



UVOS WEB REGISTRATION EXTENSION MANUAL

UNICORE Team

Document Version:	1.5.1
Component Version:	1.5.0-p6
Date:	13 04 2012

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

This work was co-funded by the EC Chemomentum project under the FP6 Grant Agreement Nr. IST-033437.



Contents

1	Introduction	1
2	Compatibility	1
3	Installation	1
3.1	Installation from RPM package (RedHat distributions)	2
3.2	Installation from the DEB package (Debian distributions)	2
3.3	Installation from the archive	2
4	Web browser path	3
5	Reference application form	3

UNICORE VO Service (UVOS) is a client-server system, developed to be used as an additional tool for large distributed systems, providing a solution for grid users management. Grid systems, especially UNICORE grid middleware, are the mainspring of the UVOS system. UVOS can be used with different systems, however is designed primarily to support UNICORE grid middleware.

For more information about UVOS visit <http://uvos.chemomentum.org>.

1 Introduction

VO application forms web UI for the UVOS system provides an on-line registration form for users applying for VO membership. For a general overview of the UVOS system please review UVOS server documentation.

The main features of this module are:

- a customizable interface for submitting VO registration requests,
- customization allows for choosing agreement form, requested attributes, base VO, request additional groups, provide contact details and provide identity in a supported format,
- multiple application forms might be used,
- updates of forms do not require server's restart.
- it is possible to connect this module with PnP CA, so user can automatically request a VO membership and a certificate.

2 Compatibility

This release is compatible with UVOS 1.4.0 and 1.4.1 (future versions might be supported too).

3 Installation

UVOS web application extension is distributed either as a platform independent archive or as an installable, platform dependent package such as RPM. Depending on the installation source used, installation method and paths after installation are different.

3.1 Installation from RPM package (RedHat distributions)

The preferred way is to use Yum to install (and subsequently update) UVOS webapp.

To perform the Yum installation, EMI Yum repository must be installed first. Refer to the EMI release documentation (available at the EMI website <http://www.eu-emi.eu/releases>) for detailed instructions. Typically installation of the EMI repository requires to download a single RPM file and install it.

After the EMI repository is configured, the following command installs UVOS webapp:

```
$> yum install unicore-uvos-webapp
```

The installed extension will be placed in the directory `/var/lib/unicore/uvos-server/webapps/`

You have to restart UVOS server to complete the installation.

3.2 Installation from the DEB package (Debian distributions)

The preferred installation way is to use apt to install and subsequently update UVOS webapp.

To perform the apt installation, EMI apt repository must be installed first. Refer to the EMI release documentation (available at the EMI website <http://www.eu-emi.eu/releases>) for detailed instructions. Typically installation of the EMI repository requires to download a single DEB file and install it.

After the EMI repository is configured, the following command installs UVOS webapp:

```
$> apt-get install unicore-uvos-webapp
```

The installed extension will be placed in the directory `/var/lib/unicore/uvos-server/webapps/`

You have to restart UVOS server to complete the installation.

3.3 Installation from the archive

If installing using a portable archive you have:

1. Download the archive from the UNICORE download site and unpack it.
2. In the unpacked archive you can find a file with a `.war` extension. Copy it to the `webapps/` directory of UVOS server. This directory is placed directly under installation root folder which was used when installing UVOS server from portable archive.

You have to restart UVOS server to complete the installation.

4 Web browser path

The name of the war file is important as it is reflected in the URL by which web application is available. By default it is uvos-webapp-VERSION.war, where VERSION is e.g. "1.2". You can freely change the name of the war file. From now we will use label WAR_FILE_NAME to refer to actual base name of your war file.

You should be able to reach the web UI indirectly (via redirect) at the address:

```
<UVOS server base address>/WAR_FILE_NAME
```

or directly at:

```
<UVOS server base address>/WAR_FILE_NAME/VOapplications.do?_flowId= ↵  
application-flow
```

Example for the default server configuration and release 1.0:

```
https://localhost:2443/uvos-webapp-1.0
```

Example application form is present in this package. See its comments for the information how to set it up. Use the UVOS command line client (UVOS CLC) to add/update/delete forms. You can use both UVOS CLC and VOManager to manage applications (accept/reject/delete/update).

VO application form changes are detected automatically by this component so you don't need to restart the UVOS server.

Since the version 1.3.1 this module can automatically send user's CSR to the new PnP CA (see <http://pnp-ca.sourceforge.net>).

