs to be started with root privileges to act on behalf of the user (uses setuid)
- ▶ Provides interface to local system for
  - ▶ Job submission
  - ▶ Status query, job monitoring
  - ▶ File handling
  - ▶ ...



# TSI Installation

- ▶ `gunzip tsi_4.1.0_build_2.tar.gz`
- ▶ `tar xvf tsi_4.1.0_build_2.tar`
- ▶ `cd tsi_4.1.0_build_2`
- ▶ `chmod 700 Install.sh`
- ▶ `./Install.sh`
  - ▶ Select Number of your target system architecture
  - ▶ Enter TSI path
  - ▶ confirm installation values



# TSI configuration

- ▶ `cd $HOME/unitu**/conf`
- ▶ Edit `tsi.properties`
  - ▶ `tsi.path=$HOME/unitu**/tsi/tsi_NOBATCH`
    - The path to the main ,tsi' file
  - ▶ `tsi.njs_machine=zam461.zam.kfa-juelich.de`
    - The name/adress of the NJS machine
  - ▶ `tsi.njs_port = 92(1-9)3 and 9303`
    - The port on which the NJS is listening for TSI worker connections
  - ▶ `tsi.my_port = 92(1-9)4 and 9304`
    - The port on which the TSI shepherd process listens for NJS requests



# TSI for Batch systems

- ▶ Choose your batch system when installing the TSI
- ▶ Modify `GetStatusListing.pm` and `Submit.pm` to your BSS specifications



# TSI startup

- ▶ cd tsi bin directory
- ▶ Edit ./start\_tsi: set PERLPATH
- ▶ ./start\_tsi



# Ports used in this installation

- ▶ Client – GW : 911(0-9)
  - ▶ \$HOME/unitu\*\*/gateway/conf/gateway.properties: **gw.port= ~~4004~~**
  - ▶ \$HOME/unitu\*\*/gateway/conf/gateways.xml: ~~4004~~
- ▶ GW – NJS: 92(1-9)1 and 9301
  - ▶ \$HOME/unitu\*\*/gateway/conf/conections: ~~4444~~
  - ▶ \$HOME/unitu\*\*/njs/conf/njs.properties: **njs.gateway.port= ~~4444~~**
- ▶ NJS Admin: 92(1-9)2 and 9302
  - ▶ \$HOME/unitu\*\*/njs/conf/njs.properties: **njs.admin\_port= ~~4555~~**
  - ▶ \$HOME/unitu\*\*/njs/bin/njs.admin: **\$njs\_port = ~~4555~~**
- ▶ TSI (shepard): 92(1-9)3 and 9303
  - ▶ \$HOME/unitu\*\*/njs/conf/njs.idb: **EXECUTION\_TSI: SOURCE ... ~~4666~~**
  - ▶ \$HOME/unitu\*\*/njs/tsi/conf/tsi.properties: = **tsi.njs\_port= ~~4666~~**
- ▶ TSI (worker): 92(1-9)4 and 9304
  - ▶ \$HOME/unitu\*\*/njs/conf/njs.idb: **EXECUTION\_TSI: SOURCE ... ~~4433~~**
  - ▶ \$HOME/unitu\*\*/njs/tsi/conf/tsi.properties: = **tsi.my\_port= ~~4433~~**



