UNICORE Web Portal: Administrator Manual

UNICORE Team

Document Version:	1.0.0
Component Version:	2.2.0
Date:	27 10 2016

This work is co-funded by the EC EMI project under the FP7 Collaborative Projects Grant Agreement Nr. INFSO-RI-261611.

Contents

1 Overview **1**

2 Installation and prerequisites **1**

2.1 Prerequisites . . . 1

2.2 Download . . . 1

2.3 Installation and configuration . . . 1

2.4 Signing Java webstart archives and applet(s) . . . 2

2.5 Preferences file . . . 2

2.6 Logging . . . 2

3 Upgrading from previous versions **2**

3.1 Upgrading from 2.1.0 . . . 2

3.2 Upgrading from 2.0.0 . . . 3

3.3 Migrating from 1.2.0 . . . 3

4 Portal configuration - credentials **4**

4.1 Configuration file . . . 4

4.2 Credential and truststore options . . . 5

4.3 Truststore examples . . . 10

4.4 Client options . . . 11

5 Portal configuration - authentication **14**

5.1 Portal authentication and registration options . . . 15

5.2 Using Unity . . . 20

5.3 Single logout configuration . . . 24

5.4 Account registration . . . 24

6 Portal configuration - server options, preferences and UI **29**

6.1 Server options . . . 29

6.2 User preferences . . . 30

6.3 Core and UI configuration options . . . 31

Overview

The UNICORE Portal is a web client for the UNICORE Grid middleware. The Portal presents a user friendly interface for all the UNICORE basic services as well as basic functionality of the UNICORE workflow system.

It offers the following functions:

- Job submission and management for UNICORE native interfaces
- Data movement (upload, download, server-to-server copy, etc) using the UNICORE storage management functions and available data transfer protocols
- Storage functions like listing, creating, editing and deleting files and folders
- Partial UNICORE workflow system support - there is a basic workflow editor and a table displaying user workflows and working directories. The Portal also supports the submission of workflow templates. You can read more about workflow templates here https://sourceforge.net/-p/unicore/wiki/WorkflowTemplates/.
- Information about the sites that the user has access to
- Various methods of authentication
- Possibility to change the language of the User Interface (UI) at login

For more information about UNICORE visit http://www.unicore.eu.

Installation and prerequisites

Prerequisites

The newest portal server requires Java version 1.8 or later (we recommend using OpenJDK Java 8) and is currently restricted to a Linux/UNIX operating system.

Download

You can get the latest version from the SourceForge UNICORE download page.

Installation and configuration

Unzip the archive into a convenient folder, which will result in the following directory structure

- bin: start/stop/status scripts

- lib: application libraries
- conf: configuration files
- webcontent: libraries for Java webstart application(s) used by the portal as well as VAADIN folder with UI theme files, icons, CSS style sheets
- logs: default log directory
- doc: readme, changelog and others

Signing Java webstart archives and applet(s)

The application includes parts that are run via Java webstart or the Java applet mechanism. These need to be signed using the portal credential. Signing requires the "jarsigner" application from the Java Development Kit (JDK).

A script *unicore-portal-sign.sh* is provided in the bin folder which signs the jar files. Please review this script and provide the proper values for your portal credential.

This process has to be done only once before running the portal for the first time.

Preferences file

The main configuration file is <portal home>/conf/portal.properties This file contains central settings such as host and port of the server. You must review it before starting the portal.

Logging

The portal log files are located in <portal home>/logs folder. By default the level of logging is set to INFO but you can edit that in the <portal home>/conf/logging.properties file.

Upgrading from previous versions

Upgrading from 2.1.0

1. Update libs folder to latest version
2. Update <unicore-portal>/conf/web.xml
3. There are three new properties, which can be set in portal.properties, but neither is mandatory:

```
portal.ui.menu.navigation.default=<the path to a default ↩
     start up view, which the user will be redirected to ↩
     after login>
portal.core.workflow.templates.enable=<true or false ->  ↩
   the workflow templates can be disabled from the UI  ↩
   of the Portal; the default value is true>
portal.core.defaultApplication=<the name of the  ↩
   application, which will be preset in the combo box  ↩
   at submitting a new job>
```

Upgrading from 2.0.0

1. Update libs and webcontent/VAADIN folders to latest version
2. Merge portal.properties file. -There is some of new properties concerning the discovery API. -The properties with prefix userprofiles.* have been renamed to portal.userprofiles.*

Migrating from 1.2.0

Due to numerous changes in the structure and the internal logic of the portal, we recommend a fresh installation of the new version 2.1.0. You can keep the old user *data* folder as well as the *conf* folder after adjusting the files as mentioned in point 4.

1. Install the new version.
2. Copy and paste the *data* folder with the user information from the old version into the newly installed portal.
3. Keep in mind to link the workspace to the same location as it was in the older version of the portal
4. It is safe to copy your *conf* folder and all its files taking into a consideration the following notes:

   ```
   4.1. Do not directly use the old portal.properties file! ↩
        There are numerous changes as well as a lot of new
   properties that have been added. Instead try to compare  ↩
      and merge the two versions.
   ```

```
For example:
The authentication properties' "implementationClass":
'portal.registration.facility.REG-SAML.implementationClass'
has changed to "type":
'portal.registration.facility.REG-SAML.type'
```

```
whereas 'type' is one of the following: 'DEMO', 'SAML', 'TLS', ' ↩
    username', 'KRB5';

The following properties are obsolete and have been removed:
-portal.grid.lists.childrenLimit
-portal.grid.lists.chunkSize
-portal.grid.JobViewShowSite
-portal.grid.JobViewShowQueue
-portal.grid.JobViewShowEndTime
```

```
4.2. Keep in mind that the web.xml file has changed  ↩
    drastically. Please remove the old web.xml file and use  ↩
    the one from the new version.

4.3. portalPlugin.xml file is obsolete. You can delete it.
```

Portal configuration - credentials

Configuration file

By default, the portal checks for the existence of a file <portal home>/conf/portal.properties and reads settings from there.

