ICT 269978

Integrated Project of the 7th Framework Programme

COOPERATION, THEME 3
Information & Communication Technologies
ICT-2009.5.3, Virtual Physiological Human

VPH-Share

Work Package: WP6

User Access Systems

Deliverable: D6.5

Production Deployment of User Access Systems

Version: 1v2

Date: 28-Feb-14

DOCUMENT INFORMATION

IST Project Num	FP7 – ICT - 269978	**Acronym**	VPH-Share
Full title	Virtual Physiological Human: Sharing for Healthcare – A Research Environment		
Project URL	http://www.vph-share.eu		
EU Project officer	Robert Begier		

Work package	**Number**	6	**Title**	User Access Systems
Deliverable	**Number**	6.5	**Title**	Production Deployment of User Access Systems

Date of delivery	**Contractual**	28-Feb-14	**Actual**	28-Feb-14
Status	Version 1v2		Final ☒	
Nature	Prototype ☒ Report ☐ Dissemination ☐ Other ☐			
Dissemination Level	Public (PU) ☒ Consortium (CO) ☐	Restricted to other Programme Participants (PP) ☐ Restricted to specified group (RE) ☐		

Authors (Partner)	Debora Testi (CINECA), Daniel Harezlak (CYF), Ernesto Coto (USFD), Juan Arenas (USFD). Vadim Surpin (IITP)			
Responsible Author	Juan Arenas		**Email**	j.arenas@sheffield.ac.uk
	Partner	USFD	**Phone**	+44 (0) 114 222 0166

Abstract (for dissemination)	This document details the Production Deployment of User Access Systems from the technical and end-users point of view. It lists the status of each component produced by WP6 and the on-going work for each of them.
Keywords	Appliances for visualisation of physiological data, scientific workflow composition/management, user interface, usability, semantic search, annotation, roles policies, webservice, taverna, plugin, security, batch processing, web composition

Version Log			
Issue Date	**Version**	**Author**	**Change**
4-Oct-13	0.1	EC, JA	Initial draft for comments
9-Oct-13	0.2	EC, JA	Initial draft after changes suggested by DT
11-Dec-13	0.3	EC, JA	Initial draft of Chapter 5
10-Jan-14	0.4	JA, DT	Include MI initial content, Chapter 3
13-Jan-14	0.5	DH	Include Cloud Services content (chapter 4)
23-Jan-14	0.6	JA, EC	First consolidate version, for review and comments from all WP members
27-Jan-14	0.7	DH	Update chapter 4
31-Jan-14	0.8	DT	Revision of MI sections
7-Feb-14	1.0	JA, EC	Second consolidate version, first for internal review. Minor changes pending.
21-Feb-14	1.0	ES	Comments from the internal reviewer, Enrico Schileo
21-Feb-14	1.0	EC,DH,DT	Comments addressed and final version from co-authors and partners.
26-Feb-14	1.1	JA, EC	Final review before submission to the PMO
28-Feb-14	1.2	PMO	Submission Version

CONTENTS

LIST OF FIGURES

LIST OF TABLES

EXECUTIVE SUMMARY

This document presents the production version of the User Access System that has been developed by the WP6 in the context of the VPH-Share project. WP6 is an integration work package and therefore the multiple interactions with the other project work packages are also reflected in this document.

The main objective of this deliverable is to present the different features that have been developed along the project and that are going to be released as part of the production version of the user access system at the end of year 3. In addition, the document outlines the future plans to support external users and use cases.

The following topics are addressed in this document:

- An overview of the final features released via the project dedicated web portal (Master Interface or MI) and their APIs, providing highlights about:
  - Final Architecture and API
    - Web application
    - Security and authentication
  - User interface
    - User registration and access
    - Discovery tools
    - Resources management and their operation
      - Data
      - Applications
      - Workflows
      - Owned resources
  - User care
    - Use cases support
    - External projects support
    - External users support

- An overview of the final VPH-Share Cloud services on which the MI relays for the iteration with the backend infostructure
  - Final Architecture and API
  - Overview of Cloud Management User Interface (it is covered in more detail in WP2's deliverable)
    - Development Mode
    - Generic Invoker
    - External Workflows
  - LOBCDER repository
  - Remote Desktop Access
  - Webservice Catalogue

- An overview of the workflow services and management, including:
  - Final Architecture

- GIMAS WebServices plugin to make command line tools available as web services
- Specification of services requiring user interaction
- VPH-Share plugin
  - Desktop composition and execution tool
  - Web composition and execution through Taverna Online
  - Support for workflows with long execution times
- Workflow execution through the MI
- Batch execution
  - Desktop batch execution
  - Web batch execution
- Workflow Manager API
- Data Provenance and Semantic
- Workflow service monitoring

- A high level overview of the goals and work plan to be accomplished during year 4.

This document is organised as follows:

- Chapter 3 describes the Master Interface with an overview of the functionalities that have been released on the production version of the User Access System.

- Chapter 4 describes the progress on VPH-Share cloud infrastructure that is under the responsibility of WP6 but in close interaction with WP2.

- Chapter 5 describes the progress on the tools that have to facilitate the workflow integration (workflow composition and execution).

- Chapter 6 provides a high-level overview of work accomplishments along year 3.

- Chapter 7 provides a high-level overview of work plan for year 4.

1 INTRODUCTION

The User Access System is the main entry point for researchers and clinicians willing to access the VPH-Share infrastructure services. As such it should contain tools that enrich the user's experience and facilitate the interaction with the platform, exposing in an intuitive way all the data and services that constitute the infrastructure.

