Identificazione documento

Titolo	Tipo	Nome file
Guida GRID di primo livello	Documentazione	SIS_guida_grid_unina_v1

Approvazioni

	Nome	Data	Firma
Redatto da	Boccia, Monorchio	30/04/2009	
Revisionato da			
Approvato da	Barone	04/05/2009	

Variazioni

Versione	Data	Autore	Paragrafi modificati
1	30/04/2009	Boccia, Monorchio	

Indice dei contenuti

1 Note preliminari per usufruire delle risorse di UNINA DATACENTER

Per poter **accedere** alle risorse del DATACENTER è **indispensabile** essere in **possesso** di un **certificato personale**.

1.1 Richiesta certificato personale

La procedura per ottenere il certificato è la seguente:

- L'utente si reca di **persona** alla Registration Authority (RA) competente (Sig. Ciotola – CSI, sede di Monte S.Angelo edificio Centri Comuni) per farsi autenticare portando con se:
    a. Un documento di riconoscimento valido (carta di identità o patente di guida) e codice fiscale
    b. L'autorizzazione del proprio referente di attività presso questa Università
- La RA ne accerta l'identità;
- La RA invia all'INFN Certification Authority (CA) i dati del richiedente: nome, cognome, indirizzo e-mail, estremi del documento di identità ed il codice fiscale;
- All'utente viene fornito una "ricevuta di richiesta certificato" contenente, tra l'altro, un **codice pseudocasuale**;
- **Entro, e non oltre 48 ore dalla comunicazione del codice da parte della RA**, l'utente, utilizzando il browser di una **macchina con IP pubblico e registrata sul DNS**, si collega al sito della CA INFN http://security.fi.infn.it/CA/ e scarica il certificato dell'INFN CA;
- Accedendo poi alla sezione "Richiesta certificati personali" disponibile al link https://security.fi.infn.it/CA/mgt/restricted/ucert.php l'utente compila il form di richiesta fornendo gli stessi dati digitati in precedenza dalla RA e sottomette la richiesta;
- Se tutto va bene**, entro 2 giorni lavorativi**, l'utente riceverà le istruzioni per scaricare il certificato personale **usando lo stesso browser con cui ha fatto la richiesta**;
- Una volta scaricato, il certificato si trova in una directory di sistema del browser che si è utilizzato. Nel paragrafo 2, sezioni 2.1, 2.2 e 2.3 sarà illustrata la procedura che consente di utilizzare il certificato personale per l'accesso alle risorse della griglia.

1.2 Iscrizione alla Virtual Organization unina.it

Dopo aver ottenuto il certificato personale o nel caso già si possegga un certificato rilasciato dalla CA INFN, occorre iscriversi alla Virtual Organization **unina.it**:

- L'utente installa il proprio certificato personale nel browser
- Compila il form presente su https://voms01.scope.unina.it:8443/voms/unina.it/Login.do
- L'utente riceve una mail dal server VOMS e convalida la sua richiesta di iscrizione visitando una determinata pagina web

- La richiesta di iscrizione alla VO **unina.it** viene inviata ai VO Manager
- Il VO Manager approva la richiesta
- L'utente riceve una mail di benvenuto nella VO **unina.it**

ATTENZIONE: Dopo la ricezione della mail di benvenuto, potrebbero essere necessarie alcune ore affinché l'affiliazione alla VO venga propagata su tutte le risorse della GRID.

2 Primo accesso alle risorse

L'accesso ai servizi della griglia avviene attraverso un unico punto di accesso configurato con il ruolo GRID di User Interface. E` possibile sia avere una propria User Interface, sia utilizzare le 2 User Interface messe a disposizione degli utenti.

In quest'ultimo caso è necessario richiedere un account al CSI, secondo le modalità illustrate sulla home page di progetto.

Per utilizzare le risorse è necessario disporre del proprio certificato personale sull'UI di accesso. Di seguito vengono descritte in maniera specifica le modalità di accesso ai servizi della griglia.

2.1 Collegamento alla User Interface (UI)

In questa sezione verrà descritta la procedura per collegarsi ad una delle due UI configurate per l'accesso alla GRID. Ci riferiremo alla UI ui02.scope.unina.it (l'altra UI risponde al nome di ui01.scope.unina.it)

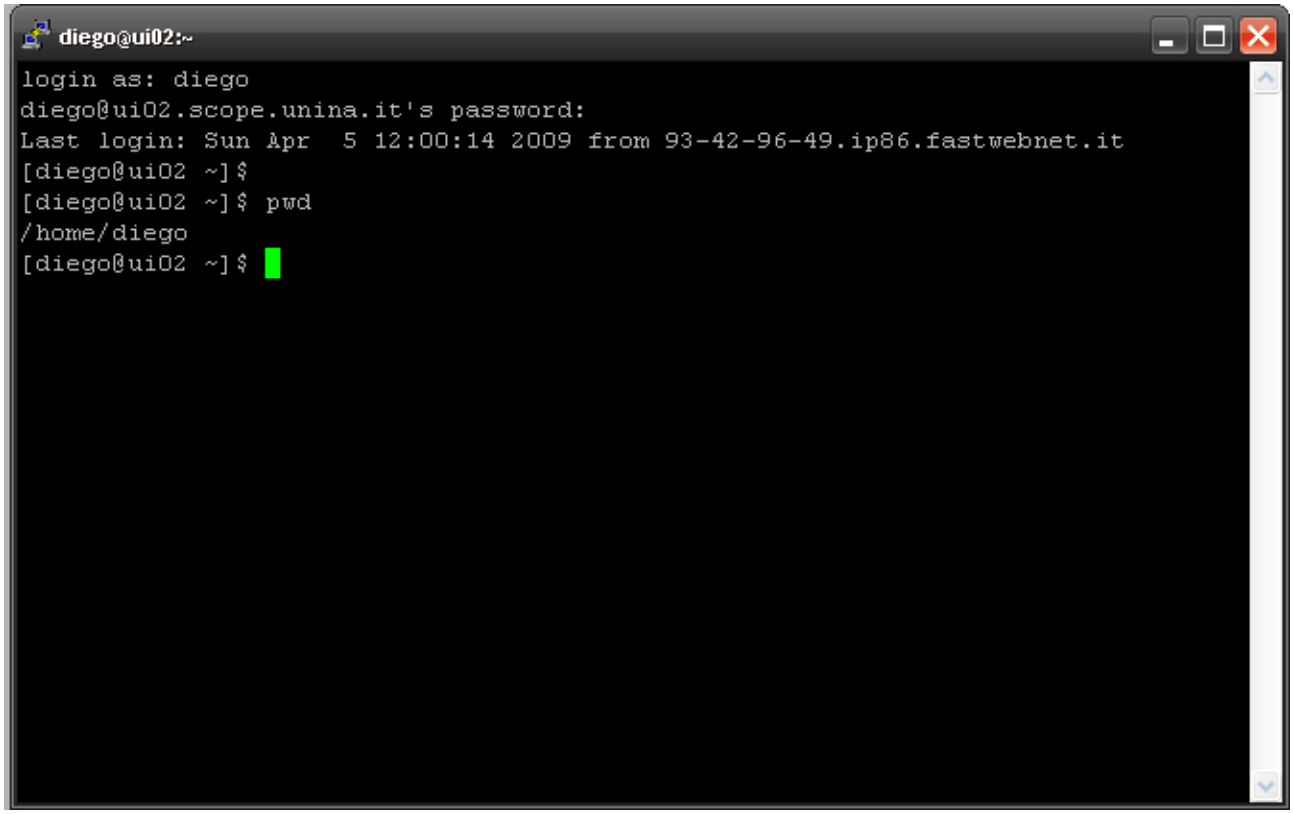
ACCESSO DA UN PC WINDOWS:

Per collegarsi alla UI è necessario utilizzare un emulatore di terminale, ad esempio "Putty", disponibile alla pagina web http://web.na.infn.it/fileadmin/compreso/ssh/ssh-pc.html

Le informazioni necessarie da inserire sono:

- Host Name: ui02.scope.unina.it
- Port: 22
- Connection type: SSH

Il tasto "OPEN" apre il terminale come visualizzato nella successiva figura in cui vengono richiesti i propri user name e password di accesso alla UI.



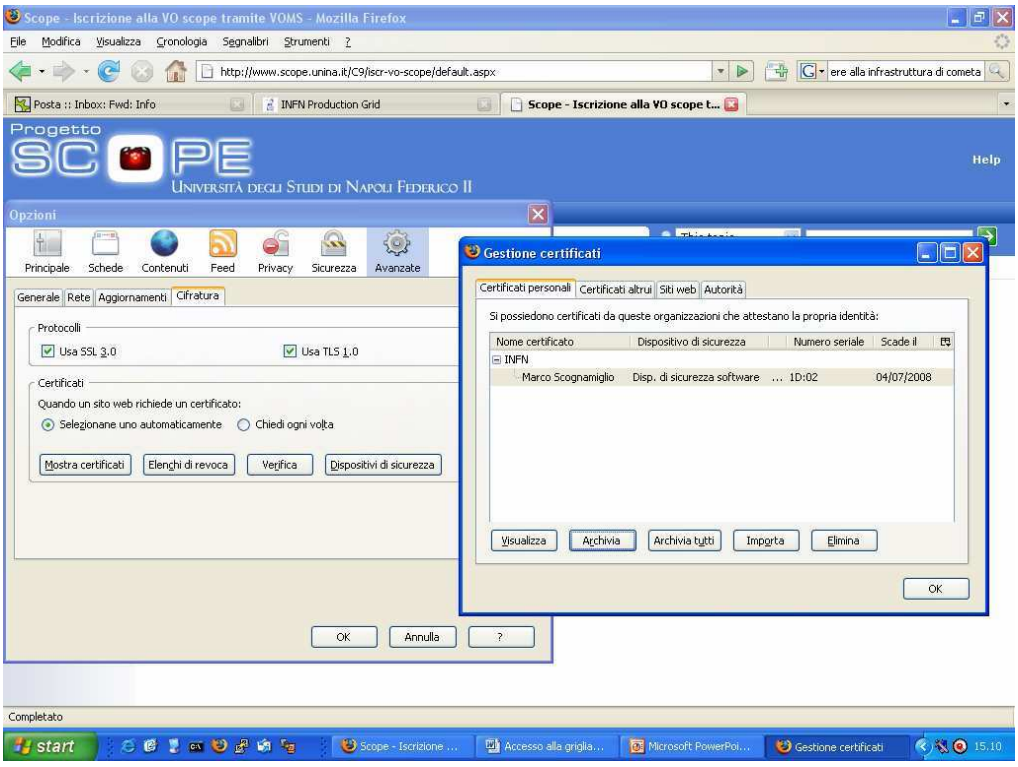
A questo punto è possibile accedere alle proprie directory ed ai propri files sulla UI.