The configuration file can contain default settings for many options, which are given in the form <option name>=<value> where <option name> is the attribute. The property values may contain variables in the form `${VAR_X}`, which are automatically replaced with the environmental variable values with the same name.

In the default configuration, DEMO-CA certificates are accepted, and the portal uses a certificate issued by the DEMO-CA. This will allow to test the portal against a UNICORE server demo installation. Furthermore, a "demo user" login is provided, however the credentials used by this account can be configured. If you wish to modify anything, please refer to the comments in the property file.

For example, to set your keystore, truststore and registry, the file would contain the following settings

```
portal.credential.path=<your credential>
portal.credential.password=XXXXXXX
portal.truststore.type=keystore
portal.truststore.keystorePath=<your trustore>
portal.truststore.keystorePassword=XXXXXX
portal.core.registries=https://localhost:8080/DEMO-SITE/services/ ↩
    Registry?res=default_registry
```

Note
To protect your passwords, you should make the file non-readable by others, for example on Unix using a command such as *chmod 600 preferences*

Credential and truststore options

In general you need a keystore containing your identity in order to use UNICORE, as well as a truststore file (or directory) containing trusted certificates. There are also other options available for authentication discussed below.

A full list of options related to credential and truststore management is available in the following tables.

Table 1: Credential properties

Property name	**Type**	**Default value / mandatory**	**Description**
`portal.credential.path`	filesystem path	*mandatory to be set*	Credential location. In case of *jks*, *pkcs12* and *pem* store it is the only location required. In case when credential is provided in two files, it is the certificate file path.
`portal.credential.format`	[jks, pkcs12, der, pem]	-	Format of the credential. It is guessed when not given. Note that *pem* might be either a PEM keystore with certificates and keys (in PEM format) or a pair of PEM files (one with certificate and second with private key).
`portal.credential.password`	string	-	Password required to load the credential.
`portal.credential.keyPath`	string	-	Location of the private key if stored separately from the main credential (applicable for *pem* and *der* types only),

Table 1: (continued)

Property name	**Type**	**Default value / mandatory**	**Description**
`portal.credential.keyPassword`	string	-	Private key password, which might be needed only for *jks* or *pkcs12*, if key is encrypted with different password then the main credential password.
`portal.credential.keyAlias`	string	-	Keystore alias of the key entry to be used. Can be ignored if the keystore contains only one key entry. Only applicable for *jks* and *pkcs12*.

Table 2: Truststore properties

Property name	**Type**	**Default value / mandatory**	**Description**
`portal.truststore.allowProxy`	[ALLOW, DENY]	`ALLOW`	Controls whether proxy certificates are supported.
`portal.truststore.type`	[keystore, openssl, directory]	*mandatory to be set*	The truststore type.
`portal.truststore.updateInterval`	integer number	`600`	How often the truststore should be reloaded, in seconds. Set to negative value to disable refreshing at runtime. *(runtime updateable)*
--- *Directory type settings* ---			
`portal.truststore.directoryConnectionTimeout`	integer number	`15`	Connection timeout for fetching the remote CA certificates in seconds.

Table 2: (continued)

Property name	**Type**	**Default value / mandatory**	**Description**
`portal.truststore.directoryDiskCachePath`	filesystem path	-	Directory where CA certificates should be cached, after downloading them from a remote source. Can be left undefined if no disk cache should be used. Note that directory should be secured, i.e. normal users should not be allowed to write to it.
`portal.truststore.directoryEncoding`	[PEM, DER]	`PEM`	For directory truststore controls whether certificates are encoded in PEM or DER.
`portal.truststore.directoryLocations.*`	list of properties with a common prefix	-	List of CA certificates locations. Can contain URLs, local files and wildcard expressions. *(runtime updateable)*
--- *Keystore type settings* ---			
`portal.truststore.keystoreFormat`	string	-	The keystore type (jks, pkcs12) in case of truststore of keystore type.
`portal.truststore.keystorePassword`	string	-	The password of the keystore type truststore.
`portal.truststore.keystorePath`	string	-	The keystore path in case of truststore of keystore type.
--- *Openssl type settings* ---			
`portal.truststore.opensslNewStoreFormat`	[true, false]	`false`	In case of openssl truststore, specifies whether the trust store is in openssl 1.0.0+ format (true) or older openssl 0.x format (false)

Table 2: (continued)

Property name	**Type**	**Default value / mandatory**	**Description**
portal.truststore.opensslNsMode	[GLOBUS_EUGRIDPMA, EU-GRIDPMA_GLOBUS, GLOBUS, EUGRIDPMA, GLOBUS_EUGRIDPMA_REQUIRE, EU-GRIDPMA_GLOBUS_REQUIRE, GLOBUS_REQUIRE, EU-GRIDPMA_REQUIRE, EU-GRIDPMA_AND_GLOBUS, EU-GRIDPMA_AND_GLOBUS_REQUIRE, IGNORE]	EUGRIDPMA_GLOBUS	In case of openssl truststore, controls which (and in which order) namespace checking rules should be applied. The *REQUIRE* settings will cause that all configured namespace definitions files must be present for each trusted CA certificate (otherwise checking will fail). The *AND* settings will cause to check both existing namespace files. Otherwise the first found is checked (in the order defined by the property).
portal.truststore.opensslPath	filesystem path	/etc/grid-security/certificates	Directory to be used for opeenssl truststore.
--- *Revocation settings* ---			
portal.truststore.crlConnectionTimeout	integer number	15	Connection timeout for fetching the remote CRLs in seconds (not used for Openssl truststores).
portal.truststore.crlDiskCachePath	filesystem path	-	Directory where CRLs should be cached, after downloading them from remote source. Can be left undefined if no disk cache should be used. Note that directory should be secured, i.e. normal users should not be allowed to write to it. Not used for Openssl truststores.