According to the evolution of the project and the consequent level of maturity of the facilities provided by the other technical work packages, a series of three incremental prototypes of the User Access System is being developed, as defined in the project. These incremental releases are allowing continuous feedback to be obtained from the users at an early stage of development and to consider usability as one of the main drivers for the development of the system. Furthermore, in accordance to this attention towards end-users (and in close collaboration with WP8), after the release of each prototype users' feedback report is issued in conjunction with WP8 and circulated among the technical partners of the project for inclusion in the requirements for the next phase of software development.

In this document, we provide a detailed description of the main tools and functionalities that constitute the production prototype of the User Access System, describing its status and the current plans for its future development. In particular, in this deployment we have included the main facilities that permit the user to start using the infrastructure and implement basic use case scenarios.

According to this, during this period we have improved services and functionalities that were already available as well as released new ones; all of them are described in the following chapters.

2 THE MASTER INTERFACE

As already mentioned the Master Interface is the entry point for the user into the VPH-Share infostructure and it can be reached at portal.vph-share.eu. In particular, it is a web application, which besides providing its own functionalities, for example for user management and permission control, it integrates and exposes the user interface for the services provided by the other technical WPs.

2.1 Final Architecture and API

The scheme below (Figure 1) represents how the Master Interface is connected to other WPs or external services.

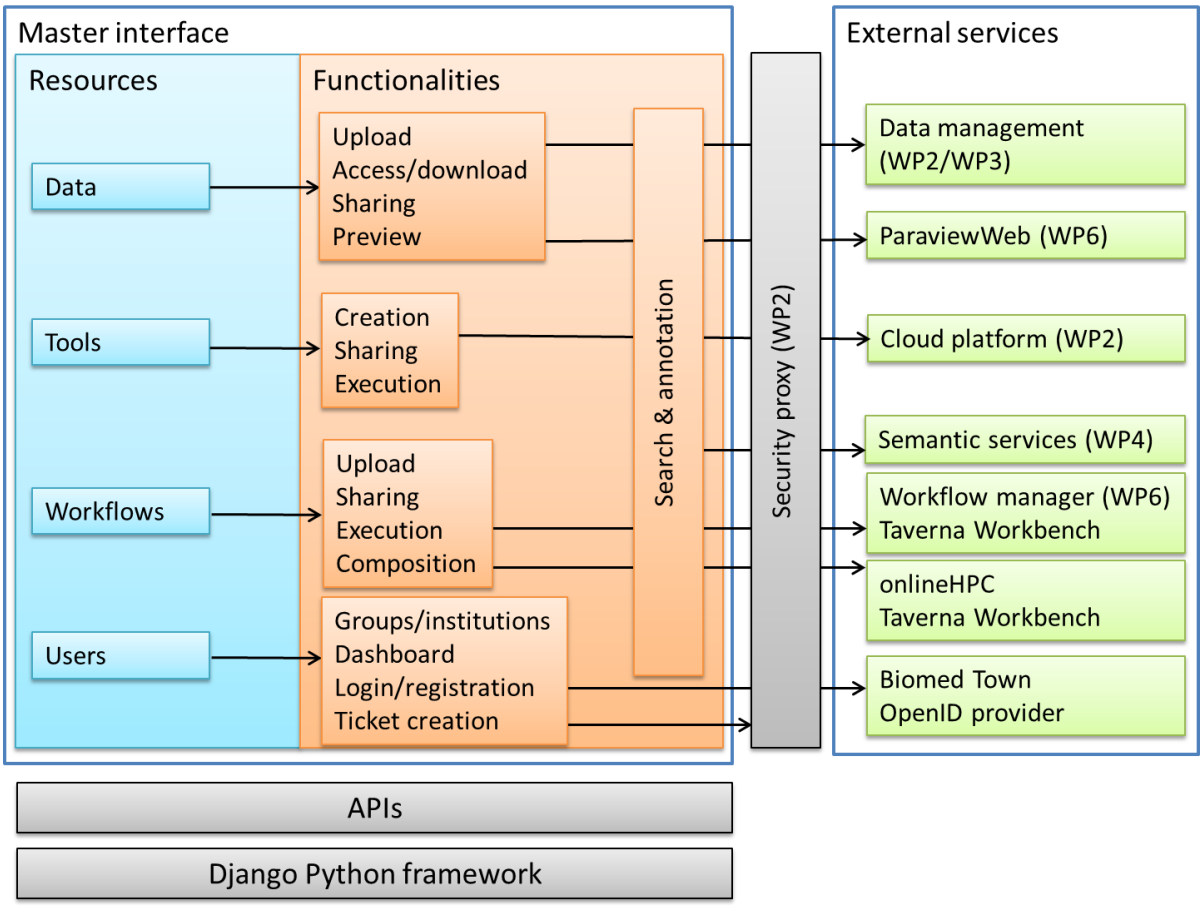


Figure 1. Master Interface component diagram

The Master Interface is implemented as reported in the next section. Some APIs to integrate or connect custom components to the MI are listed into the portal help pages and reported also here for completeness.

Any updates to the API will be documented at https://portal.vph-share.eu/help/api/.

2.2 API description

2.2.1 *Authentication Ticket:*

2.2.1.1 */api/validatetkt (also /validatetkt works)*

Description It validates given authentication ticket; it returns a **JSON** representation of the user attributes, and it raises **403** if ticket is invalid.