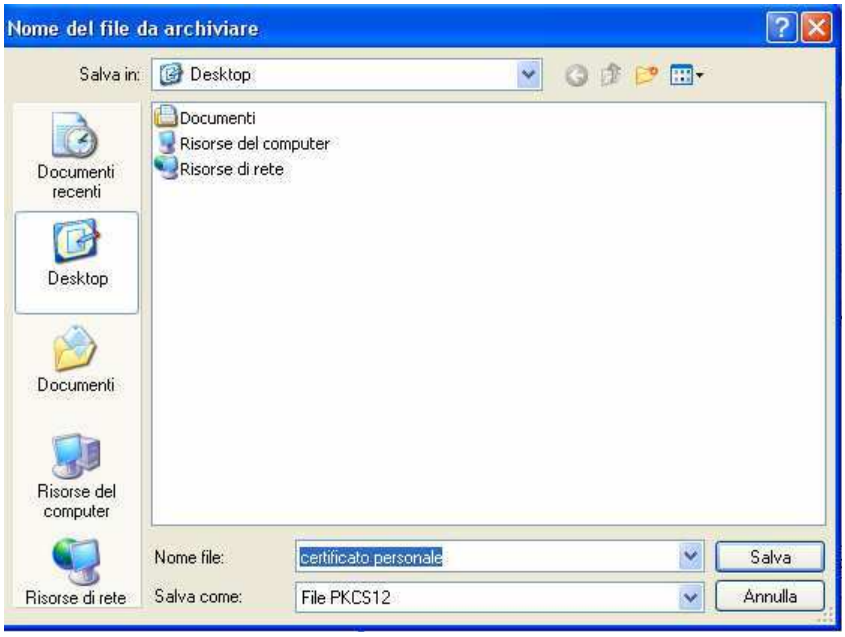
Per essere riconosciuti in GRID e sottomettere i job sarà necessario installare nella propria home directory il certificato personale ottenuto dalla CA.

2.2 Esportazione del certificato personale dal browser (Mozilla e Internet Explorer

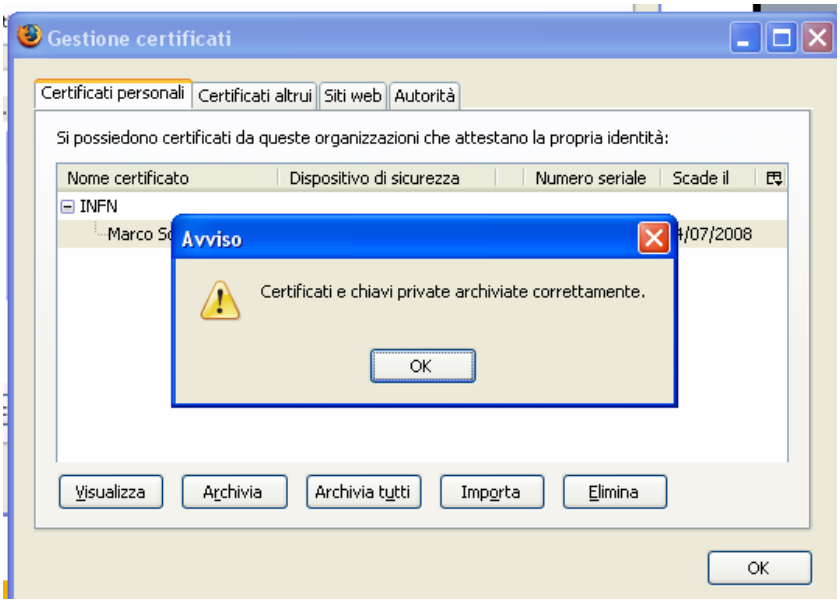
Mozilla Firefox

Nel browser Mozilla/Firefox utilizzare il menu strumenti -> Opzioni -> Avanzate -> Mostra Certificati -> Archivia. Nella scheda personale utilizzare il pulsante archivia e seguire i seguenti passi:




Nome del file da archiviare
Salva in:
Desktop
Documenti
Risorse del computer
Risorse di rete
Documenti recenti
Desktop
Documenti
Risorse del computer
Risorse di rete
Nome file:
certificato personale
Salva
Salva come:
File PKCS12
Annulla

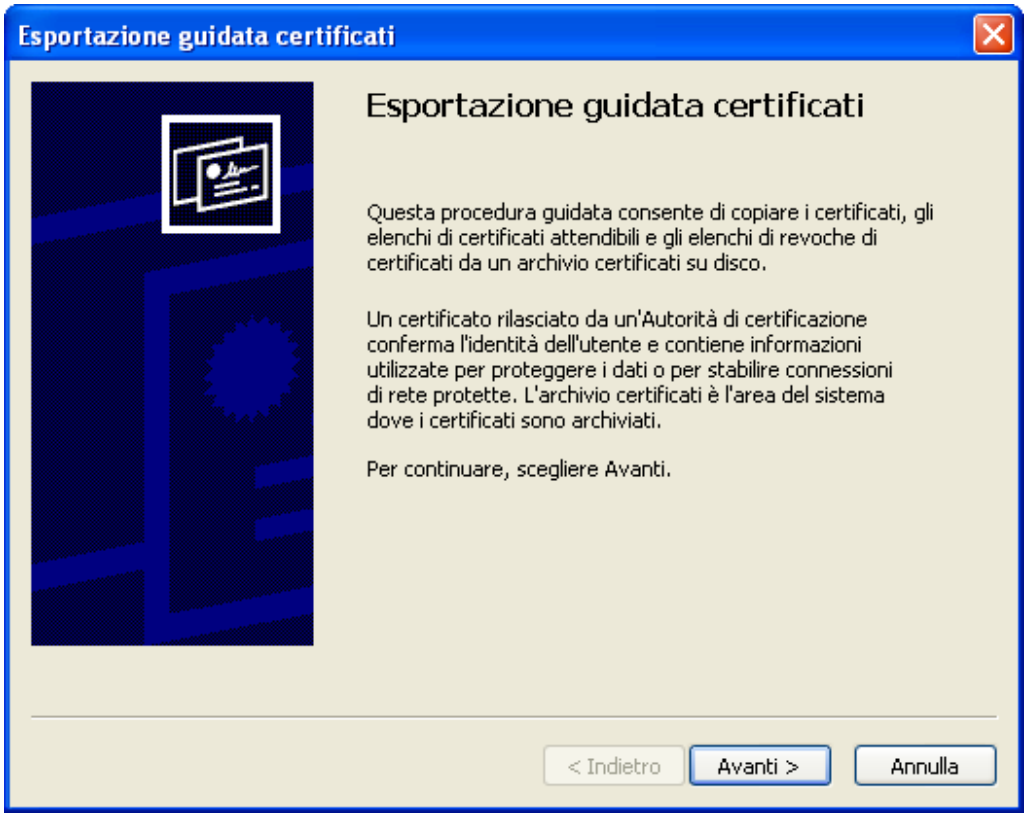

Gestione certificati
Certificati personali
Certificati altrui
Siti web
Autorità
Si possiedono certificati da queste organizzazioni che attestano la propria identità:
Nome certificato
Dispositivo di sicurezza
Numero seriale
Scade il
INFN
Password richiesta
Inserire la password principale per Disp. di sicurezza software.
******
OK
Annulla
Visualizza
Archivia
Archivia tutti
Importa
Elimina
OK