Table 2: (continued)

Property name	**Type**	**Default value / mandatory**	**Description**
`portal.truststore.crlLocations.*`	list of properties with a common prefix	-	List of CRLs locations. Can contain URLs, local files and wildcard expressions. Not used for Openssl truststores. *(runtime updateable)*
`portal.truststore.crlMode`	[REQUIRE, IF_VALID, IGNORE]	`IF_VALID`	General CRL handling mode. The IF_VALID setting turns on CRL checking only in case the CRL is present.
`portal.truststore.crlUpdateInterval`	integer number	`600`	How often CRLs should be updated, in seconds. Set to negative value to disable refreshing at runtime. *(runtime updateable)*
`portal.truststore.ocspCacheTtl`	integer number	`3600`	For how long the OCSP responses should be locally cached in seconds (this is a maximum value, responses won't be cached after expiration)
`portal.truststore.ocspDiskCache`	filesystem path	-	If this property is defined then OCSP responses will be cached on disk in the defined folder.
`portal.truststore.ocspLocalResponders.<NUMBER>`	list of properties with a common prefix	-	Optional list of local OCSP responders
`portal.truststore.ocspMode`	[REQUIRE, IF_AVAILABLE, IGNORE]	`IF_AVAILABLE`	General OCSP ckecking mode. REQUIRE should not be used unless it is guaranteed that for all certificates an OCSP responder is defined.
`portal.truststore.ocspTimeout`	integer number	`10000`	Timeout for OCSP connections in miliseconds.

Table 2: (continued)

Property name	Type	Default value / mandatory	Description
portal.truststore.revocationOrder	[CRL_OCSP, OCSP_CRL]	OCSP_CRL	Controls overal revocation sources order
portal.truststore.revocationUseAll	[true, false]	false	Controls whether all defined revocation sources should be always checked, even if the first one already confirmed that a checked certificate is not revoked.

Truststore examples

- Here are some examples for commonly used truststore configurations.

Directory truststore with a minimal set of options

```
portal.truststore.type=directory
portal.truststore.directoryLocations.1=/trust/dir/*.pem
```

- Directory truststore with more options

```
portal.truststore.type=directory
portal.truststore.allowProxy=DENY
portal.truststore.updateInterval=1234
portal.truststore.directoryLocations.1=/trust/dir/*.pem
portal.truststore.directoryLocations.2=http://caserver/ca.pem
portal.truststore.directoryEncoding=PEM
portal.truststore.directoryConnectionTimeout=100
portal.truststore.directoryDiskCachePath=/tmp
portal.truststore.crlLocations.1=/trust/dir/*.crl
portal.truststore.crlLocations.2=http://caserver/crl.pem
portal.truststore.crlUpdateInterval=400
portal.truststore.crlMode=REQUIRE
portal.truststore.crlConnectionTimeout=200
portal.truststore.crlDiskCachePath=/tmp
```

- Java keystore used as a truststore:

```
portal.truststore.type=keystore
portal.truststore.keystorePath=/some/dir/truststore.jks
portal.truststore.keystoreFormat=JKS
portal.truststore.keystorePassword=xxxxxx
```

- OpenSSL truststore

```
portal.truststore.type=openssl
portal.truststore.opensslPath=/truststores/openssl
portal.truststore.opensslNsMode=EUGRIDPMA_GLOBUS_REQUIRE
portal.truststore.allowProxy=ALLOW
portal.truststore.updateInterval=1234
portal.truststore.crlMode=IF_VALID
```

Client options

The configuration file may also contain low-level options, for example if you need to specify connection timeouts, http proxies etc.

Table 3: Client options

Property name	**Type**	**Default value / mandatory**	**Description**
`portal.client.digitalSigningEnabled`	[true, false]	`true`	Controls whether signing of key web service requests should be performed.
`portal.client.httpAuthnEnabled`	[true, false]	`false`	Whether HTTP basic authentication should be used.
`portal.client.httpPassword`	string	*empty string*	Password for use with HTTP basic authentication (if enabled).
`portal.client.httpUser`	string	*empty string*	Username for use with HTTP basic authentication (if enabled).
`portal.client.inHandlers`	string	*empty string*	Space separated list of additional handler class names for handling incoming WS messages

Table 3: (continued)

Property name	Type	Default value / mandatory	Description
`portal.client.maxWsCallRetries`	integer number	`3`	Controls how many times the client should try to call a failing web service. Note that only the transient failure reasons cause the retry. Note that value of 0 enables unlimited number of retries, while value of 1 means that only one call is tried.
`portal.client.messageLogging`	[true, false]	`false`	Controls whether messages should be logged (at INFO level).
`portal.client.outHandlers`	string	*empty string*	Space separated list of additional handler class names for handling outgoing WS messages
`portal.client.securitySessions`	[true, false]	`true`	Controls whether security sessions should be enabled.
`portal.client.serverHostnameChecking`	[NONE, WARN, FAIL]	`WARN`	Controls whether server's hostname should be checked for matching its certificate subject. This verification prevents man-in-the-middle attacks. If enabled WARN will only print warning in log, FAIL will close the connection.
`portal.client.sslAuthnEnabled`	[true, false]	`true`	Controls whether SSL authentication of the client should be performed.
`portal.client.sslEnabled`	[true, false]	`true`	Controls whether the SSL/TLS connection mode is enabled.
`portal.client.wsCallRetryDelay`	integer number	`10000`	Amount of milliseconds to wait before retry of a failed web service call.
--- *HTTP client settings* ---			
`portal.client.http.allow-chunking`	[true, false]	`true`	If set to false, then the client will not use HTTP 1.1 data chunking.