Url structure `portal.vph-share.eu/api/validatetkt?ticket=<ticket>`

Method GET

Parameters **ticket**
your authentication ticket

Returns **Status code 200**

```
{
    "username": "username",
    "language": "",
    "country": "ITALY",
    "role": [
        "developer",
        "friend",
    ],
    "postcode": "40033",
    "fullname": "Master Interface",
    "email": "mail@example.com"
}
```

Status code 403 Ticket not valid
Status code 500 Any other kind of error

2.2.2 Groups Management

2.2.2.1 */api/searchuser*

Description It searches for users. The provided term is searched in users' email, first name and last name. The search is case insensitive. Without any term, the full list of users is returned. It is available to all authenticated users. A **JSON** list of user attributes is returned.

Url structure `portal.vph-share.eu/api/searchuser?term=<term>&ticket=<ticket>`

Method GET

Parameters **term**
the search term **optional**
ticket
your authentication ticket

Returns

Status code 200

```
[
    {"username": "mbalasso", "email": "email@email.com",
 "fullname": "Matteo Balasso"},
    {"username": "testuser", "email": "email@email.com",
 "fullname": "Test User"}
]
```

Status code 403 Ticket not valid
Status code 500

2.2.2.2 /api/searchgroup

Description It searches for groups. The search is case insensitive. Without any term, the full list of groups is returned. A **JSON** list of group names is returned. It is available to all authenticated users.

Url structure `portal.vph-share.eu/api/searchgroup?term=<term>&ticket=<ticket>`

Method GET

Parameters

Term
the search term **optional**
ticket
your authentication ticket

Returns

Status code 200

```
[
    {
        "groupname": "testgroup0001",
        "subscribers": 1
    },
    {
        "groupname": "testgroup0002",
        "subscribers": 3
    }
]
```

Status code 403 Ticket not valid
Status code 500

2.2.2.3 /api/creategroup

Description It creates a new group (available only to staff users). With the parent parameter provided, the group is created as child of the given group.

Url structure `portal.vph-share.eu/api/creategroup?group=<group>&parent=<parent>&ticket=<ticket>`

Method GET

Parameters
group
the group name (must be globally unique)
parent
the parent group name optional
ticket
your authentication ticket

Returns
Status code 200
"OK"
Status code 403 Ticket not valid or user not allowed to invoke the service
Status code 500

2.2.2.4 /api/deletegroup

Description It deletes an existing group (available only to staff users). Even after deletion, the group name will not be available for new groups.

Url structure
portal.vph-share.eu/api/deletegroup?group=<group>&ticket=<ticket>

Method GET

Parameters
group
the group name
ticket
your authentication ticket

Returns
Status code 200
"OK"
Status code 403 Ticket not valid or user not allowed to invoke the service
Status code 500

2.2.2.5 /api/addtogroup

Description It adds a user or a group to a group. The requester must be one of the group managers (the same user who has created the group). With the *recursive* parameter, if the group has any child group, the user will be added to all of them as well.

Url structure
portal.vph-share.eu/api/addtogroup?group=<group>&name=<name_to_add>&ticket=<ticket>

Method GET

Parameters
group
the group name
name
the user name or the group name to add
recursive
if present with any non-empty value, add user to all children group optional
ticket
your authentication ticket

Returns
Status code 200
"OK"
Status code 403 Ticket not valid or user not allowed to invoke the service
Status code 500

2.2.2.6 /api/removeuser

Description It removes a user from a group. The requester must be one of the group managers (the same user who has created the group). With the *recursive* parameter, if the group has any child group, the user will be removed from all of them as well.

Url structure
`portal.vph-share.eu/api/removeuser?group=<group>&username=<username>&ticket=<ticket>`

Method GET

Parameters
group
the group name
username
the user name
recursive
if present with any non-empty value, remove user from all children group optional
ticket
your authentication ticket

Returns
Status code 200
"OK"
Status code 403 Ticket not valid or user not allowed to invoke the service
Status code 500

2.2.2.7 /api/groupmembers

Description It searches for group members. Given a group name, the service returns all group members and children groups. A **JSON** dictionary with the list of user attributes and the list of children groups is returned.

Url structure
`portal.vph-share.eu/api/groupmembers?group=<group>&ticket=<ticket>`

Method GET

Parameters
Group
the group name
ticket
your authentication ticket

Returns
Status code 200

```
{
    "users": [
        {
            "username": "mbalasso",
            "fullname": "Matteo Balasso",
            "email": "mail@example.com"
        },
        {
            "username": "testuser",
            "fullname": "Test User",
            "email": "mail@example.com"
        }
    ],
    "groups": [
        {
            "groupname": "testgroup0002",
            "subscribers": 3
        }
    ]
}
```

Status code 403 Ticket not valid or user not allowed to invoke the service
Status code 404 Group with given name does not exists
Status code 500

2.2.2.8 /api/usergroups

Description It searches for users group. Given a username, the service returns all groups the user is part of. A **JSON** list of group names is returned.

Url structure `portal.vph-share.eu/api/usergroups?username=<username>&ticket=<ticket>`

Method GET

Parameters
username
the user username
ticket
your authentication ticket

Returns
Status code 200

```
[
    {
        "groupname": "testgroup0001",
        "subscribers": 1
    },
    {
```


```
        "groupname": "testgroup0002",
        "subscribers": 3
    }
]
```

Status code 403 Ticket not valid or user not allowed to invoke the service
Status code 404 User with given username does not exists
Status code 500

2.2.2.9 /api/promoteuser

Description The given user is added to the given group managers. If the group has (or will have) any child group, the user will be manager of them as well.

Url structure portal.vph-share.eu/api/promoteuser?group=**<group>**&username=**<username>**&ticket=**<ticket>**

Method GET

Parameters
username
the user username
group
the group name
ticket
your authentication ticket

Returns
Status code 200
OK
Status code 403 Ticket not valid or user not allowed to invoke the service
Status code 500

2.2.2.10 /api/downgradeuser

Description The given user is removed from the given group managers. If the group has any child group, the user will be removed from its managers as well.