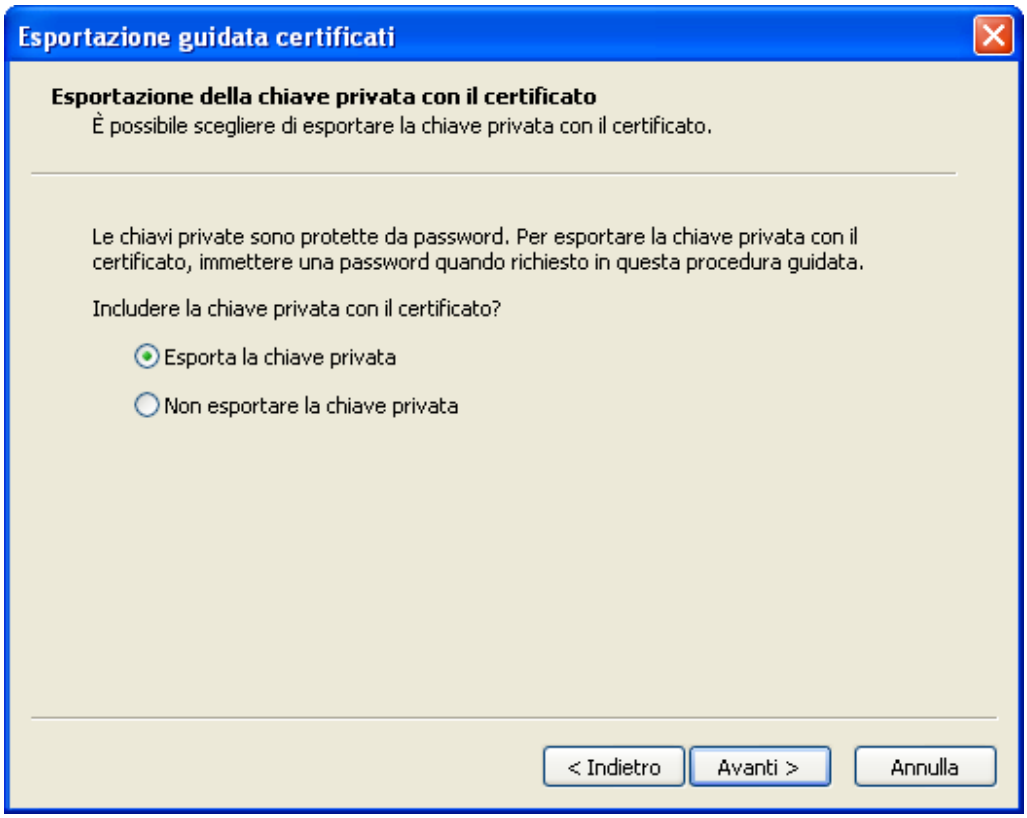


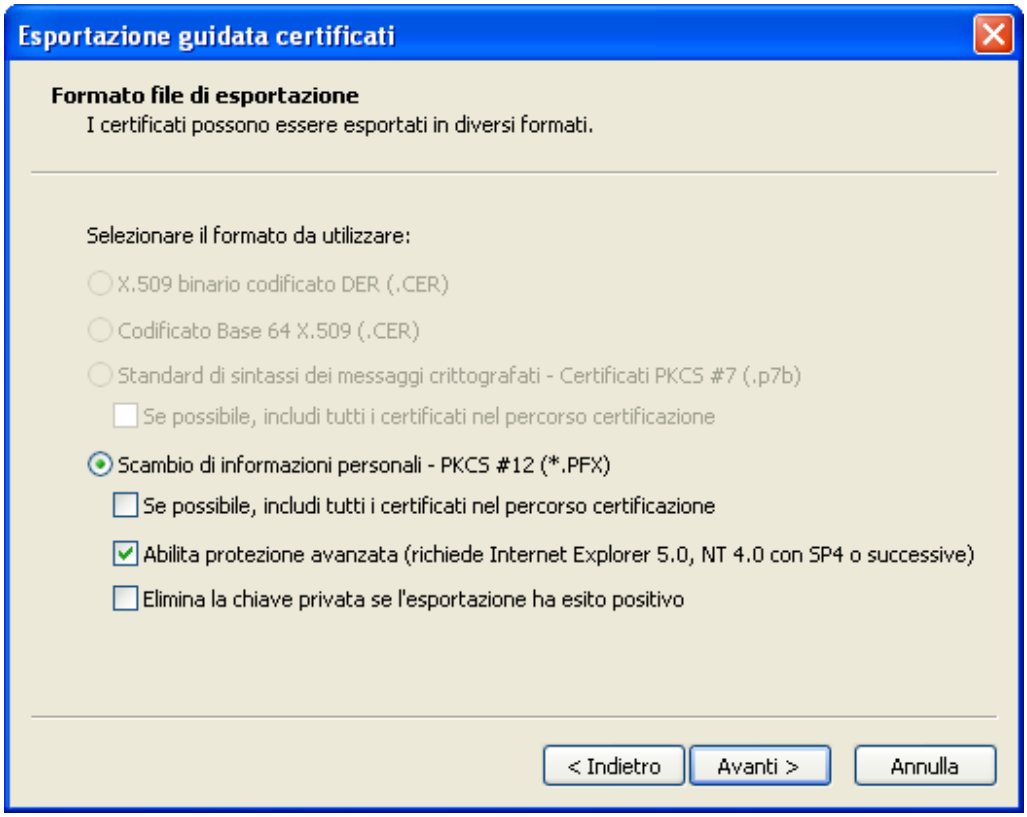
Il certificato è stato così salvato nella directory scelta in formato PKCS12 (estensione p12).

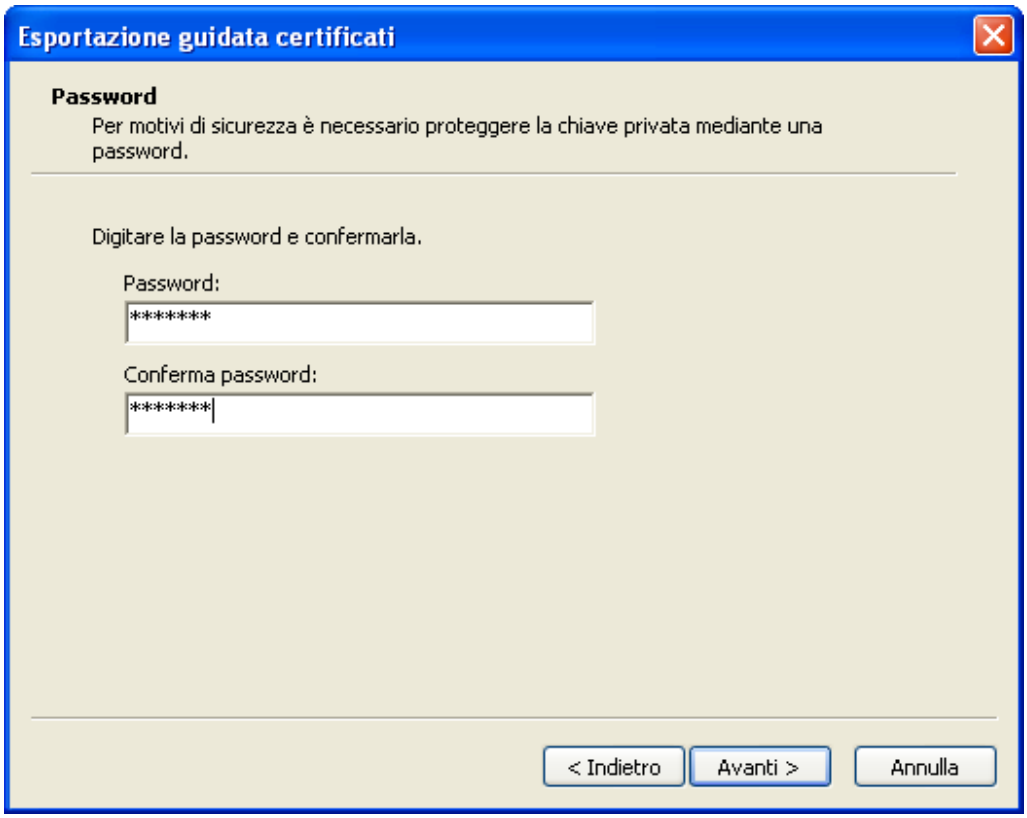
Internet Explorer

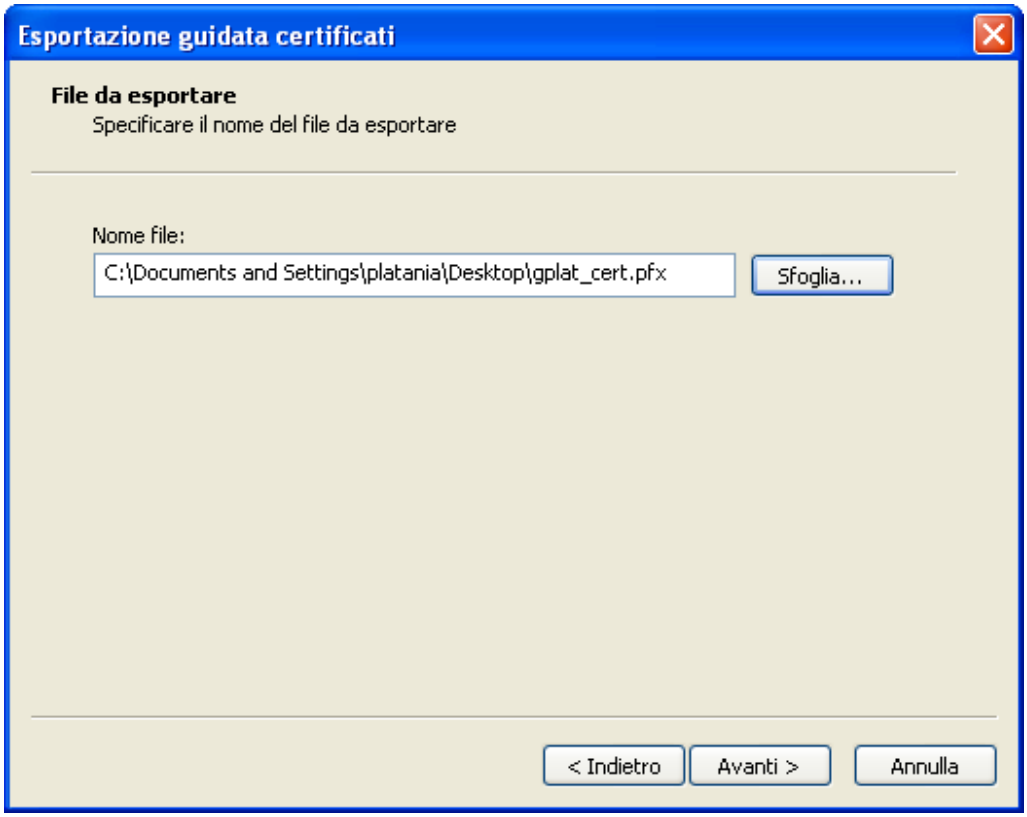
Nel browser Internet Explorer utilizzare il menu strumenti > opzioni internet -> contenuto -> certificati; nella scheda personale utilizzare il pulsante esporta e seguire i seguenti passi:

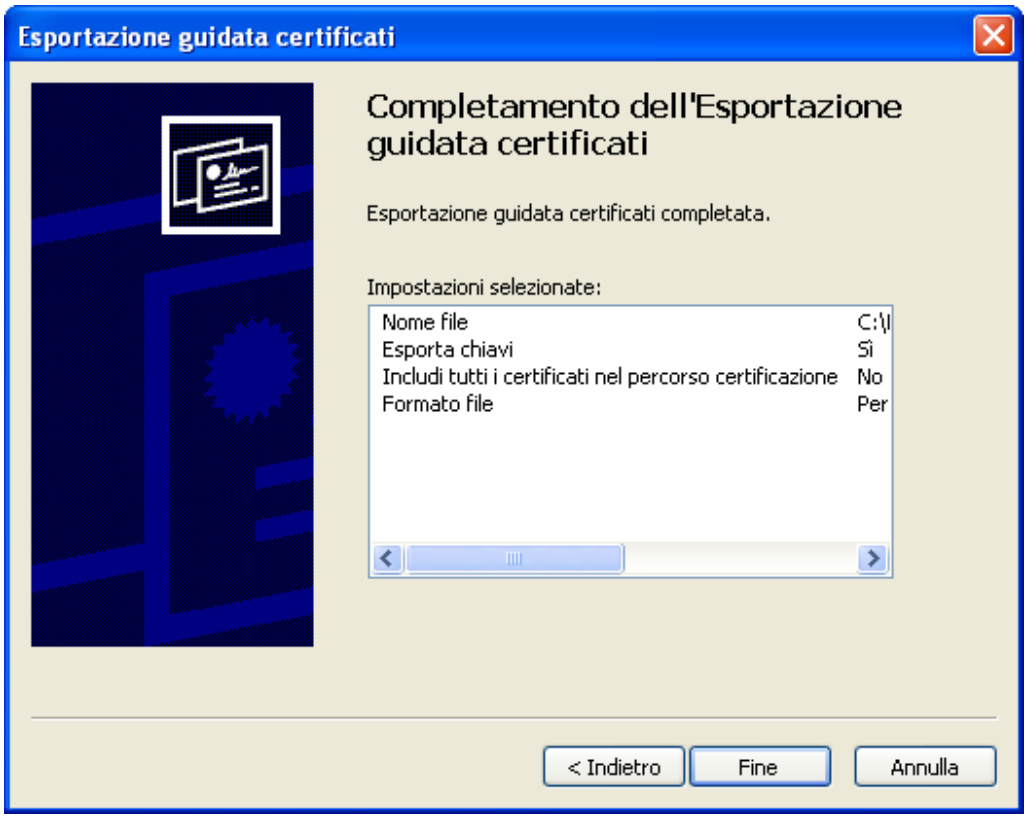

Esportazione guidata certificati
Esportazione guidata certificati
Questa procedura guidata consente di copiare i certificati, gli elenchi di certificati attendibili e gli elenchi di revoche di certificati da un archivio certificati su disco.
Un certificato rilasciato da un'Autorità di certificazione conferma l'identità dell'utente e contiene informazioni utilizzate per proteggere i dati o per stabilire connessioni di rete protette. L'archivio certificati è l'area del sistema dove i certificati sono archiviati.
Per continuare, scegliere Avanti.
< Indietro
Avanti >
Annulla


Esportazione guidata certificati
Esportazione della chiave privata con il certificato
È possibile scegliere di esportare la chiave privata con il certificato.
Le chiavi private sono protette da password. Per esportare la chiave privata con il certificato, immettere una password quando richiesto in questa procedura guidata.
Includere la chiave privata con il certificato?
Esporta la chiave privata
Non esportare la chiave privata
< Indietro
Avanti >
Annulla


Esportazione guidata certificati
Formato file di esportazione
I certificati possono essere esportati in diversi formati.
Selezionare il formato da utilizzare:
X.509 binario codificato DER (.CER)
Codificato Base 64 X.509 (.CER)
Standard di sintassi dei messaggi crittografati - Certificati PKCS #7 (.p7b)
Se possibile, includi tutti i certificati nel percorso certificazione
Scambio di informazioni personali - PKCS #12 (*.PFX)
Se possibile, includi tutti i certificati nel percorso certificazione
Abilita protezione avanzata (richiede Internet Explorer 5.0, NT 4.0 con SP4 o successive)
Elimina la chiave privata se l'esportazione ha esito positivo
< Indietro
Avanti >
Annulla


Esportazione guidata certificati
Password
Per motivi di sicurezza è necessario proteggere la chiave privata mediante una password.
Digitare la password e confermarla.
Password:
****
Conferma password:
****
< Indietro
Avanti >
Annulla


Esportazione guidata certificati
File da esportare
Specificare il nome del file da esportare
Nome file:
C:\Documents and Settings\platania\Desktop\gplat_cert.pfx
Sfoglia...
< Indietro
Avanti >
Annulla


Esportazione guidata certificati
Completamento dell'Esportazione guidata certificati
Esportazione guidata certificati completata.
Impostazioni selezionate:
Nome file C:\
Esporta chiavi Sì
Includi tutti i certificati nel percorso certificazione No
Formato file Per
< Indietro
Fine
Annulla

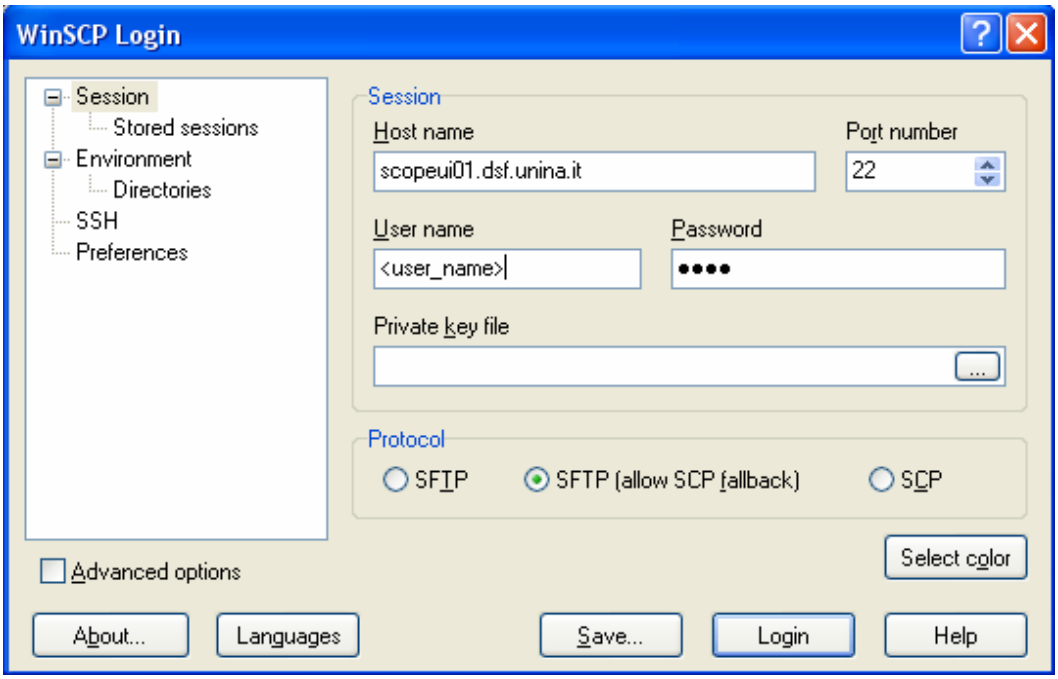
2.3 Copia del certificato personale sulla User Interface (UI)

A questo punto occorre copiare il certificato personale memorizzato in formato .p12 o .pfx nella propria home directory sulla UI.

Se il certificato si trova su una macchina Linux, la copia va effettuata attraverso il comando scp, utilizzando la seguente sintassi:

```
$ scp  certificato_personale.p12  diego@ui02.scope.unina.it:./
```

Se il certificato si trova su una macchina Windows si può utilizzare il software WinSCP, disponibile alla pagina web http://web.na.infn.it/fileadmin/compreso/ssh/ssh-pc.html Tale software consente di collegarsi alla UI e di trasferire il certificato alla home directory.



A partire dal certificato personale, copiato sulla UI, vanno generati due file in formato PEM con il comando openssl. La sintassi da utilizzare è la seguente:

```
$ openssl pkcs12 -nocerts              -in certificato_personale.p12  -out userkey.pem
$ openssl pkcs12 -clcerts  -nokeys -in certificato_personale.p12  -out usercert.pem
```

dove:

certificato_personale.p12 è l'input file in formato PKCS12.

userkey.pem è l'output file in formato PEM della chiave privata.

usercert.pem è l'output file in formato PEM del certificato.