Table 3: (continued)

Property name	**Type**	**Default value / mandatory**	**Description**
`portal.client.http.connection-close`	[true, false]	`false`	If set to true then the client will send connection close header, so the server will close the socket.
`portal.client.http.connection.timeout`	integer number	`20000`	Timeout for the connection establishing (ms)
`portal.client.http.maxPerRoute`	integer number	`6`	How many connections per host can be made. Note: this is a limit for a single client object instance.
`portal.client.http.maxRedirects`	integer number	`3`	Maximum number of allowed HTTP redirects.
`portal.client.http.maxTotal`	integer number	`20`	How many connections in total can be made. Note: this is a limit for a single client object instance.
`portal.client.http.socket.timeout`	integer number	`0`	Socket timeout (ms)
--- *HTTP proxy settings* ---			
`portal.client.http.nonProxyHosts`	string	-	Space (single) separated list of hosts, for which the HTTP proxy should not be used.
`portal.client.http.proxy.password`	string	-	Relevant only when using HTTP proxy: defines password for authentication to the proxy.
`portal.client.http.proxy.user`	string	-	Relevant only when using HTTP proxy: defines username for authentication to the proxy.
`portal.client.http.proxyHost`	string	-	If set then the HTTP proxy will be used, with this hostname.

Table 3: (continued)

Property name	Type	Default value / mandatory	Description
`portal.client.http.proxyPort`	integer number	-	HTTP proxy port. If not defined then system property is consulted, and as a final fallback 80 is used.
`portal.client.http.proxyType`	string	`HTTP`	HTTP proxy type: HTTP or SOCKS.

Table 4: HTTP options for the Portal

Property name	Description
`portal.client.http.proxyHost`	HTTP(s) proxy to use
`portal.client.http.proxyPort`	Port of the HTTP(s) proxy to use
`portal.client.http.nonProxyHosts`	Space separated list of host name fragments for which NOT to go via the proxy. If the target URL contains such a fragment, it is accessed directly
`portal.client.http.connection.timeout`	Timeout to use when establishing a HTTP connection
`portal.client.http.socket.timeout`	Timeout to use when reading/writing from/to HTTP connection

For example, to set the timeout when establishing a connection to 5 seconds, you would use

```
portal.client.http.connection.timeout=5000
```

Portal configuration - authentication

The web portal offers various possibilities for authentication and registration of users. Currently implemented are:

- a demo account for testing
- authentication with a user certificate that has been imported in the browser
- authentication via kerberos
- username/password
- authentication via Unity

Portal authentication and registration options

Table 5: Enabled authentication and registration types

Property name	Type	Default value / mandatory	Description
`portal.authn.enabledFacilities`	string	*mandatory to be set*	List of the enabled authentication facilities names. For example: DEMO, TLS, AUTH-USER, KRB5 or AUTH-SAML but it can be any other string (please see the note below).
`portal.registration.enabledFacilities`	string	*mandatory to be set*	List of the enabled registration facilities names. For example: REG-USER, REG-TLS, REG-SAML but it can be any other string (please see the note below).

For example to enable the possibility for the user to login with username/password, as well as with their certificate, imported in the browser, and a demo login for testing, the following line has to be configured

```
portal.authn.enabledFacilities=AUTH-USER TLS DEMO
```

To enable registration with username/password as well as register certificates, imported in the browser, you need to enable the following

```
portal.registration.enabledFacilities=REG-USER REG-TLS
```

Table 6: Common properties for all authentications

Property name	**Type**	**Default value / mandatory**	**Description**
`portal.authn.facility.<type>.description[.*]`	string *can have subkeys*	*mandatory to be set*	Description of the authenticator, to be presented in the login screen.
`portal.authn.facility.<type>.description.[.*]`	string *can have subkeys*	-	Under this prefix could be specified language specific descriptions of the authenticator.
`portal.authn.facility.<type>.name[.*]`	string *can have subkeys*	*mandatory to be set*	Human readable name of the authenticator, to be presented in the login screen. Should be unique among all authenticators.
`portal.authn.facility.<type>.name.[.*]`	string *can have subkeys*	-	Under this prefix could be specified language specific names of the authenticator.
`portal.authn.facility.<type>.type`	string	*mandatory to be set*	Authentication method to be used. Typically one of: DEMO, TLS, username, KRB5 or SAML.

The table represents a few properties that are common for all types of authentication where <*type*> is the string corresponding to one of the values in the enabled statement's right hand side. Please note that

Note
In the statement portal.*.enabledFacilities=<*enabled_type*> (where * is any of "authn" and "registration") the value of <enabled_type> can be any string. However it is important that the value of <enabled_type> from the *property portal.*.enabledFacilities = <enabled_type>* is equal to the <enabled_type> in *portal.authn.facility.<enabled_type>.*=...*

Example for username/password registration

```
portal.authn.enabledFacilities=USER

portal.authn.facility.USER.type=username
portal.authn.facility.USER.name=Username and password login
portal.authn.facility.USER.name.de=Name und Kennwort Anmeldung
portal.authn.facility.USER.description=Login using your username  ↩
    and password
portal.authn.facility.USER.description.de=Anmeldung über  ↩
    Benutzername und Kennwort
```

Table 7: DEMO authentication properties

Property name	**Type**	**Default value / mandatory**	**Description**
`portal.authn.facility.DEMO.credential.[.*]`	string *can have subkeys*	-	Under this prefix one can use standard UNICORE credential configuration. This credential will be used as demo user's grid identity. Note: this option is used only if the delegation was not set.
`portal.authn.facility.DEMO.delegation`	filesystem path	-	Path to a file with trust delegation directed at the portal. It will be used to bootstrap grid security of the demo user, i.e. demo user will act as the user who issued the delegation.
`portal.authn.facility.DEMO.description[.*]`	string *can have subkeys*	*mandatory to be set*	Description of the authenticator, to be presented in the login screen.
`portal.authn.facility.DEMO.description.[.*]`	string *can have subkeys*	-	Under this prefix could be specified language specific descriptions of the authenticator.
`portal.authn.facility.DEMO.name`	string	`Demo user`	Human friendly name of the demo user.
`portal.authn.facility.DEMO.name.[.*]`	string *can have subkeys*	-	Under this prefix could be specified language specific names of the authenticator.
`portal.authn.facility.DEMO.noLoginRequired`	[true, false]	`false`	If enabled then users are not required to enter login and password.
`portal.authn.facility.DEMO.password`	string	`demo`	Password of the demo user.