# Part 2: Client

- ▶ Content
  - ▶ unicoreSites.xml
  - ▶ Download
  - ▶ Installation



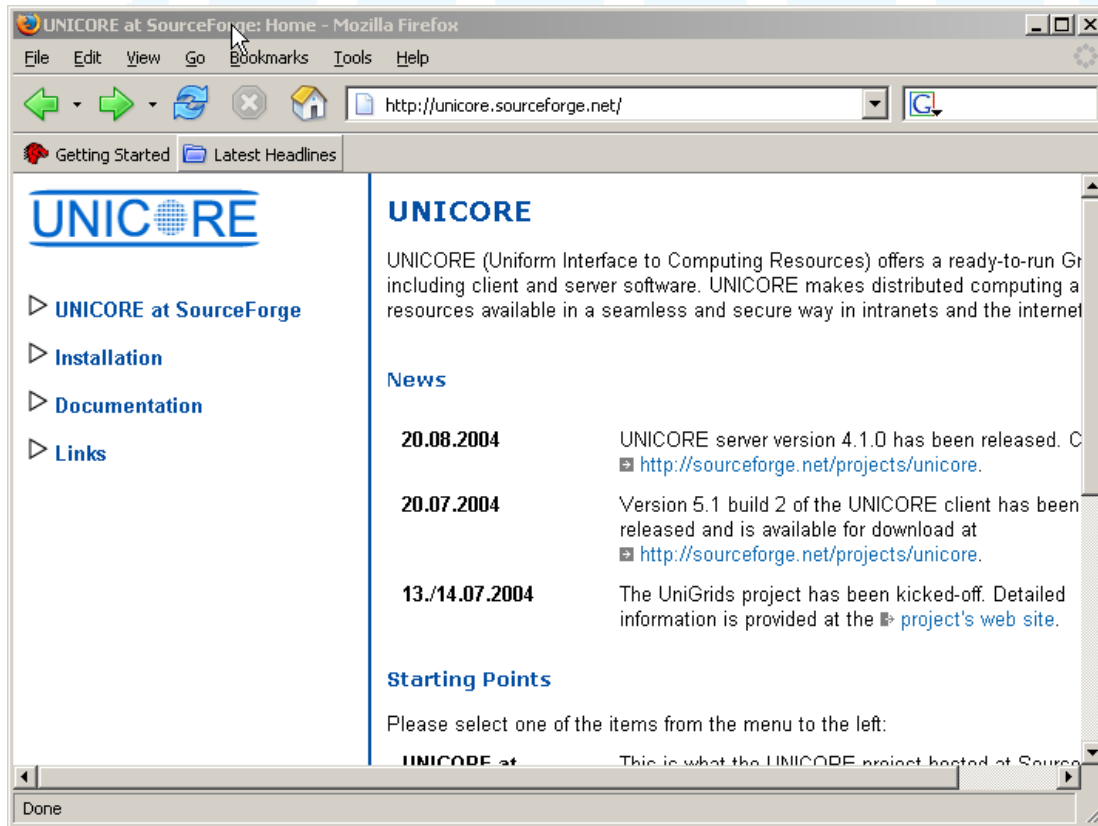
# unicoreSites.xml

- ▶ File which contains gateway information for the client

```
<!-- FZJ DGrid test server-->
<UsiteList>
  <Usite name=„D-Grid Test“ description=„zam461“
    address="zam461.zam.kfa-juelich.de" port="9110">
  </Usite>
  ...
</UsiteList>
```