Per quanto riguarda i file che contengono la chiave privata ed il certificato pubblico dell'utente, si consiglia di utilizzare i nomi standard usercert.pem e userkey.pem . Tali file vanno poi spostati nella directory .globus :

```
$ mkdir .globus
$ cp userkey.pem usercert.pem .globus/
```

A questo punto occorre cambiare i permessi a queste chiavi:

```
$ cd .globus/
$ chmod 644 usercert.pem
$ chmod 400 userkey.pem
```

Se si sono utilizzati i nomi standard per i files contenenti la chiave privata ed il certificato pubblico dell'utente, è possibile generare il certificato proxy attraverso il seguente comando:

```
$ voms-proxy-init --voms unina.it
```

Nel caso si utilizzino nomi diversi, utilizzare tale sintassi:

```
$ voms-proxy-init --voms unina.it -key nome_chiave_privata -cert nome_certificato_pubblico -
certdir .nome_directory
```

3 Utilizzo delle risorse

3.1 Fase di autenticazione: generazione proxy delle credenziali

Comandi voms-proxy-init e voms-proxy-info

La sottomissione di un job su Grid necessita della creazione di un certificato proxy utente utilizzando il seguente comando, che fa anche richiesta della password con la quale è stato criptato il certificato:

```
$ voms-proxy-init --voms unina.it

Enter GRID pass phrase:
Your identity: /C=IT/O=INFN/OU=Personal Certificate/L=Federico II/CN=Diego Monorchio
Creating temporary proxy ..................................................... Done
Contacting  voms01.scope.unina.it:15003 [/C=IT/O=INFN/OU=Host/L=Federico
II/CN=voms01.scope.unina.it] "unina.it" Done
Creating proxy ..................................... Done
Your proxy is valid until Tue Apr  7 06:24:43 2009
```

Nel caso in cui si desideri estendere la validità del proxy delle credenziali utilizzare il comando

-hours H in questo modo:

```
$ voms-proxy-init --voms unina.it -hours 48
```

```
Enter GRID pass phrase:
Your identity: /C=IT/O=INFN/OU=Personal Certificate/L=Federico II/CN=Diego Monorchio
Creating temporary proxy ..... Done
Contacting  voms01.scope.unina.it:15003 [/C=IT/O=INFN/OU=Host/L=Federico
II/CN=voms01.scope.unina.it] "unina.it" Done
Creating proxy ..... Done
Your proxy is valid until Wed Apr  8 18:28:40 2009
```

E' possibile visualizzare le informazioni del proprio certificato proxy con il comando:

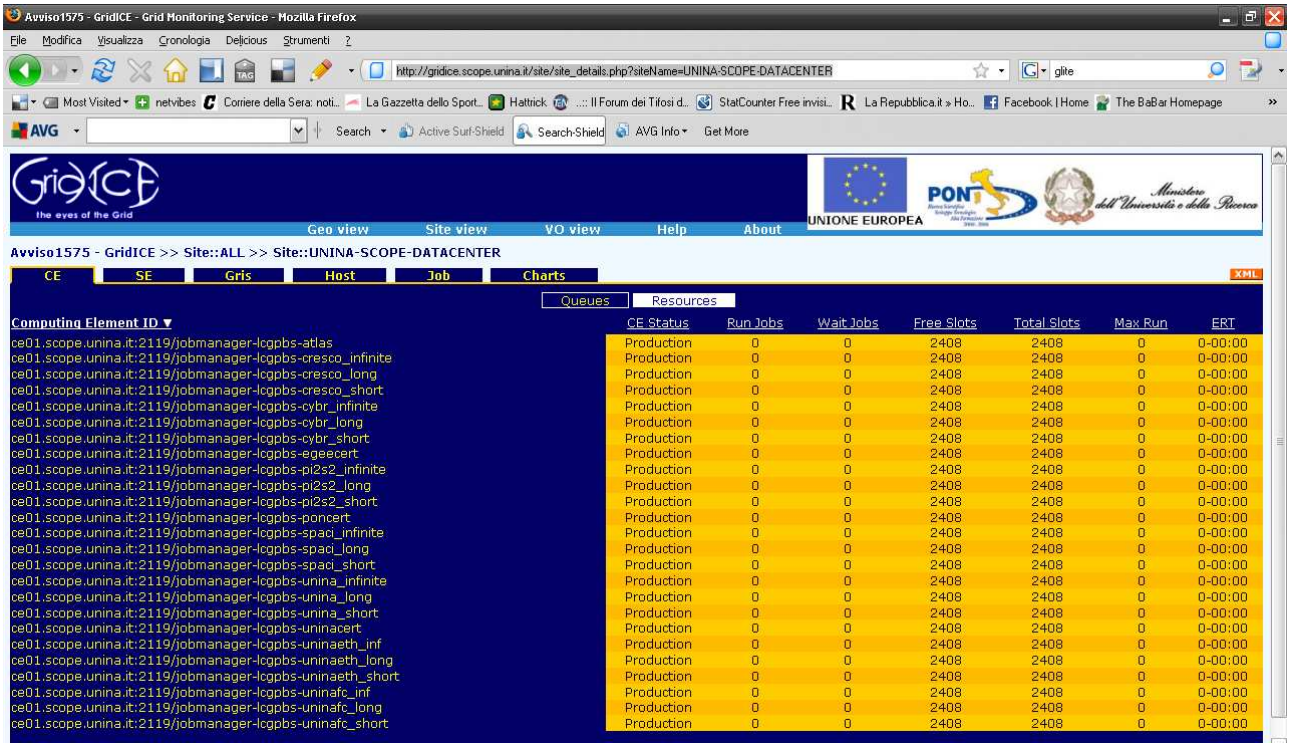
```
$ voms-proxy-info

subject   : /C=IT/O=INFN/OU=Personal Certificate/L=Federico II/CN=Diego Monorchio/CN=proxy
issuer    : /C=IT/O=INFN/OU=Personal Certificate/L=Federico II/CN=Diego Monorchio
identity  : /C=IT/O=INFN/OU=Personal Certificate/L=Federico II/CN=Diego Monorchio
type      : proxy
strength  : 512 bits
path      : /tmp/x509up_u505
timeleft  : 47:58:33
```

3.2 Scelta delle risorse da utilizzare per la computazione

Grid fornisce all'utente la scelta delle risorse, rappresentate da un CE che gestisce un insieme di Worker Nodes, sui quali desidera sottomettere un determinato job.

La scelta delle risorse per la sottomissione dei jobs va effettuata tenendo conto delle caratteristiche delle applicazioni che devono girare come la durata media dei job ed il tipo di calcolo (seriale, parallelo o multicore) .L'elenco delle code disponibili è fornito da un servizio di monitoring, denominato GridICE, ed è disponibile all'indirizzo web:

http://gridice.scope.unina.it/site/site_details.php?siteName=UNINA-SCOPE-DATACENTER



Computing Element ID	CE Status	Run Jobs	Wait Jobs	Free Slots	Total Slots	Max Run	ERT
ce01.scope.unina.it:2119/jobmanager-lcgpbs-atlas	Production	0	0	2408	2408	0	0-00:00
ce01.scope.unina.it:2119/jobmanager-lcgpbs-cresco_infinite	Production	0	0	2408	2408	0	0-00:00
ce01.scope.unina.it:2119/jobmanager-lcgpbs-cresco_long	Production	0	0	2408	2408	0	0-00:00
ce01.scope.unina.it:2119/jobmanager-lcgpbs-cresco_short	Production	0	0	2408	2408	0	0-00:00
ce01.scope.unina.it:2119/jobmanager-lcgpbs-cybr_infinite	Production	0	0	2408	2408	0	0-00:00
ce01.scope.unina.it:2119/jobmanager-lcgpbs-cybr_long	Production	0	0	2408	2408	0	0-00:00
ce01.scope.unina.it:2119/jobmanager-lcgpbs-cybr_short	Production	0	0	2408	2408	0	0-00:00
ce01.scope.unina.it:2119/jobmanager-lcgpbs-egeecert	Production	0	0	2408	2408	0	0-00:00
ce01.scope.unina.it:2119/jobmanager-lcgpbs-pi2s2_infinite	Production	0	0	2408	2408	0	0-00:00
ce01.scope.unina.it:2119/jobmanager-lcgpbs-pi2s2_long	Production	0	0	2408	2408	0	0-00:00
ce01.scope.unina.it:2119/jobmanager-lcgpbs-pi2s2_short	Production	0	0	2408	2408	0	0-00:00
ce01.scope.unina.it:2119/jobmanager-lcgpbs-poncert	Production	0	0	2408	2408	0	0-00:00
ce01.scope.unina.it:2119/jobmanager-lcgpbs-spaci_infinite	Production	0	0	2408	2408	0	0-00:00
ce01.scope.unina.it:2119/jobmanager-lcgpbs-spaci_long	Production	0	0	2408	2408	0	0-00:00
ce01.scope.unina.it:2119/jobmanager-lcgpbs-spaci_short	Production	0	0	2408	2408	0	0-00:00
ce01.scope.unina.it:2119/jobmanager-lcgpbs-unina_infinite	Production	0	0	2408	2408	0	0-00:00
ce01.scope.unina.it:2119/jobmanager-lcgpbs-unina_long	Production	0	0	2408	2408	0	0-00:00
ce01.scope.unina.it:2119/jobmanager-lcgpbs-unina_short	Production	0	0	2408	2408	0	0-00:00
ce01.scope.unina.it:2119/jobmanager-lcgpbs-uninacert	Production	0	0	2408	2408	0	0-00:00
ce01.scope.unina.it:2119/jobmanager-lcgpbs-uninaeth_inf	Production	0	0	2408	2408	0	0-00:00
ce01.scope.unina.it:2119/jobmanager-lcgpbs-uninaeth_long	Production	0	0	2408	2408	0	0-00:00
ce01.scope.unina.it:2119/jobmanager-lcgpbs-uninaeth_short	Production	0	0	2408	2408	0	0-00:00
ce01.scope.unina.it:2119/jobmanager-lcgpbs-uninafc_inf	Production	0	0	2408	2408	0	0-00:00
ce01.scope.unina.it:2119/jobmanager-lcgpbs-uninafc_long	Production	0	0	2408	2408	0	0-00:00
ce01.scope.unina.it:2119/jobmanager-lcgpbs-uninafc_short	Production	0	0	2408	2408	0	0-00:00