Url structure portal.vph-share.eu/api/downgradeuser?group=**<group>**&username=**<username>**&ticket=**<ticket>**

Method GET

Parameters
username
the user username
group
the group name
ticket
your authentication ticket

Returns

Status code 200
`OK`
Status code 403 Ticket not valid or user not allowed to invoke the service
Status code 500

2.2.3 Resource Access

2.2.3.1 /api/hasrole

Description It checks if the user is granted the target role over the given resources.
Resources can be referred by metadata global ids OR by local ids and resource type.
User authentication can be performed by passing the ticket as an URL parameter OR with http basic authentication (username:ticket)
It returns **True** if the user is granted the target role over the given resource, **False** if not.

Url structure

`portal.vph-share.eu/api/hasrole?local_id=<local_id1>&local_id=<local_id2>&type=<type>&role=<role>`
`portal.vph-share.eu/api/hasrole?global_id=<global_id1>&global_id=<global_id2>&role=<role>`

Method GET

Parameters

global_id
the resource global id **optional**
local_id
the resource local id **optional**
type
the resource type **optional**
role
the user target role
ticket
your authentication ticket **optional**

Returns

Status code 200
`True`
Status code 200
`False`
Status code 403 Ticket not valid
Status code 404 Requested resources not found
Status code 500 Any other kind of error

2.2.3.2 /api/resources

Description It gets the list of the resources of the given type for which the user is granted the target role.
User authentication can be performed by passing the ticket as an URL parameter OR with http basic authentication (username:ticket)
It returns a *JSON* list of resources with their local and global ids.

Url `portal.vph-share.eu/api/resources?type=<type>&role=<role>`

structure

Method GET

Parameters
type
the resource type
role
the user target role
ticket
user authentication ticket **optional**

Returns
Status code 200

```
[
    {"local_id": <localid1>, "global_id": <localid1>},
    {"local_id": <localid2>, "global_id": <localid2>},
]
```

Status code 403 Ticket not valid
Status code 500 Any other kind of error

2.2.4 Notification Service

2.2.4.1 /api/notify

Description It provides the service to notify a specific user or a group member.
When the service is invocated, the user (or members group) receives an email and at the same time if he/she loads a Master interface page, a message popup appears.
The notification service can be performed by passing the ticket, the recipient (username or groupid), the message and the subject, as URL parameters.
It returns status code 200 if the notification is delivered, 400 if there is a malformed request, and 403 in case of error.

Url structure

```
portal.vph-share.eu/api/notify?ticket=<ticket>&recipient=<recipient>&message=<message>&subject=<subject>
```

Method GET

Parameters
ticket
your authentication ticket **required**
recipient
the username or group id **required**
message
the content of notification **required**
subject
you have the possibility to specific a notification subject, if need **optional**

Returns
Status code 200
Status code 403 Ticket not valid
Status code 400 Malformed request

2.2.5 The web application

The Master Interface represents the main access point for users to data, workflows and services from the VPH-Share Infostructure. It is a web-application, which has been developed as part of Task 6.4 relying on the Django high-level Python Web framework[1].

Due to the high level of interaction with all the technical WPs, the set-up of a collaborative environment was highly important to efficiently and productively work together. To this purpose, the source code was shared with all the developer team on Github[2] and two instances of the application were deployed:

- A production instance reachable at portal.vph.share.eu on which only consolidated features are deployed for end-users evaluation,
- A development instance, at devel.vph-share.eu, which is used by the developers during the implementation to test the new functionalities and their integration.

The overall system is described in the next sections with preliminary technical information on the security and authentication layers.

2.2.6 The security layer

Security is an important aspect of the VPH-Share platform. The security proxy is implemented as part of WP2 and here just the aspects related to the Master Interface are described (see D2.6 for more details on the security proxy). Being the MI the point of access for the user, the MI is responsible for passing to the other services the information on the user identity and granted permissions.

The Master interface uses Biomed Town as OpenID identity provider (see next section for more details); if the user is recognised as a valid one, a session is opened in the MI and a valid ticket generated. The authentication ticket holds all the information about the user and is signed by the MI to prove its authenticity. The ticket is structured as the follow:

uid=<username>;validuntil=<expire-time>;cip=<ip>;tokens=<roles>;***udata*=<username>,<fullname>,<email>,<language>,<country>,<postcode>**;sig=<ticket signature>

(The parts in bold are under discussion for removal from the ticket but the general structure of the ticket would not change).

- uid: username of the valid user;
- validuntil: timestamp indicating when the ticket validity ends; at present this is set to 12 hours from its creation; after this time it is refused as invalid; in case of longer processes services are provided to automatically regenerate the ticket and not block the execution;
- cip: IP of the client which generated the ticket;
- tokens: here are inserted the roles and all the permissions assigned to the user. The

[1] https://www.djangoproject.com/
[2] https://github.com/b3c/vphshare

permissions are used in particular by WP2 security proxy, and are in the form <nameresource>_<typeresource>_<permission(admin/read/edit>);

- udata: contains information on the user;
- sign: this is the ticket signature that is generated from the MI with a private key DSA 2048 bit. This is used by all external services to verify the ticket validity.