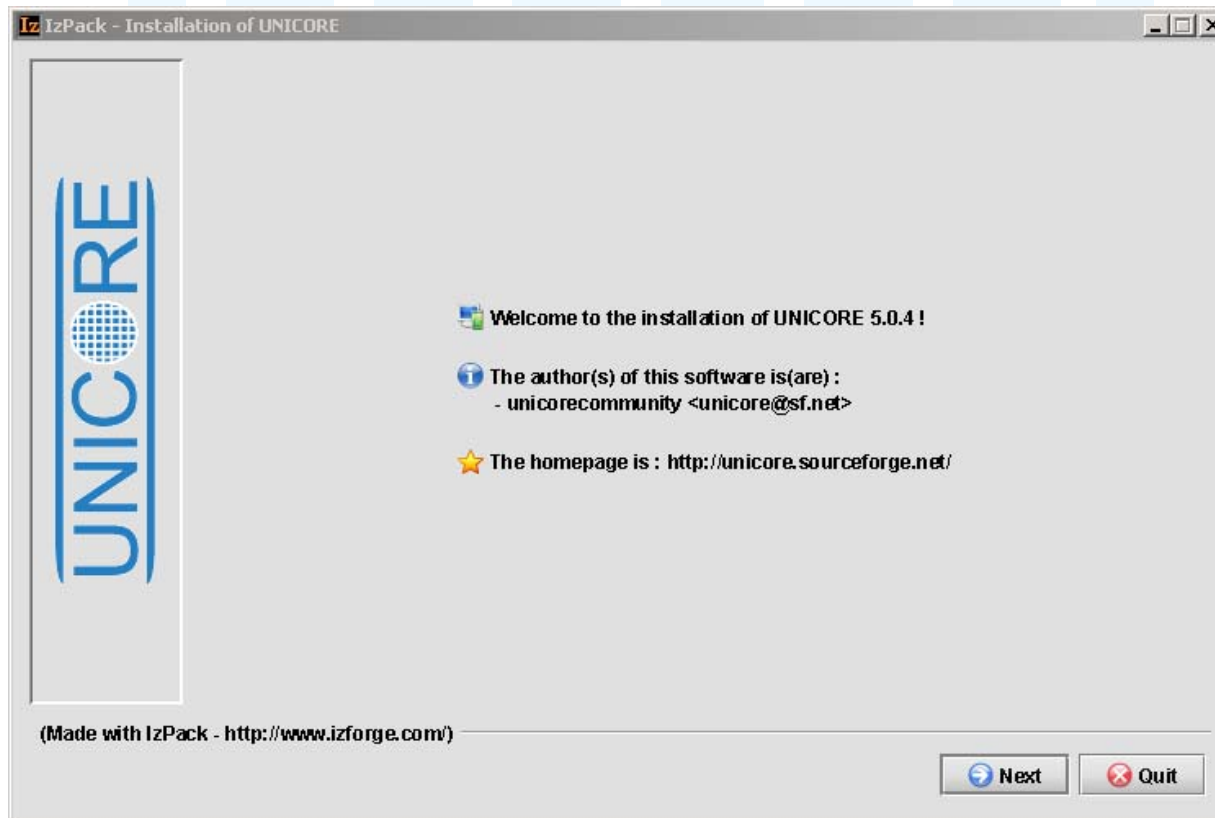


# UNICORE at sourceforge.net



# Installation

- ▶ Easy installation with a graphical installer



# Submitting the request to the CA

- ▶ Select your CA and submit



# Getting a user certificate

- ▶ Generating a certificate request

**Generate certification request**

*individual*

Common Name: \* Momo Mustermann

Email: \* m.mustermann@fz-juelich.de

*is associated with*

Organization: \* Forschungszentrum Juelich

Organizational Unit:

*is located*

Locality:

State:

country(shortcut) \* DE

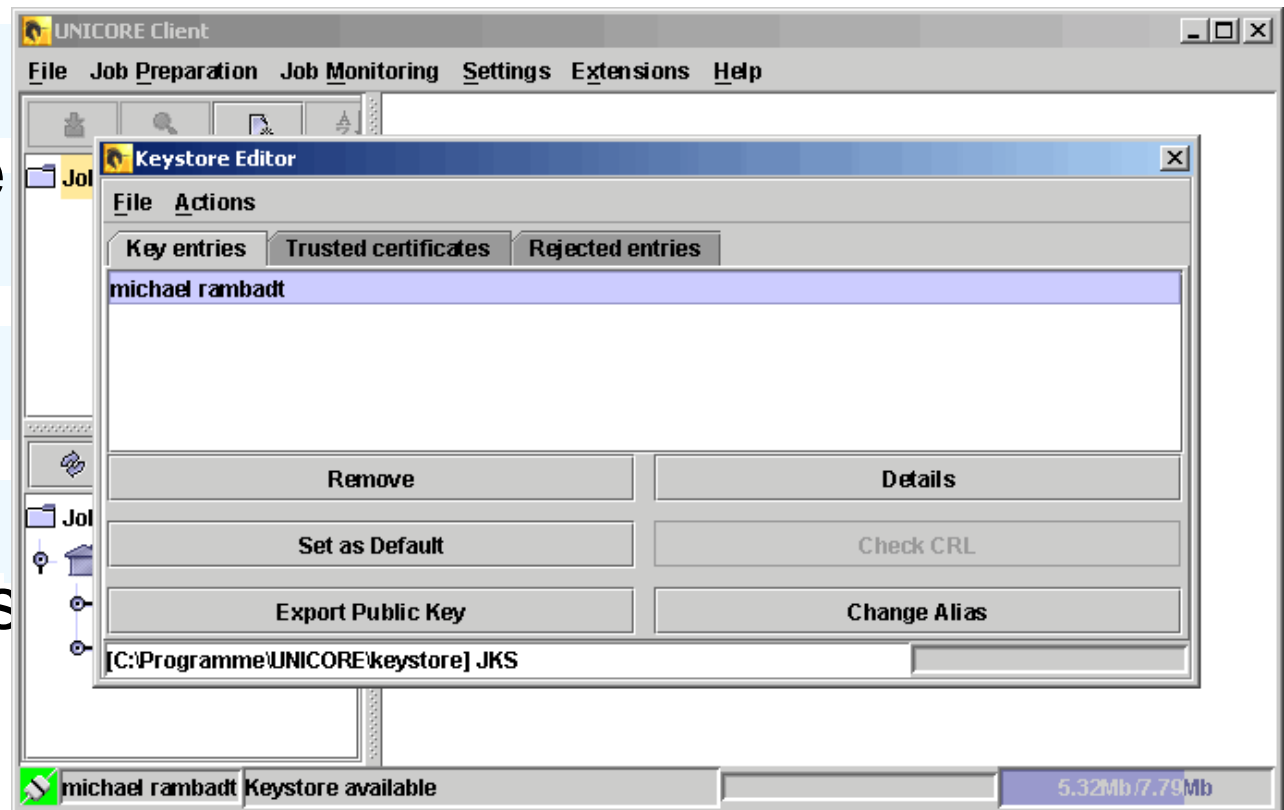
OK Cancel



# Importing the certificate into UNICORE

- ▶ After submitting the request to the local CA and getting back the signed certificate:

- ▶ Import use certificate
- ▶ Import gateway certificate(s)



# Import unicoloreSites.xml

- ▶ Open UNICORE Client
- ▶ Select Settings -> User Defaults ->  
URL(s) for UNICORE Site Server
- ▶ Choose unicoloreSites.xml from your local disk