Gli utenti appartenenti alla VO unina.it possono utilizzare le code i cui nomi contengano le stringhe "unina", "uninafc" ed "uninaib". La differenza tra queste tre categorie consiste nel tipo di connessione sfruttata dai corrispondenti Worker Nodes per comunicare. Le code "unina" corrispondono ad una connessione di tipo "ethernet", le code "uninafc" corrispondono ad una connessione in fibra, mentre le code "uninaib" corrispondono ad una connessione di tipo "infiniband". I suffissi "short", "long" ed "infinite" indicano la durata massima dei jobs da sottomettere alle corrispondenti code:

short: durata fino a 2 h

long: durata fino a 24 h

infinite: durata fino a 72 h

Elenco delle code disponibili per gli utenti unina.it:

ce01.scope.unina.it:2119/jobmanager-lcgpbs-unina_infinite
ce01.scope.unina.it:2119/jobmanager-lcgpbs-unina_long
ce01.scope.unina.it:2119/jobmanager-lcgpbs-unina_short
ce01.scope.unina.it:2119/jobmanager-lcgpbs-uninaib_inf
ce01.scope.unina.it:2119/jobmanager-lcgpbs-uninaib_long
ce01.scope.unina.it:2119/jobmanager-lcgpbs-uninaib_short
ce01.scope.unina.it:2119/jobmanager-lcgpbs-uninafc_inf
ce01.scope.unina.it:2119/jobmanager-lcgpbs-uninafc_long
ce01.scope.unina.it:2119/jobmanager-lcgpbs-uninafc_short

ce02.scope.unina.it:2119/jobmanager-lcgpbs-unina_infinite
ce02.scope.unina.it:2119/jobmanager-lcgpbs-unina_long
ce02.scope.unina.it:2119/jobmanager-lcgpbs-unina_short
ce02.scope.unina.it:2119/jobmanager-lcgpbs-uninaib_inf
ce02.scope.unina.it:2119/jobmanager-lcgpbs-uninaib_long

ce02.scope.unina.it:2119/jobmanager-lcgpbs-uninaib_short
ce02.scope.unina.it:2119/jobmanager-lcgpbs-uninafc_inf
ce02.scope.unina.it:2119/jobmanager-lcgpbs-uninafc_long
ce02.scope.unina.it:2119/jobmanager-lcgpbs-uninafc_short

Nei prossimi paragrafi verrà descritta la procedura che un utente deve adottare per specificare le risorse che ha deciso di utilizzare per un determinato job.

3.3 Sottomissione dei jobs su Grid

L'esempio "Hello World"

Un esempio di job da sottomettere può essere il semplice "Hello World", che usufruisce del comando /bin/echo. Per sottomettere un job su grid è necessario creare un file di configurazione in formato jdl (Job Description Language) che contiene tutte le informazioni utili al fine di individuare le risorse adatte all'esecuzione del job. Nel semplice esempio "Hello World" il file jdl sarà del tipo:

```
Executable = "/bin/echo";
Arguments = "Hello World";
StdOutput = "message.txt";
StdError = "stderror";
OutputSandbox = {"message.txt","stderror"};
```

Nell'esempio proposto "Executable" indica l'eseguibile presente su macchina che dovrà effettuare il calcolo, "Arguments" indica gli argomenti dell'eseguibile, "StdOutput" e "StdError" indicano i file dove verranno rediretti rispettivamente lo standard output e lo standard error. "OutputSandbox" indica il nome del file che viene restituito da Grid in output.

E' possibile sottomettere il job utilizzando il comando:

```
$ glite-wms-job-submit -a Hello.jdl
```

L'opzione –a garantisce delegazione del proxy al wms (Workload Management System) ed è necessaria affinché il job venga correttamente sottomesso.

Una volta sottomesso il job, esso restituirà il seguente output che contiene l'ID del job come nell'esempio:

```
Connecting to the service https://wms01.scope.unina.it:7443/glite_wms_wmproxy_server
==================== glite-wms-job-submit Success ====================
The job has been successfully submitted to the WMProxy
Your job identifier is:
https://wms01.scope.unina.it:9000/5GwugI8dbmCyB9TQZNsGTg
======================================================================
```

Il formato del jobID è:

```
https://<LB_hostname>[:<port>]/<unique_string>
```

dove <unique_string> è una stringa univocamente assegnata al job e <LB_hostname> è l'host-name del server di Logging &Bookkeeping (LB) utilizzato per il JOB.

Per visionare lo stato del job occorre utilizzare il numero ID ad esso associato con il comando:

```
$ glite-wms-job-status https://wms01.scope.unina.it:9000/5GwugI8dbmCyB9TQZNsGTg
```

L'output che segue corrisponde allo stato "Running" del job

```
**************************************************************
BOOKKEEPING INFORMATION:
Status info for the Job : https://wms01.scope.unina.it:9000/5GwugI8dbmCyB9TQZNsGTg
Current Status:     Running
Status Reason:      Job successfully submitted to Globus
Destination:        ce01.scope.unina.it:2119/jobmanager-lcgpbs-uninafc_inf
Submitted:          Tue Apr  7 14:52:24 2009 CEST
**************************************************************
```

Se il jobs viene completato con successo si ottiene il seguente output:

```
**************************************************************
BOOKKEEPING INFORMATION:
Status info for the Job : https://wms01.scope.unina.it:9000/5GwugI8dbmCyB9TQZNsGTg
Current Status:     Done (Success)
Logged Reason(s):
    -
    - Job terminated successfully
Exit code:          0
Status Reason:      Job terminated successfully
Destination:        ce01.scope.unina.it:2119/jobmanager-lcgpbs-uninafc_inf
Submitted:          Tue Apr  7 14:52:24 2009 CEST
**************************************************************
```

L'opzione -o <file path> di glite-wms-job-submit permette agli utenti di specificare un file nel quale aggiungere il jobID del job sottomesso. Questo file può essere dato come argomento ad altri comandi di job management per effettuare operazioni su più di un job con un singolo comando ed è un modo conveniente per tener traccia dei jobs sottomessi. Supponiamo ad esempio di aver sottomesso due jobs, raccogliendo i jobID nel file "prova_JOBLIST" con i comandi:

```
$ glite-wms-job-submit -a -o prova_JOBLIST Ciao.jdl
$ glite-wms-job-submit -a -o prova_JOBLIST Hola.jdl
```

il contenuto del file "prova_JOBLIST" è il seguente:

```
$ more prova_JOBLIST
###Submitted Job Ids###
https://wms01.scope.unina.it:9000/0y9-abow_HgUvCQ_l-sfFQ
https://wms01.scope.unina.it:9000/VCAZHjCUfTbObmFQWl0-oQ
```

Possiamo passare in input il file "prova_JOBLIST" al comando glite-wms-job-status attraverso l'opzione –i:

```
$ glite-wms-job-status -i prova_JOBLIST
```

ottenendo l'output:

```
------------------------------------------------------------------
1 : https://wms01.scope.unina.it:9000/0y9-abow_HgUvCQ_l-sfFQ
2 : https://wms01.scope.unina.it:9000/VCAZHjCUfTbObmFQWl0-oQ
a : all
q : quit
------------------------------------------------------------------

Choose one or more jobId(s) in the list - [1-2]all:
```

Possiamo dunque selezionare i jobs della lista per i quali vogliamo monitorare lo stato.

Nel messaggio relativo al BOOKKEEPING INFORMATION, ottenuto con il comando, l'argomento "Destination" indica che il job viene indirizzato verso il Computing Element (CE) ce01.scope.unina.it. Nel caso in cui, come in precedenza, il file jdl non contenga alcun requisito riguardo la scelta del CE da utilizzare per la computazione, il job girerà sulla macchina con capacità computazionale migliore scelta da Grid in maniera trasparente all'utente.

Nel caso in cui invece l'utente desideri eseguire il job su uno specifico CE ed in particolare su una specifica coda di job, in base ai requisiti della propria applicazione, leggere il prossimo paragrafo.

Per ottenere i files di output di un job si può utilizzare il comando glite-wms-job-output passandogli come argomento il jobID opportuno:

```
$ glite-wms-job-output https://wms01.scope.unina.it:9000/5GwugI8dbmCyB9TQZNsGTg

Connecting to the service https://wms01.scope.unina.it:7443/glite_wms_wmproxy_server
================================================================================
                        JOB GET OUTPUT OUTCOME

Output sandbox files for the job:
https://wms01.scope.unina.it:9000/5GwugI8dbmCyB9TQZNsGTg
have been successfully retrieved and stored in the directory:
/tmp/jobOutput/diego_5GwugI8dbmCyB9TQZNsGTg

================================================================================
```

A questo punto l'output sarà copiato nella directory /tmp/jobOutput/diego_5GwugI8dbmCyB9TQZNsGTg/.

Il contenuto del file di output "message.txt" sarà la stringa "Hello World":

```
$ ls /tmp/jobOutput/diego_5GwugI8dbmCyB9TQZNsGTg/
message.txt  sterror
$ more /tmp/jobOutput/diego_5GwugI8dbmCyB9TQZNsGTg/message.txt
Hello World
```

Sottomissione di jobs con esportazione dell'eseguibile e di input files

Nell'esempio "Hello World" il job che viene eseguito fa uso solo del comando di sistema /bin/echo che risiede sui worker nodes sui quali il job verrà eseguito (oltre che sulla User Interface da cui viene sottomesso).

Supponiamo ora di aver compilato sulla UI un programma in linguaggio C++ e di avere un eseguibile chiamato simulazione.exe in /home/<user_name>/SIMULAZIONE. Affinché il job possa essere eseguito sui Worker Nodes è necessario che il file eseguibile venga opportunamente esportato. A tale scopo è necessario aggiungere il campo "InputSandbox" al file jdl:

```
Executable = "Simulazione.exe";
StdOutput = "mc.out";
StdError = "mc.err";
Arguments = "aaa";
InputSandbox = "/home/<user_name>/SIMULAZIONE/Simulazione.exe";
OutputSandbox = {"mc.out","mc.err"};
```

Il campo "InputSandbox" i file eseguibili da inviare (assieme al file JDL) in input al worker node (WN) che eseguirà il job. Inoltre se l'esecuzione del job necessita di altri file di dati in input, occorre utilizzare tale sintassi:

```
InputSandbox = {"/home/<user_name>/SIMULAZIONE/Simulazione.exe";,
"/home/<user_name>/SIMULAZIONE/input1";"…"}
```

Nel caso in cui il job necessiti di più argomenti utilizzare tale sintassi nel file jdl:

```
Arguments = "arg1 arg2 …";
```

L'argomento "InputSandbox" è necessario nel caso in cui la macchina che deve eseguire un determinato job non contiene l'eseguibile o non contiene i files di input necessari.