Table 7: (continued)

Property name	**Type**	**Default value / mandatory**	**Description**
`portal.authn.facility.DEMO.type`	string	*mandatory to be set*	Authentication method to be used. Typically one of: DEMO, TLS, username, KRB5 or SAML.
`portal.authn.facility.DEMO.user`	string	`demo`	Username of the demo user.

Table 8: TLS authentication properties

Property name	**Type**	**Default value / mandatory**	**Description**
`portal.authn.facility.TLS.autoRegister`	[true, false]	`false`	If true then users authenticated with a certificate issued by a trusted CA will be automatically registered locally.
`portal.authn.facility.TLS.description[.*]`	string *can have subkeys*	*mandatory to be set*	Description of the authenticator, to be presented in the login screen.
`portal.authn.facility.TLS.description.[.*]`	string *can have subkeys*	-	Under this prefix could be specified language specific descriptions of the authenticator.
`portal.authn.facility.TLS.name[.*]`	string *can have subkeys*	*mandatory to be set*	Human readable name of the authenticator, to be presented in the login screen. Should be unique among all authenticators.
`portal.authn.facility.TLS.name.[.*]`	string *can have subkeys*	-	Under this prefix could be specified language specific names of the authenticator.

Table 8: (continued)

Property name	Type	Default value / mandatory	Description
`portal.authn.facility.TLS.type`	string	*mandatory to be set*	Authentication method to be used. Typically one of: DEMO, TLS, username, KRB5 or SAML.

Table 9: Kerberos authentication properties

Property name	Type	Default value / mandatory	Description
`portal.authn.facility.KRB5.credentialProvider`	Class extending eu.unicore.portal.authn.krb5.CredentialProvider	*not set*	Java class which provides a X.509 credential for the authenticated user.
`portal.authn.facility.KRB5.delegationValidity`	integer number	`10`	Validity time in days of the delegation SAML generated for the portal.
`portal.authn.facility.KRB5.description[.*]`	string *can have subkeys*	*mandatory to be set*	Description of the authenticator, to be presented in the login screen.
`portal.authn.facility.KRB5.description.[.*]`	string *can have subkeys*	-	Under this prefix could be specified language specific descriptions of the authenticator.
`portal.authn.facility.KRB5.krb5Conf`	string	`/etc/krb5.conf`	Path to the Kerberos configuration file.
`portal.authn.facility.KRB5.name[.*]`	string *can have subkeys*	*mandatory to be set*	Human readable name of the authenticator, to be presented in the login screen. Should be unique among all authenticators.
`portal.authn.facility.KRB5.name.[.*]`	string *can have subkeys*	-	Under this prefix could be specified language specific names of the authenticator.

Table 9: (continued)

Property name	**Type**	**Default value / mandatory**	**Description**
`portal.authn.facility.KRB5.type`	string	*mandatory to be set*	Authentication method to be used. Typically one of: DEMO, TLS, username, KRB5 or SAML.

Using Unity

If your Grid installation is using the Unity identity management service (see http://www.unity-idm.eu), you can setup the configuration file with the help of the following properties.

Table 10: SAML properties

Property name	**Type**	**Default value / mandatory**	**Description**
`portal.authn.facility.AUTH-SAML.autoRegister`	[true, false]	`false`	If true then remotely authenticated users will be automatically registered locally. Note that in such case it is important to ensure that the IdP provides required attributes.
`portal.authn.facility.AUTH-SAML.description[.*]`	string *can have subkeys*	*mandatory to be set*	Description of the authenticator, to be presented in the login screen.
`portal.authn.facility.AUTH-SAML.description.[.*]`	string *can have subkeys*	-	Description of the authenticator, to be presented in the login screen.
`portal.authn.facility.AUTH-SAML.emailAttribute`	string	`email`	Name of the SAML attribute with the user's e-mail.

Table 10: (continued)

Property name	**Type**	**Default value / mandatory**	**Description**
`portal.authn.facility.AUTH-SAML.idpLogoutUrl`	string	-	If defined then the value will be used as a URL of a SAML HTTP logout endpoint of the IDP. For Unity the URL is https://HOST:PORT/-UNICORE-WEB-ENDPOINT/SLO-WEB. If the value is left undefinedthen single logout functionality won't be used. Note that for integrating SLO with Unity, the Unity unicore endpoint configuration must include in the UNICORE portal SP definition also portal's `certificate` and both `postLogoutResponseEndpoint` and `postLogoutResponseEndpoint` (both set to the base portal URL). Minimum version of Unity is 1.8.0.
`portal.authn.facility.AUTH-SAML.idpName`	string	*mandatory to be set*	Short, human readable name of the Identity Provider.

Table 10: (continued)

Property name	**Type**	**Default value / mandatory**	**Description**
`portal.authn.facility.AUTH-SAML.idpTruststore.[.*]`	string *can have subkeys*	-	Under this prefix truststore should be configured using standard UNICORE truststore settings. The truststore must contain ONLY certificates of trusted Identity Providers and NO other certificates, in particular NO CA certificates. Typically the truststore will contain only one certificate of the IdP in question, but can also conatin other IdPs certificates.
`portal.authn.facility.AUTH-SAML.idpUrl`	string	*mandatory to be set*	Full URL of the SAML Identity Provider to be used.
`portal.authn.facility.AUTH-SAML.localSamlId`	string	-	Full identifier of this SAML Service Provider, in SAML Entity format (typically a URI). It is used to identify this service to the Identity Provider. Identity Provider checks if this identifier is matching its configuration. When using UNICORE aware IdP this property will be automatically set to portal's certificate DN. In other cases it must be set.
`portal.authn.facility.AUTH-SAML.name[.*]`	string *can have subkeys*	*mandatory to be set*	Human readable name of the authenticator, to be presented in the login screen. Should be unique among all authenticators.
`portal.authn.facility.AUTH-SAML.name.[.*]`	string *can have subkeys*	-	Human readable name of the authenticator, to be presented in the login screen. Should be unique among all authenticators.