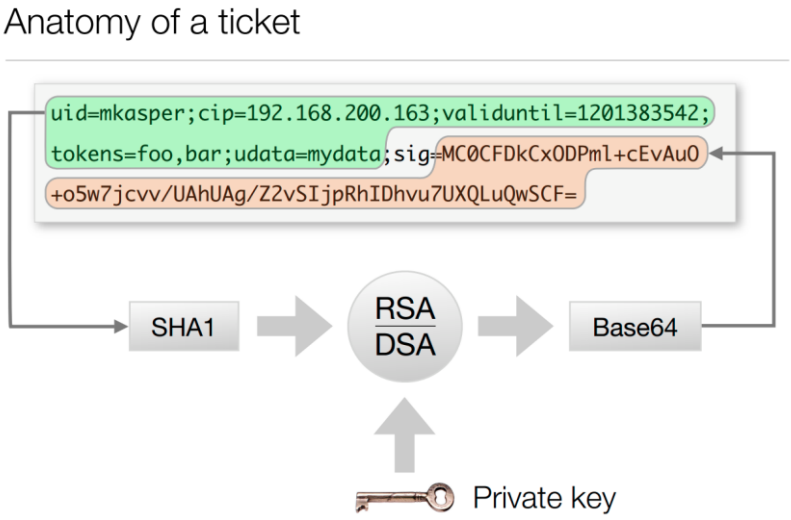


Figure 2. Ticket signature creation

The authentication ticket is based on the Apache mod_auth_pubtkt[3] technology. The mod_auth_library5[4], a Python package to support the ticket creation and validation, has been developed and integrated into the Master Interface. The ticket is signed with the DSA Algorithm[5] with a 2048 bytes key. The ticket is then compressed in base64 to make easier the communication of different systems.

Anyone, who has the below public key and a ticket, can verify it and if valid it can open a session being sure that the information contained in it are safe and generated from the MI.

-----BEGIN PUBLIC KEY-----
MIIDOzCCAi4GByqGSM44BAEwggIhAoIBAQDYHe5/lDkW2i2bJTx4jcMERgL477G+
T13a2KhOw3ld17asktxZtvlJsxEpkdlTUTe2FFdssrcjQ+bVytBJgUf3KjfK7rhK
SuVLzNC09Z3vjhr983WrBG7or7r/hKjPlgaLsNNS1GVjHEHS+jurGaE+7LhnXmfu
Z4Ly5wA2NQdbp1AbcQ6CjqXdtKYw+7MQdj3cacM1PArWhnVDdPC02TwZJA7ae4O1
wbPcUQmxtIMw3FYR5emjbz2C98VUvUdI6FbS1QdV/ZLPDP3j0IRcdQ/RrnyMLtbN
32p+H4xdrHbDQOnoRWhGhxcNp/k/xgJtHx2z+/lpbtUyCqPVOwsq6cO9AhUA+ySB
y+jH9iSi/TITIISmarwuA0ECggEBALilFPDXQTT4UpxkLer46KV0cnoek5AbNMFh
fKdm/E2P9CzBgHLk/Q1UiZRou7rRKvtvond4d7CeTK2XVa7uQM0Rbg1O7ABbczvh
a04dWggGAHr5rEZk6OBPzgW++YwvLEisF9f2vTufzWhAgOMUYWSI+joc/lLjFOuf
TNLey2s7bjELP7nA6TmGrCtR/XOliLPjloA9O5TSjYDHLMCq9r+TGQEm51dh2Tfs
KQnkbu/0It6ECr0t/G+5chJin5R0x5qO/yqdywEzhhNwd+G0eMcY6B/aJwNn/wNe
zyuMenTyPwoFZHm4PtXjyuz0+41n39v0V5Rhc3VC9R27t0wa1hsDggEFAAKCAQBZ
aQZsWveEVgi73QL8qb+9b+EeG3GPEM1H5AOxOqq+rPRHS2+dJjiDvUeZD/cDNW+c

[3] https://neon1.net/mod_auth_pubtkt/
[4] http://pypi.python.org/pypi/mod_auth_library/1.0
[5] http://en.wikipedia.org/wiki/Digital_Signature_Algorithm

EYv949skpIrZkthDiRWxaf6ZmhAB66mg4dmgLJWtN61lrYzD8n1RacH0HmBe20R8
DL+UKBoyRllB27IVLbFGCTl77jYsDxP6Q7uL4koJOkN2FtYJCxqOMGAfhIqbtibg
WJm1CQAKYb0mON+rTonOwzoK8GHtzqXtkbeY5HbBSOdiOHJCjtfFZDEZS0FaZXT+
OfULRFdOouooIdiQyNxKsId/pkL6hBXL6QvVzfVaGiE5nhWskgtmOsKanWWLeGtR
sv5tHXp48zLsDtXeFncp
-----END PUBLIC KEY-----

At the Master Interface level the same ticket is saved in a cookie, named *vph-tkt*, and it validates it at each request of a page. In fact, the MI provides a ticket validation service that allows the ticket consumers to validate the ticket and to retrieve a JSON26 notation of the user attributes. In accordance to the current regulations, the user will have to explicitly accept the use of cookies for the correct functioning of the system.

The validation service receives the ticket to be validated as an URL parameter. If the ticket is valid, a JSON notation of the user information is returned:

{
"username": "mbalasso",
"language": "",
"country": "ITALY",
"role": ["developer", "friend"],
"postcode": "40033",
"fullname": "Matteo Balasso",
"email": "m.balasso@scsitaly.com"
}

If the given ticket is not valid, the HTTP 403 status code is returned.

If a user needs a copy of his/her own actual ticket (for example in the deployment of Applications/Atomic services) he/she can copy it into the clipboard by using the button available in the user profile page.