Sottomissione di jobs con specifica dei requisiti per le risorse da utilizzare.

Gli utenti possono specificare dei requisiti per le risorse che intendono utilizzare per l'esecuzione di un job. Ad esempio un utente che vuole eseguire un job di breve durata troverà conveniente utilizzare le code "short" che, come visto nel paragrafo 3.2, impongono un limite massimo alla durata dei job sottomessi.

L'opportunità di scegliere una specifica coda opportunità viene garantita inserendo nel file jdl il requisito:

```
requirements = other.GlueCEUniqueID=="ce01.scope.unina.it:2119/jobmanager-lcgpbs-unina_short";
```

dove:

requirements: indica la presenza di un requisito nel file jdl.

other.GlueCEUniqueID: indica che il requisito si riferisce alla scelta di uno specifico CE.

"ce01.scope.unina.it:2119/jobmanager-lcgpbs-unina_short" indica che viene scelto il CE denominato ce01.scope.unina.it:2119 per la sottomissione del job che avverrà sulla coda jobmanager-lcgpbs-unina_short.

3.4 Gestione dei dati

Gli esempi finora illustrati prevedono l'uso dei campi InputSandbox e OutputSandbox che forniscono il trasferimento di files di piccole dimensioni necessari per l'esecuzione del job e per il controllo dei suoi risultati. I files di output specificati in OutputSandbox vengono trasferiti al termine del job sul WMS e possono essere scaricati localmente sulla UI attraverso il comando

glite-wms-job-output

come descritto in precedenza. È importante sottolineare che il WMS non fornisce un servizio di storage vero e proprio e dunque impone un limite massimo di 100 Mb sulle dimensioni dei singoli files di output. Inoltre i files scritti sul WMS vengono periodicamente cancellati dal sistema (con cadenza settimanale) e dunque l'utente deve avere cura di prelevare tempestivamente l'output a cui è interessato. Qualora si decida di sottomettere un numero molto elevato di jobs, tale che le dimensioni complessive dei files di output da scrivere raggiungano le decine di Gb, è buona norma destinare l'output su uno Storage Element. La procedura per scrivere su uno SE è illustrata nel seguente esempio.

I membri della VO unina.it hanno i permessi di scrittura e di lettura nell'area /grid/unina.it del filesystem virtuale. Ciò significa che i Logical File Names (LFN) dei files che verranno scritti sugli Storage Elements saranno caratterizzati dal path:

/grid/unina.it/<sub_path>

L'accesso ai files può avvenire attraverso i comandi lfc-* ed lcg-*.

Per creare una directory di test dove scrivere il nostro output utilizziamo il comando lfc-mkdir:

```
$ lfc-mkdir  /grid/unina.it/diegoDIR
```

Supponiamo di voler copiare il file Ciao.txt dalla UI ad uno Storage Element e di volerlo registrare sul Logical File Catalogue. Quest'operazione può essere effettuata attraverso il comando lcg-cr:

```
$ lcg-cr Ciao.txt --vo unina.it -l lfn://grid/unina.it/diegoDIR/Ciao.txt
```

Dove l'opzione il Logical File Name "lfn://grid/unina.it/diegoDIR/Ciao.txt" viene assegnato attraverso l'opzione –l mentre l'opzione "--vo unina.it" specifica la VO di appartenenza. Siccome non è stato specificato uno SE sul quale scrivere il file, esso verrà assegnato dal sistema. L'output che riceveremo è il seguente:

```
[BDII] wms01.scope.unina.it:2170: Warning, no GlueVOInfo information found about tag '(null)' and SE
'se01.scope.unina.it'
guid:160c18c3-5105-4919-b700-a2a9b79e8297
```

da cui si evince che il file è stato scritto sullo SE se01.scope.unina.it e che gli è stato assegnato il guid (Grid Unique Identifier) 160c18c3-5105-4919-b700-a2a9b79e8297.

Per specificare lo SE su cui scrivere il file si può utilizzare l'opzione –d :

```
$ lcg-cr Ciao.txt --vo unina.it -l lfn://grid/unina.it/diegoDIR/Ciao_2.txt -d se01.scope.unina.it
[BDII] wms01.scope.unina.it:2170: Warning, no GlueVOInfo information found about tag '(null)' and SE
'se01.scope.unina.it'
guid:24971448-60cc-4ace-b916-cfd0eac5515f
```

La lista dei files presenti all'interno di un certo path si ottiene attraverso il comando lcg-ls:

```
$ lcg-ls -l lfn:/grid/unina.it/diegoDIR
-rw-rw-r--   1   105   101      10 Ciao.txt
-rw-rw-r--   1   105   101      10 Ciao_2.txt
```

Supponiamo ora di lanciare l'eseguibile Simulazione.exe che produce il file di output mc.out e di voler scrivere tale file sullo SE. A tal scopo è necessario introdurre il campo "OutputData" al file jdl come segue:

```
Executable = "Simulazione.exe";
StdOutput = "mc.log";
StdError = "mc.err";
Arguments = "0 0 1";
InputSandbox = "/home/diego/SIMULAZIONE/Simulazione.exe";
OutputSandbox = {"mc.log","mc.err"};
requirements = other.GlueCEUniqueID=="ce01.scope.unina.it:2119/jobmanager-lcgpbs-unina_short";
OutputData = {
[
OutputFile="mc.out";
LogicalFileName="lfn:/grid/unina.it/diegoDIR/mc.out";
StorageElement = "se01.scope.unina.it";
]
};
```

dove OutputFile indica il nome del file di output, LogicalFileName indica il suo logical file name da assegnare al file e StorageElement specifica lo SE su cui scrivere.

Il job va sottomesso con l'usuale comando

```
$ glite-wms-job-submit -o jobID.txt -a  Simulazione.jdl
```

Al termine dell'esecuzione è possibile verificare la presenza del file di output con lcg-ls:

```
$ lcg-ls -l lfn:/grid/unina.it/diegoDIR
-rw-rw-r--   1   105   101      10 Ciao.txt
-rw-rw-r--   1   105   101      10 Ciao_2.txt
-rw-rw-r--   1   105   101      26 mc.out
```

Per poter leggere copiare il file di output dallo Storage Element alla User Interface si può utilizzare il comando lcg-cp:

```
$ lcg-cp  lfn:/grid/unina.it/diegoDIR/mc.out mc.out_local
```

Il file è ora disponibile localmente sulla UI e può essere letto dall'utente:

```
$ ls mc.out_local
mc.out_local
```

Nel caso in cui l'eseguibile debba leggere un file di input che risiede sullo SE va aggiunto il campo InputData al file jdl:

```
InputData = {LogicalFileName="lfn:/grid/unina.it/diegoDIR/inputFile"};
```

3.5 Sottomissione di jobs per il calcolo parallelo

In questa sezione sarà illustrata la procedura per la sottomissione di jobs paralleli sulla grid.

Il protocollo di comunicazione tra i nodi è il **Message Passing Interface** (MPI). Rispetto alla sottomissione di jobs non paralleli, sarà necessario introdurre una modifica al file JDL ed utilizzare un wrapper che provveda a copiare via ssh l'eseguibile MPI sui vari nodi allocati per l'esecuzione.

Il sito unina.it permette di utilizzare diverse implementazioni di MPI:

- MPICH
- MPICH2
- OPENMPI
- ….

ciascuna caratterizzata da un proprio compilatore. Proponiamo di seguito un esempio di job parallelo che fa uso del compilatore MPICH.

Esempio di sottomissione di jobs MPI

Creiamo una directory di test:

```
$ mkdir MPI_TEST_MPICH
$ cd MPI_TEST_MPICH/
```

Copiamo nella directory appena creata i seguenti files:

cpi.c (codice sorgente. Implementa il calcolo di pi greco utilizzando in parallelo più nodi)

```
#include "mpi.h"
#include <stdio.h>
#include <math.h>

double f( double );
double f( double a )
{
    return (4.0 / (1.0 + a*a));
}

int main( int argc, char *argv[])
{
    int done = 0, n, myid, numprocs, i;
```

```
    double PI25DT = 3.14159265358979323846264338327950288419716939937510;
    double mypi, pi, h, sum, x;
    double startwtime = 0.0, endwtime;
    int  namelen;
    char processor_name[MPI_MAX_PROCESSOR_NAME];

    MPI_Init(&argc,&argv);
    MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
    MPI_Comm_rank(MPI_COMM_WORLD,&myid);
    MPI_Get_processor_name(processor_name,&namelen);

    fprintf(stderr,"Process %d on %s\n",
            myid, processor_name);

    n = 0;
    while (!done)
    {
        if (myid == 0)
        {
            if (n==0) n=100; else n=0;

            startwtime = MPI_Wtime();
        }
        MPI_Bcast(&n, 1, MPI_INT, 0, MPI_COMM_WORLD);
        if (n == 0)
            done = 1;
        else
        {
            h   = 1.0 / (double) n;
            sum = 0.0;
            for (i = myid + 1; i <= n; i += numprocs)
            {
                x = h * ((double)i - 0.5);
                sum += f(x);
            }
            mypi = h * sum;

            MPI_Reduce(&mypi, &pi, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD);

            if (myid == 0)
                {
                printf("pi is approximately %.16f, Error is %.16f\n",
                       pi, fabs(pi - PI25DT));
                      endwtime = MPI_Wtime();
                      printf("wall clock time = %f\n",
                             endwtime-startwtime);
                }
        }
    }
    MPI_Finalize();

    return 0;
}
```