5 Reference application form

The `referenceApplicationForm.xml` file which can be found in this extension distribution provides a complete and up to date example of what can be configured in the application form. Its contents is also presented below.

```
<?xml version="1.0" encoding="UTF-8"?>  
<tns:VOApplicationForm  
    xmlns:tns="urn:icm:chemomentum:uvos:internalWSapi: ↵  
    application"  
    xmlns:uvos="urn:icm:chemomentum:uvos:internalWSapi">  
  
<!--  
    This is a full example of an application form definition.  
    Use it for reference, when creating your own form.  
-->
```

```
<!-- Mandatory: Form name, will be displayed at the WWW interface ←
-->
<tns:friendlyName>Example Form</tns:friendlyName>

<!-- Mandatory: Form description, will be displayed together with ←
the name -->
<tns:description> Example form description</tns:description>

<!-- Mandatory: Form id. It is used to match original form when ←
updating an existing form.
Otherwise (when creating a new form) it is ignored and so can be ←
negative number -->
<tns:id>-1</tns:id>

<!-- Mandatory: Agreement. User will have to accept this. This ←
element is mandatory,
however you can leave the contents empty; then agreement won't ←
be shown.-->
<tns:agreement>Do you agree to make Bars and are you a real Foo?
<br>
VO service will store your personal identity data and will expose ←
it to other parties
what will be subject to VO policy.
</tns:agreement>

<!-- Mandatory: Base group. User will apply for membership in ←
this group, and if the form is
approved by an admin, the user will be added to this group. -->
<tns:baseGroup>
  <uvos:path>VO</uvos:path>
  <uvos:path>subgroup</uvos:path>
</tns:baseGroup>

<!-- Mandatory: Additional constraints -->
<tns:constraints>

<!-- Allowed format of user identity. User will have to choose ←
one among them
(if more then one is allowed) and then provide data in suitable ←
format. Valid formats are:

urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName
urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress
urn:unicore:idType:X509Certificate

If this constraint is absent then all formats are allowed.
-->
  <tns:allowedIdFormats>urn:oasis:names:tc:SAML:1.1:nameid- ←
format:X509SubjectName</tns:allowedIdFormats>
```

```
<tns:allowedIdFormats>urn:oasis:names:tc:SAML:1.1:nameid- ↵
    format:emailAddress</tns:allowedIdFormats>

<!-- [Currently not implemented!]
    Mandatory: Whether the user is allowed to provide ↵
        additional custom attributes
    (or more precisely propose some attributes) -->
<tns:customAttributes>true</tns:customAttributes>

<!-- List of additional attributes (possibly scoped) with ( ↵
    optional) values. Name is required,
    scope and value(s) not. Value must be base64 encoded.-->
<tns:attributesSelection>
    <uvos:name>urn:authz:intervo:vo</uvos:name>
    <uvos:scope>/Math-VO/Scientists</uvos:scope>
    <uvos:value>aWRlbnRpdHlDdGw=</uvos:value>
</tns:attributesSelection>
    <tns:attributesSelection>
    <uvos:name>favouriteNumbers</uvos:name>
</tns:attributesSelection>

<!-- [Currently not implemented!]
    Mandatory: Whether the user is allowed to provide ↵
        additional custom groups
    (or more precisely propose them) -->
<tns:customSubgroups>true</tns:customSubgroups>

<!-- List of additional groups. User will be allowed to apply ↵
    for membership in those groups
    (additionally to the membership in the base group).-->
<tns:subgroupsSelection>
    <uvos:path>Math-VO</uvos:path>
    <uvos:path>Staff</uvos:path>
</tns:subgroupsSelection>
<tns:subgroupsSelection>
    <uvos:path>Math-VO</uvos:path>
    <uvos:path>Scientists</uvos:path>
</tns:subgroupsSelection>

<!-- By including this optional section user will be ↵
    allowed to submit CSR. The CSR will be send to
    the PnP CA defined inside. Also application request will be ↵
    stored in UVOS db however to accept it,
    administrator will need to update it with an issued ↵
    certificate first.
    To make it working user will need a possibility to choose ↵
    X509Certificate
    or a DN identity type (so make it available above). -->
<tns:allowCSR>
    <!-- Not really used right now -->
```

```
<tns:friendlyName>Example PnP CA</tns:friendlyName>
<!-- Address of the CA. In case of using a old PnP CA ↵
      installation (as can be downloaded
      from the UNICORE SF project) provide a raw host:port ↵
      address only (and nothing more).

      If using a newr PnP CA (from the http://sourceforge.net/ ↵
      projects/pnp-ca) provide a full URL.
      It looks like this:
      https://pnpca.example.com:8443/services/ ↵
      CertificationAuthorityFacade
      Also don't forget to provide newPnPCCA marker below.
-->
      <tns:address>pnpca.example.com:8443</tns:address>

      <!-- Optional: used to mark that new PnP CA (with ↵
      web service interface) is being used.
      Defaults to false. -->
      <tns:newPnPCCA>true</tns:newPnPCCA>
</tns:allowCSR>

<!-- Optional email address of the vo form administrator. ↵
      It will receive an email when the
      form was filled. -->
<tns:notifyAddress>root@localhost</tns:notifyAddress>

<!-- Optional (default is false): if set to true and there are ↵
      defined any additional subgroups
      via 'subgroupsSelection' element then those groups are always ↵
      applied
      and are not presented in the application form. -->
<tns:forceGroupSelection>>false</tns:forceGroupSelection>

<!-- Optional (default is false): if set to true and there are ↵
      defined any additional attributes
      via 'attributesSelection' element then those attributes are ↵
      always applied
      and are not presented in the application form. -->
<tns:forceAttributeSelection>>false</tns:forceAttributeSelection ↵
>

<!-- Optional (default is true): when set to true then ↵
      possibility to submit additional CA
      certificate is disabled. -->
<tns:disableUserCASubmission>true</tns:disableUserCASubmission>

<!-- Optional (default is false): if set to false then ↵
      certificate submitted as identity
      not checked. Otherwise (what is the default behavior) the ↵
      certificate is checked against
```

```
the server's truststore, CRLs and for overall validity. -->
<tns:disableCertificateValidation>>false</tns: ←
    disableCertificateValidation>

</tns:constraints>
</tns:VOApplicationForm>
```