Two APIs are available associated to the ticket:

1. https://devauth.biomedtown.org/refresh_tkt?ticket=<ticket>
   It renews the ticket timeout for other 12 hours. It accepts requests GET (to be deprecated in the future) and POST. It returns a plain text answer with the new ticket.
2. https://devauth.biomedtown.org/user_login?username=<username>&password=<password>&domain=VPHSHARE
   It generates a valid ticket for the Master Interface. It accepts requests GET (to be deprecated in the future) and POST. It returns a plain text answer with the ticket.

[6] http://en.wikipedia.org/wiki/JSON

2.2.7 Authentication mechanisms

As already reported in D6.3 and repeated here for completeness, the authentication mechanism provided by the MI uses a decentralised method based on the OpenID protocol. The MI acts as the relaying party and the authentication is demanded to an external Identity Provider in which the user's information is stored. The Master Interface relies on this Identity Provider to assign the correct privileges to the user. A schematic representation of the authentication method is presented below (Figure 3).

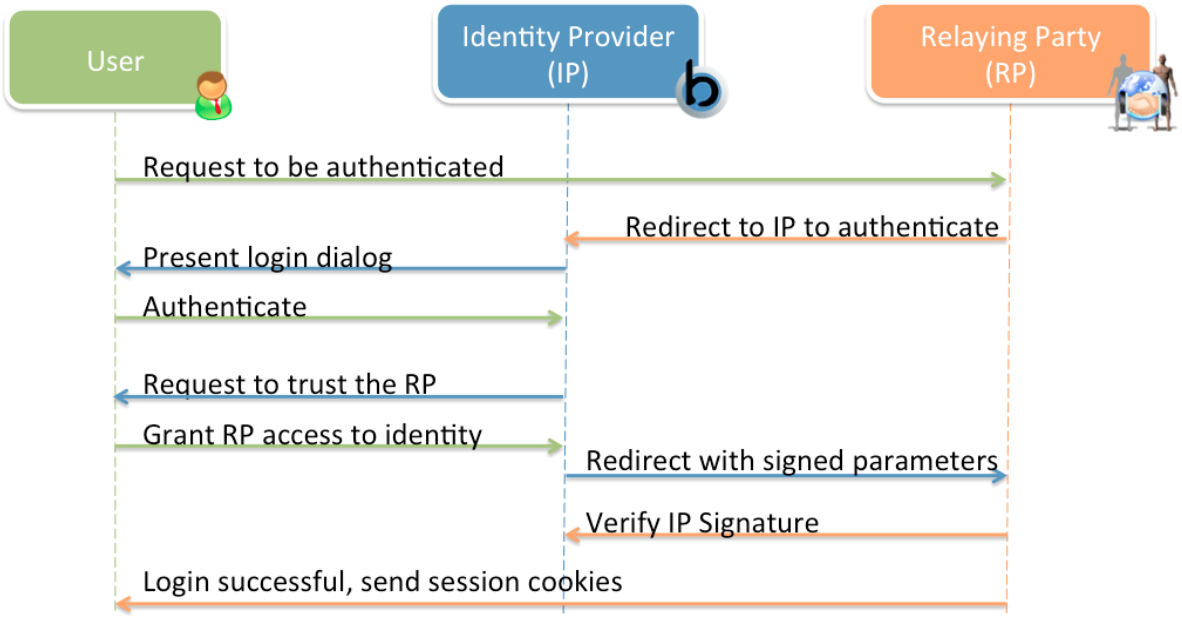


Figure 3. Schematic representation of the OpenID authentication mechanism

The main actors of this process are:

- The User, who wants to access the MI and needs to be authenticated;
- The Identity Provider, representing a trusted service where the User's identity is registered;
- The Relying Party, representing the service where the user wants to be authenticated (in this case the MI).

The authentication mechanism works as follows: when the user tries to login into the MI (Relaying Party), his/her request is redirected to an external Identity Provider that shows the login dialog and handles the authentication process. According to this authentication, the Identity Provider assigns the appropriate privileges to the user. These privileges are sent back to the MI that will now be able to successfully terminate the login process and open a dedicated session for the user, according to his access rights.

Currently the only Identity Provider supported by VPH-Share is the Biomed Town[7]

[7] http://www.biomedtown.org

community portal. This has been a choice of opportunity to simplify access to the thousands of users of the biomedical community already subscribed to Biomed Town. However, the technical solution in place will allow in the future to update the MI to have others relying parties.

2.3 The user interface

In this and the following sub-sections, we will provide an overview of the available functionalities in the third year prototype of the MI and their user interfaces and we will always refer to the production instance of the Master Interface and associated services so to give a clear presentation on what a general user will get now by entering the VPH-Share system. A number of other services and improvements are already available in draft form in development and they will be mentioned just when appropriate to the user experience description.

The home page of the portal (at https://portal.vph-share.eu) shown in Figure 4, provides the user all most important links to the VPH-Share infrastructure services at a glance.

Figure 4. Master interface home page for not logged user

At the top right side the links to the help and documentation pages are available together with the log in/registration.

In the top bar, you can find links to all VPH-Share resources:

- data: this links to the main data page which provides access to the main actions, which can be executed on the data that VPH-Share host: browse, search, upload, and manage.
- applications: this link provides (after log in) access to the services/tools/application deployed on the cloud platform and called also Appliances.
- workflows: this redirects to the main workflow pages, where the user can find which actions are available on these resources: browse, search, upload, and manage.
- search: it sends to the main global search services which allows the users to look for any type of resource hosted into the infrastructure.

In the bottom of the page there are links to:

- beta user program: where the user can find information on how to be part of the beta evaluation of the VPH-Share platform.
- workspace: this link (not yet active) will lead to the page where the user can compose his/her own data and tools to compose his/her own workflows or to access the history of workflows executions (a preliminary version is ready on development but not yet on production).
- groups: this link provides access to the institutions and groups management and subscription page.
- search: it sends you to the main global search services which allows to look for any type of resource hosted into the infrastructure.