Table 10: (continued)

Property name	Type	Default value / mandatory	Description
`portal.authn.facility.AUTH-SAML.nameAttribute`	string	`cn`	Name of the SAML attribute with the user's name.
`portal.authn.facility.AUTH-SAML.organizationAttribute`	string	`o`	Name of the SAML attribute with the user's organization.
`portal.authn.facility.AUTH-SAML.photoAttribute`	string	`jpegPhoto`	Name of the SAML attribute with the user's photo.
`portal.authn.facility.AUTH-SAML.type`	string	*mandatory to be set*	Authentication method to be used. Typically one of: DEMO, TLS, username, KRB5 or SAML.
`portal.authn.facility.AUTH-SAML.unicoreIdp`	[true, false]	`true`	If true, then it is assumed that Identity Provider is UNICORE aware and will provide Grid credentials. As a side effect localSamlId will be automatically set.

An example setup for Unity authentication

```
portal.authn.facility.AUTH-SAML.type=unity
portal.authn.facility.AUTH-SAML.name=Login via Unity
portal.authn.facility.AUTH-SAML.description=Login using your  ↩
    federated identity from Unity
portal.authn.facility.AUTH-SAML.idpUrl=https://<host>:<port>/ ↩
    unicore-soapidp/saml2unicoreidp-soap-oidc/AuthenticationService
portal.authn.facility.AUTH-SAML.idpName=Unity preview
portal.authn.facility.AUTH-SAML.autoRegister=true
portal.authn.facility.AUTH-SAML.idpTruststore.type=keystore
portal.authn.facility.AUTH-SAML.idpTruststore.keystorePath=<your- ↩
    keystore-location>/<idp-truststore>.jks
portal.authn.facility.AUTH-SAML.idpTruststore.keystorePassword=< ↩
    keystore-password>
```

Single logout configuration

From Unity 1.8.0 on a single logout from Unity and Portal is available. To configure this:

1) In portal's config in authN section of unity add:

portal.authn.facility.AUTH-SAML.idpLogoutUrl=https://UNITY-ADDR/unicore-idp/SLO-WEB

unicore-idp must be the base path of the UNICORE **web** endpoint in Unity.

2) in Unity config (assuming that login already works) add in the web UNICORE endpoint configuration, in the section defining portal as a trusted SP:

unity.saml.acceptedSP.1.certificate=PORTAL-CERT unity.saml.acceptedSP.1.postLogoutResponseEndpoint=PORTAL-URL unity.saml.acceptedSP.1.postLogoutEndpoint=PORTAL-URL

where PORTAL-CERT is a name of a portal's certificate as defined in pki.properties and the PORTAL-URL is simply the base URL of the portal (e.g. https://host:port)

Account registration

Typically users need to be registered in the portal server in order to be able to login properly. The table represents a few properties that are common for all types of registration where <*type*> is the string corresponding to one of the enabled types of registration.

Table 11: Basic registration properties

Property name	**Type**	**Default value / mandatory**	**Description**
`portal.registration.facility.<type>.description`	string	*empty string*	Description of the registrator, to be presented in the registration screen.
`portal.registration.facility.<type>.description.[.*]`	string *can have subkeys*	-	Description of the registrator, to be presented in the login screen.
`portal.registration.facility.<type>.name`	string	*mandatory to be set*	Human readable name of the registrator, to be presented in the login screen. Should be unique among all registrators.
`portal.registration.facility.<type>.name.[.*]`	string *can have subkeys*	-	Human readable name of the registrator, to be presented in the login screen. Should be unique among all registrator.

Table 11: (continued)

Property name	**Type**	**Default value / mandatory**	**Description**
`portal.registration.facility.<type>.type`	string	*mandatory to be set*	Registration method to be used. Typically one of: DEMO, TLS, username, KRB5 or SAML.

Even when the authentication happens via Unity, the portal still needs the user to be registered. The necessity of manual registration by the user can be avoided if *portal.authn.facility.AUTH-SAML.autoRegister* is set to true. Other properties concerning the registration of Unity users include

Table 12: SAML properties for registration

Property name	**Type**	**Default value / mandatory**	**Description**
`portal.registration.facility.REG-SAML.description[.*]`	string *can have subkeys*	*mandatory to be set*	Description of the authenticator, to be presented in the login screen.
`portal.registration.facility.REG-SAML.description.[.*]`	string *can have subkeys*	-	Description of the authenticator, to be presented in the login screen.
`portal.registration.facility.REG-SAML.emailAttribute`	string	`email`	Name of the SAML attribute with the user's e-mail.
`portal.registration.facility.REG-SAML.idpName`	string	*mandatory to be set*	Short, human readable name of the Identity Provider.