compile.sh (permette di compilare il codice sorgente utilizzando il compilatore mpich):

```
/opt/mpich-1.2.7p1/bin/mpicc $1
```

mpi-hooks.sh

```
#!/bin/sh

# This function will be called before the MPI executable is started.
# You can, for example, compile the executable itself.
#
pre_run_hook () {

  # Compile the program.
  echo "Compiling ${I2G_MPI_APPLICATION}"

  # Actually compile the program.
  cmd="mpicc ${MPI_MPICC_OPTS} -o ${I2G_MPI_APPLICATION} ${I2G_MPI_APPLICATION}.c"
  echo $cmd
  $cmd
  if [ ! $? -eq 0 ]; then
     echo "Error compiling program.  Exiting..."
     exit 1
  fi

  # Everything's OK.
  echo "Successfully compiled ${I2G_MPI_APPLICATION}"

  return 0
}

# This function will be called before the MPI executable is finished.
# A typical case for this is to upload the results to a storage element.
post_run_hook () {
  echo "Executing post hook."
  echo "Finished the post hook."

  return 0
}
```

mpi-start-wrapper.sh

```
#!/bin/bash

# Pull in the arguments.
MY_EXECUTABLE=`pwd`/$1
MPI_FLAVOR=$2

# Convert flavor to lowercase for passing to mpi-start.
MPI_FLAVOR_LOWER=`echo $MPI_FLAVOR | tr '[:upper:]' '[:lower:]'`

# Pull out the correct paths for the requested flavor.
eval MPI_PATH=`printenv MPI_${MPI_FLAVOR}_PATH`

# Ensure the prefix is correctly set.  Don't rely on the defaults.
eval I2G_${MPI_FLAVOR}_PREFIX=$MPI_PATH
export I2G_${MPI_FLAVOR}_PREFIX

# Touch the executable.  It exist must for the shared file system check.
# If it does not, then mpi-start may try to distribute the executable
```

```
# when it shouldn't.
touch $MY_EXECUTABLE

# Setup for mpi-start.
export I2G_MPI_APPLICATION=$MY_EXECUTABLE
export I2G_MPI_APPLICATION_ARGS=
export I2G_MPI_TYPE=$MPI_FLAVOR_LOWER
export I2G_MPI_PRE_RUN_HOOK=mpi-hooks.sh
export I2G_MPI_POST_RUN_HOOK=mpi-hooks.sh

# If these are set then you will get more debugging information.
export I2G_MPI_START_VERBOSE=1
export I2G_MPI_START_DEBUG=1

chmod 755 $MY_EXECUTABLE

# Invoke mpi-start.
$I2G_MPI_START
```

wrapper-MPICH.jdl (file JDL per la sottomissione)

```
Type = "Job";
JobType = "MPICH";
CpuNumber = 33;
Executable = "mpi-start-wrapper.sh";
Arguments = "cpi MPICH";
StdOutput = "mpich-test.out";
StdError = "mpich-test.err";
InputSandbox = {"mpi-start-wrapper.sh","mpi-hooks.sh","cpi.c"};
OutputSandbox = {"mpich-test.err","mpich-test.out"};
#Requirements =
#  Member("MPI-START", other.GlueHostApplicationSoftwareRunTimeEnvironment)
#  && Member("MPICH", other.GlueHostApplicationSoftwareRunTimeEnvironment);
Requirements = other.GlueCEUniqueID == "ce02.scope.unina.it:2119/jobmanager-lcgpbs-unina_short";
RetryCount = 1;
```

Il comando compile.sh permette di compilare localmente il programma utilizzando la versione mpich del compilatore.

```
$ source compile.sh cpi.c
```

Vengono prodotti il file oggetto cpi.o e l'eseguibile a.out. La sottomissione del job sui nodi della griglia può essere eseguita nel modo usuale:

```
$ glite-wms-job-submit -a -o jobID.txt wrapper-MPICH.jdl
```

Con le impostazioni adottate in wrapper-MPICH.jdl il codice sorgente viene copiato e compilato sui worker nodes. Inoltre vengono copiati anche i files mpi-start-wrapper.sh e mpi-hooks.sh che forniscono le corrette impostazioni di variabili di ambiente necessarie per l'esecuzione del job.

Al termine del job, l'output può essere recuperato nel modo usuale:

```
$ glite-wms-job-output -i jobID.txt
```

3.6 Rinnovo automatico delle credenziali (myproxy service)

Il proxy delle credenziali generato con il comando voms-proxy-init ha una durata di default di 12 ore oppure una durata scelta dall'utente con l'utilizzo di una specifica opzione (vedi paragrafo: Generazione proxy delle credenziali e Sottomissione job su Grid). I job sottomessi dunque possono stare in stato di "Running" solo per tale periodo, al termine del quale vengono distrutti nonostante non abbiano terminato i propri calcoli.

Tale limitazione, relativa ai job di lunga durata che vanno oltre l'intervallo di tempo in cui il proxy iniziale rimane valido, può essere risolta con l'utilizzo di un servizio fornito agli utenti denominato MyProxy service, che consente di rinnovare in maniera automatica il proxy delle credenziali al suo scadere.

La procedura di sottomissione dei job è simile a quella già descritta nei paragrafi precedenti, con l'aggiunta dei comandi che gestiscono il myproxy delle credenziali.

Il primo passo prevede dunque, come già visto, la creazione del certificato proxy utente utilizzando il comando:

```
$ voms-proxy-init --voms unina.it

Enter GRID pass phrase:
Your identity: /C=IT/O=INFN/OU=Personal Certificate/L=Federico II/CN=Diego Monorchio
Creating temporary proxy ..................................... Done
Contacting  voms01.scope.unina.it:15003 [/C=IT/O=INFN/OU=Host/L=Federico
II/CN=voms01.scope.unina.it] "unina.it" Done
Creating proxy ..................................... Done
Your proxy is valid until Thu Apr  9 02:41:54 2009
```

Le info del proprio certificato proxy vengono visualizzate con il comando:

```
$ voms-proxy-info

subject   : /C=IT/O=INFN/OU=Personal Certificate/L=Federico II/CN=Diego Monorchio/CN=proxy
issuer    : /C=IT/O=INFN/OU=Personal Certificate/L=Federico II/CN=Diego Monorchio
identity  : /C=IT/O=INFN/OU=Personal Certificate/L=Federico II/CN=Diego Monorchio
type      : proxy
strength  : 512 bits
path      : /tmp/x509up_u505
timeleft  : 11:57:02
```

A questo punto è possibile generare il myproxy delle credenziali con il comando:

```
$ myproxy-init -d –n
```

Il comando myproxy-init memorizza le credenziali sul myproxy server, disponibili di default per 168 ore. In questo periodo viene eseguito il rinnovo del proxy delle credenziali, che avviene in maniera automatica allo scadere dello stesso proxy.

L'opzione –d istruisce il server ad associare il DN dell'utente al proxy e l'opzione –n evita l'uso di una password per accedere al proxy così che il WMS può effettuare il rinnovo automaticamente.

L'output restituito è il seguente:

```
Your identity: /C=IT/O=INFN/OU=Personal Certificate/L=Federico II/CN=Diego Monorchio
Enter GRID pass phrase for this identity:
Creating proxy ..................................... Done
Proxy Verify OK
Your proxy is valid until: Wed Apr 15 14:48:07 2009
A proxy valid for 168 hours (7.0 days) for user /C=IT/O=INFN/OU=Personal Certificate/L=Federico
II/CN=Diego Monorchio now exists on myproxy01.scope.unina.it.
```

Le informazioni del proprio certificato myproxy vengono visualizzate con il comando:

```
$ myproxy-info –d

username: /C=IT/O=INFN/OU=Personal Certificate/L=Federico II/CN=Diego Monorchio
owner: /C=IT/O=INFN/OU=Personal Certificate/L=Federico II/CN=Diego Monorchio
  timeleft: 167:51:50  (7.0 days)
```

Nel caso in cui si voglia estendere la durata del myproxy usare l'opzione –c hours in questo modo:

```
$ myproxy-init -d -n -c 200   (estensione a 200 ore)

Your identity: /C=IT/O=INFN/OU=Personal Certificate/L=Federico II/CN=Diego Monorchio
Enter GRID pass phrase for this identity:
Creating proxy ..................................... Done
Proxy Verify OK
Your proxy is valid until: Fri Apr 17 00:22:40 2009
A proxy valid for 200 hours (8.3 days) for user /C=IT/O=INFN/OU=Personal Certificate/L=Federico
II/CN=Diego Monorchio now exists on myproxy01.scope.unina.it.
```

Nel caso in cui si voglia distruggere il myproxy delle credenziali eseguire il comando:

```
$ myproxy-destroy –d

Default MyProxy credential for user /C=IT/O=INFN/OU=Personal Certificate/L=Federico II/CN=Diego
Monorchio was successfully removed.
```

Una volta generato il myproxy, è possibile sottomettere jobs, monitorarli e importarne l'output attraverso i comandi glite-wms-job-submit , glite-wms-job-status e glite-wms-job-output come descritto nei paragrafi precedenti. Un fattore importante da ricordare è che non è possibile eseguire questi tre comandi quando il proxy delle credenziali generato con voms-proxy-init ha un timeleft (validità) inferiore a 20 minuti (il timeleft può essere visualizzato con il comando voms-proxy-info).

In questo caso dunque, l'utente che desideri eseguire uno di questi tre comandi deve rigenerare il proxy delle credenziali con il solito comando voms-proxy-init, senza occuparsi di ricreare il myproxy delle credenziali, a meno che quest'ultimo non sia scaduto.

Infine nell'intervallo di tempo in cui il myproxy delle credenziali sia valido, l'utente ha la possibilità di eseguire qualsiasi operazione sul proxy delle credenziali (voms-proxy-destroy, voms-proxy-init), non determinando la distruzione dei job in running.