When the website is opened, at the top of the pages a warning is presented to the user on the fact that the website uses cookies and asks for acceptance on their use.

In collaboration with WP8 and the PMO, a new version of the MI homepage is under design; the aim is to make the first page where the user arrives more informative on the number and types of resources hosted.

2.3.1 User registration

Relying on the previously described authentication layer, the user is allowed to register to the MI by clicking on the corresponding link in the home page.

A form will appear which asks the user to fill in a series of information:

- username
- contact details
- acceptance of the privacy terms
- security capture to avoid spam.

Due to the fact that the VPH-Share system is meant to be used by professionals and the contact information will be used also by resource owner to take decisions for granting access, the form is checked by a manual operator within 48 hours. If the information is incorrect or incomplete (i.e. an institutional email address is required), the request is temporarily rejected with a request of update; otherwise the request is approved and an email is sent to the user for the password set-up.

Once the password is inserted into the system, the user is automatically added to the VPH-Share user database and he/she can access all the information that requires a log in.

2.3.2 User access

2.3.2.1 Login

As much as possible of the VPH-Share services, and resources can be browsed and viewed without the need of an account, but if the user would like to access specific resources, he/she needs first to register to the platform and then to login in the platform.

To access the web application, a *login* button is available in the top right side of the page.

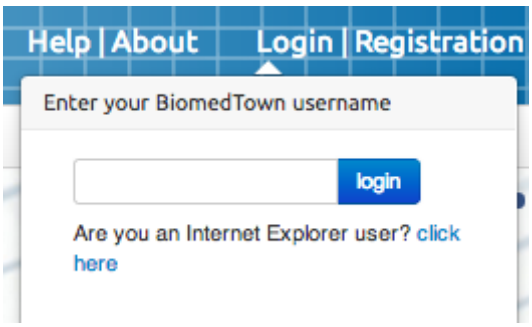


Figure 5. The login modal window

The user can at this point insert his/her own credentials as set at the registration step (or as already set up for Biomed Town) and accept the privacy policies to enter into the system. The choice on the privacy policies will be required only at the first log in.

After the log in, the corresponding link at the top right side changes with the user name and provides a roll down menu which includes:

- dashboard
- profile
- admin tools
- log out

The dashboard is described in section 3.3.7.1 while the others will be described in the following sub-sections.

2.3.2.2 Profile

If the user clicks on the profile link, he/she can get information on his/her own account as provided at registration time, get a copy of his/her own authentication ticket (for its use in the application deployment), or access the groups page:

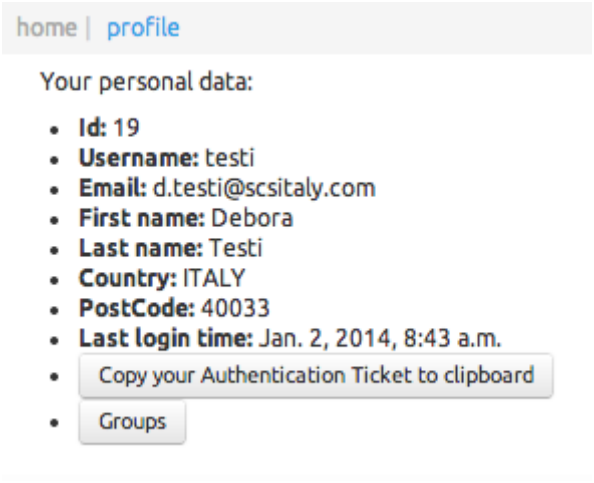


Figure 6. The user profile page

More information will be added in the future, with the possibility also to edit/update some of information provided at registration time.

2.3.2.3 Groups

The Master Interface does not only allow the user to register and access the system, but it provides the user with the functionalities to manage and organise groups of users to assign access rights to the available resources.

Groups of users in the VPH-Share infostructure have been organised and implemented around the concepts of institution and study.

- Institutions represent virtual or physical organisation representing a community of users, like a university department, a research project team. Each institution can create and manage one or more studies.
- Studies are a specific sub-group of researchers typically working on the same research or clinical question.

When clicking on the Groups link, the user is provided with the list of the available institutions and studies present into the system (in green and blue background respectively).

Figure 7. The institutions/studies

The institutions/studies of which the user is member are marked with a green arrow.

If the name of the institution/study is clicked its details are provided. If the user is already part of the selected institution, he/she can see its members and the available studies, while otherwise he/she will have a link to request subscription to the institution. If clicked, the institution managers will receive a notification and will be able to accept or refuse the subscription.