Table 12: (continued)

Property name	**Type**	**Default value / mandatory**	**Description**
`portal.registration.facility.REG-SAML.idpTruststore.[.*]`	string *can have subkeys*	-	Under this prefix truststore should be configured using standard UNICORE truststore settings. The truststore must contain ONLY certificates of trusted Identity Providers and NO other certificates, in particular NO CA certificates. Typically the truststore will contain only one certificate of the IdP in question, but can also conatin other IdPs certificates.
`portal.registration.facility.REG-SAML.idpUrl`	string	*mandatory to be set*	Full URL of the SAML Identity Provider to be used.
`portal.registration.facility.REG-SAML.localSamlId`	string	-	Full identifier of this SAML Service Provider, in SAML Entity format (typically a URI). It is used to identify this service to the Identity Provider. Identity Provider checks if this identifier is matching its configuration. When using UNICORE aware IdP this property will be automatically set to portal's certificate DN. In other cases it must be set.
`portal.registration.facility.REG-SAML.name[.*]`	string *can have subkeys*	*mandatory to be set*	Human readable name of the authenticator, to be presented in the login screen. Should be unique among all authenticators.

Table 12: (continued)

Property name	Type	Default value / mandatory	Description
`portal.registration.facility.REG-SAML.name.[.*]`	string *can have subkeys*	-	Human readable name of the authenticator, to be presented in the login screen. Should be unique among all authenticators.
`portal.registration.facility.REG-SAML.nameAttribute`	string	`cn`	Name of the SAML attribute with the user's name.
`portal.registration.facility.REG-SAML.organizationAttribute`	string	`o`	Name of the SAML attribute with the user's organization.
`portal.registration.facility.REG-SAML.photoAttribute`	string	`jpegPhoto`	Name of the SAML attribute with the user's photo.
`portal.registration.facility.REG-SAML.requireAttributesFromIdp`	[true, false]	`false`	If set to true the IdP must provide at least the email and name attributes in the returned assertion. What is more the user can not edit them. If false then editing is possible and lack of attributes from IdP is not considered a problem.
`portal.registration.facility.REG-SAML.type`	string	*mandatory to be set*	Registration method to be used. Typically one of: DEMO, TLS, username, KRB5 or SAML.
`portal.registration.facility.REG-SAML.unicoreIdp`	[true, false]	`true`	If true, then it is assumed that Identity Provider is UNICORE aware and will provide Grid credentials. As a side effect localSamlId will be automatically set.

The following properties concern username-password registration fields on the UI

Table 13: Username - password properties

Property name	Type	Default value / mandatory	Description
`portal.registration.facility.REG-USER.description`	string	*empty string*	Description of the registrator, to be presented in the registration screen.
`portal.registration.facility.REG-USER.description.[.*]`	string *can have subkeys*	-	Description of the registrator, to be presented in the login screen.
`portal.registration.facility.REG-USER.minPasswordLength`	integer number	`8`	Minimum allowed password length.
`portal.registration.facility.REG-USER.name`	string	*mandatory to be set*	Human readable name of the registrator, to be presented in the login screen. Should be unique among all registrators.
`portal.registration.facility.REG-USER.name.[.*]`	string *can have subkeys*	-	Human readable name of the registrator, to be presented in the login screen. Should be unique among all registrator.
`portal.registration.facility.REG-USER.preRegisterAction`	Class extending eu.unicore.portal.authn.username.PreRegistrationHook	*not set*	Class name of preRegistrationAction which is executed before the user is added to the database.
`portal.registration.facility.REG-USER.requireSecurePassword`	[true, false]	`true`	If true then additional constraints are placed on the password: characters must belong to different classes, special, easy to guess sequences are not allowed.

Table 13: (continued)

Property name	**Type**	**Default value / mandatory**	**Description**
`portal.registration.facility.REG-USER.type`	string	*mandatory to be set*	Registration method to be used. Typically one of: DEMO, TLS, username, KRB5 or SAML.

Example of setting the properties of the username/passwrod registration

```
portal.registration.facility.REG-USER.type=username
portal.registration.facility.REG-USER.name=Username registration
portal.registration.facility.REG-USER.description=Register a new  ↩
    local account, for logging with username and password
portal.registration.facility.REG-USER.minPasswordLength=6
portal.registration.facility.REG-USER.requireSecurePassword=false
```

Portal configuration - server options, preferences and UI

Server options

Table 14: Portal server properties

Property name	**Type**	**Default value / mandatory**	**Description**
`portal.server.address.<NUMBER>`	list of properties with a common prefix	*mandatory to be set*	URLs to bind to. Both http and https can be used.
`portal.server.resourceBase`	string	`.`	Web application resources base path.
`portal.server.webconfigPath`	string	`conf/web.xml`	Web application config file (web.xml) path.

User preferences

The following table represents how to setup a connection to the database where the user preferences are to be stored.

Table 15: User preferences properties

Property name	Type	Default value / mandatory	Description
		--- Database ---	
`portal.userprofiles.dialect`	[h2, mysql]	`h2`	Database SQL dialect. Must match the selected driver, however sometimes more then one driver can be available for a dialect.
`portal.userprofiles.driver`	Class extending java.sql.Driver	`org.h2.Driver`	Database driver class name. This property is optional - if not set, then a default driver for the chosen database type is used.
`portal.userprofiles.jdbcUrl`	string	`jdbc:h2:data/userprofiles`	Database JDBC URL.
`portal.userprofiles.mapconfig`	string	`FROM_CLASSPATH`	MyBatis configuration file. By default it is read from the classpath.
`portal.userprofiles.password`	string	*empty string*	Database password.
`portal.userprofiles.sql`	string	`conf/db/h2.sql`	Database setup SQL script.
`portal.userprofiles.username`	string	`sa`	Database username.

Example of setting up the database connection

```
portal.userprofiles.driver=org.h2.Driver
portal.userprofiles.jdbcUrl=jdbc:h2:<path_to_userprefs>
portal.userprofiles.username=<username>
portal.userprofiles.password=<password>
portal.userprofiles.sql=<path_to_db_creation_script>/h2.sql
```

Core and UI configuration options

Table 16: Portal configuration properties

Property name	**Type**	**Default value / mandatory**	**Description**
`portal.core.customTranslationDirectory`	filesystem path	-	If defined specifies a directory with custom translations. The directory must contain properties message bundles called `messages.properties` with per-language variants as `message_de.properties`. Those messages override default system messages. The message keys must be found in the portal source code.
`portal.core.defaultApplication`	string	-	Can specify an application, that is selected by default in the generic application creation view.

Table 16: (continued)

Property name	**Type**	**Default value / mandatory**	**Description**
`portal.core.discoveryMaxAllowedLoad`	floating point number	`3`	Load is defined as an average, cumulative number of refreshes that are supposed to be performed per second. This property defines up to what load the discovery is using the regular refresh intervals. When the normal load threshold is exceeded, the slow down mechanism is activated. The slow down mechanism increases the intervals so that the discovery stays in the given threshold. The proper value depends on your machine power, especially number of CPUs. If the service generates too large load on a machine this setting should be reduced. If machine is powerfull but discovery seems to be slow, then this setting can be increased.
`portal.core.discoveryMediumServiceRefresh`	integer number	`30000`	The regular interval in ms between semi dynamic resource (TSS or SMS under SMSF) status refreshes
`portal.core.discoveryRegistryRefresh`	integer number	`30000`	The regular interval in ms between registry status refreshes
`portal.core.discoveryServiceDeathCheckRefresh`	integer number	`60000`	The regular interval in ms between status refreshes of resources for which we can expect only its removal (e.g. finished jobs)

Table 16: (continued)

Property name	Type	Default value / mandatory	Description
`portal.core.discoveryTopServiceRefresh`	integer number	`30000`	The regular interval in ms between top service (TSF, global SMS, . . .) status refreshes
`portal.core.discoveryVolatileServiceRefresh`	integer number	`10000`	The regular interval in ms between volatile resource (jobs, workflows) status refreshes
`portal.core.enableGoogleMaps`	[true, false]	`true`	Controls whether Google maps should be included in the sites view
`portal.core.enabledLocales.<NUMBER>`	list of properties with a common prefix	-	List of locales enabled in the portal. Each entry must have a language code as *en* or *pl* first, and then, after a space an optional, short name which will be presented in the UI
`portal.core.registries*`	list of properties with a common prefix	*mandatory to be set*	List of registry URLs
`portal.core.threadPoolSize`	integer number	`20`	Maximum number of threads to be used by the portal to schedule tasks performed by users.
`portal.core.workflow.templates.enable`	[true, false]	`true`	Enable the possibility to import and submit workflow templates through the New Job view.
`portal.core.workspace.root*`	list of properties with a common prefix	*mandatory to be set*	URL to directory holding user workspaces accessible through UNICORE 6 storage management service.

An important configuration item refers to the location where user workspaces are stored. By default, they are stored on the portal machine, but they can also be stored remotely on a UNICORE storage.

Example of setting up the user workspace

```
If the workspace is located on the local file system, the prefix ↩
    file:/// needs to be used.
portal.core.workspace.root = file:///tmp/portal-workspaces

If the workspace is located remotely on a UNICORE storage, the ↩
    prefix u6:// needs to be used.
portal.grid.workspace.root = u6://<host>:<port>/<site>/services/ ↩
    StorageManagement?res=default_storage#portalWorkspaces
```

Tha table has been autogenerated. Please exchange of configuring different languages for the UI

```
portal.core.enabledLocales.1=en English
portal.core.enabledLocales.2=de Deutsch
portal.core.enabledLocales.3=pl Polski
```

Table 17: Portal configuration properties for the User Interface (UI)

Property name	**Type**	**Default value / mandatory**	**Description**
`portal.ui.menu.*.contents.<NUMBER>`	list of properties with a common prefix	-	Identifiers of menu entries of the menu in order of appearance. Identifiers are the same as those in generated HTML.
`portal.ui.menu.*.default`	string	`eu.unicore.portal.ui.views.switchTo.main`	Specify string to the default view, to which the user will be redirected after log in. The value has to be one of the views in the menu.contents.
`portal.ui.menu.*.disabled`	[true, false]	`false`	If true then the whole menu is disabled
`portal.ui.footer.<NUMBER>`	Structured list	-	Configuration of footer contents. Footer includes images which are links to a fixed URL.
`portal.ui.footer.<NUMBER>.footerDescription`	string	-	Tooltip to be used for the footer entry link.

Table 17: (continued)

Property name	**Type**	**Default value / mandatory**	**Description**
`portal.ui.footer.<NUMBER>.footerIcon`	string	*mandatory to be set*	Abstract path within the theme (../../icons/*) of the footer entry icon.
`portal.ui.footer.<NUMBER>.footerUrl`	string	*mandatory to be set*	URL to which for the footer entry link will forward on image click.
`portal.ui.header.<NUMBER>`	Structured list	-	Configuration of header logos.
`portal.ui.homepage.html*`	list of properties with a common prefix	-	Local HTML file or an URLthat builds the content on the home page of the portal.
`portal.ui.homepage.rss*`	list of properties with a common prefix	-	RSS/ATOM page that builds the content on the home page of the portal.
`portal.ui.homepage.title[.*]`	string *can have subkeys*	-	Customize a title / labelwhich you would like to appear on the home page. Can be localized in a different language.
`portal.ui.header.<NUMBER>.mainLogo`	string	-	Abstract path within the theme (../../icons/*) of the header main icon.
`portal.ui.menu.*`	Structured list	-	Configuration of main menus contents. Available menu keys are: [navigation, buttons]
`portal.ui.workspace.hide`	[true, false]	`false`	Prevent the user workspace from showing on UI.

Example of configuring menu options for the UI

```
portal.ui.menu.navigation.disabled=false
portal.ui.menu.navigation.contents.1=eu.unicore.portal.ui.views. ↩
    switchTo.main
portal.ui.menu.navigation.contents.2=eu.unicore.portal.ui.views. ↩
    switchTo.app
portal.ui.menu.navigation.contents.3=eu.unicore.portal.ui.views. ↩
    switchTo.jobs
```