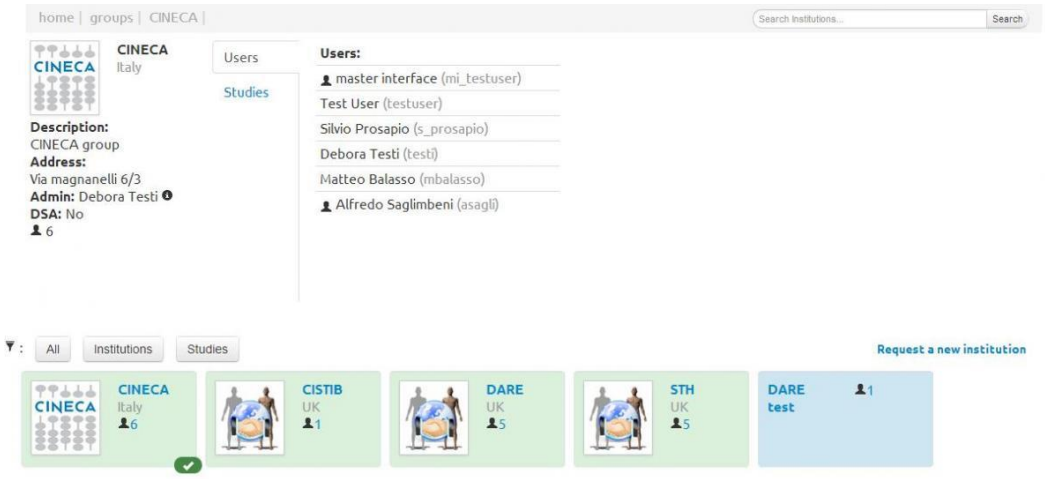


Figure 8. The institutions/studies details view

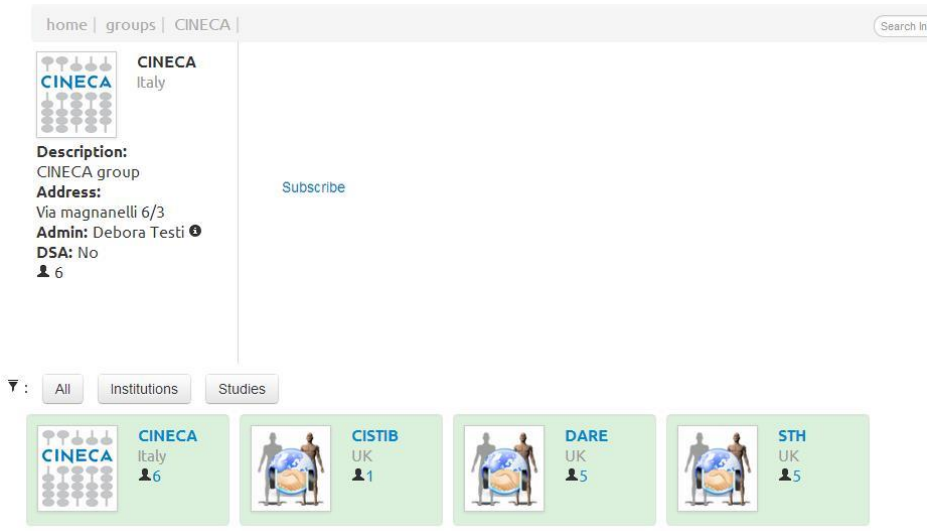


Figure 9. The institutions/studies subscription view

The managers can also remove or add new users with the provided management interface, see Figure 10.

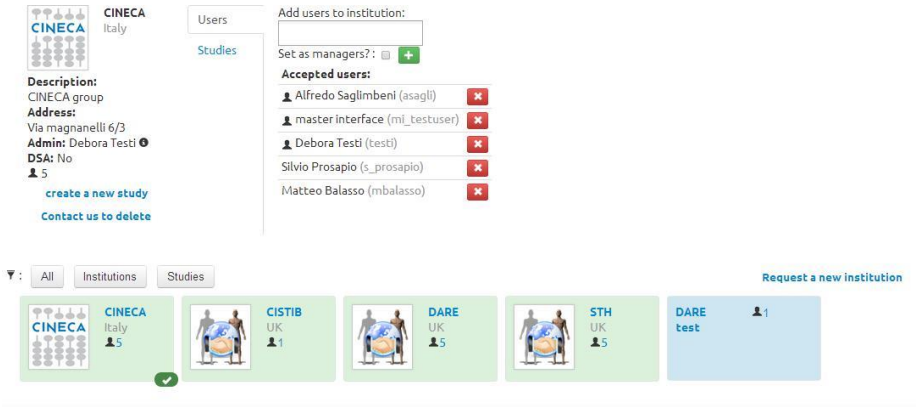


Figure 10. The institutions/studies user management view

Studies can be created by the institution managers with the form shown in Figure 11.

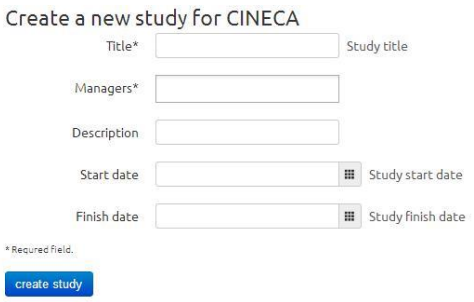


Figure 11. The institutions/studies creation study view

If an institution of interest is not available in the system, the user can request its creation by filling in the form provided, see Figure 12.

The portal administrators will then check the information and if appropriate approve the creation of the new group.

Resource owners can then use the available groups and studies to grant access permission to resources without having to select each user individually.

2.3.2.4 Admin tools

This page is available only to the portal administrator, and it allows assigning specific permissions to subscribed users (like the developer role or the administrator one) to grant access and control on resources not opened to the standard users.

Figure 12. The request institution view

2.3.3 Discovery tools

The aim of VPH-Share is to provide effective ways to the user to discover resources they might be interested in. For this reason, different discovery and search tools have been implemented and deployed. In most of the cases, the MI provides the User Interface while calling backend services developed by other WPs as shown in Figure 13.

2.3.3.1 Plain Search

This is a simple text search i.e. Google style. The button to use it is in the right side of the top bar. The user is allowed to insert one or more textual terms. By clicking on the drop down '*All resources'* menu the user is allowed to filter the search on a specific resource type, i.e. Dataset for structured data.

Figure 13. Plain search interface

When the *Search* button is pressed, the terms are searched on all the selected resource types and the results are presented.

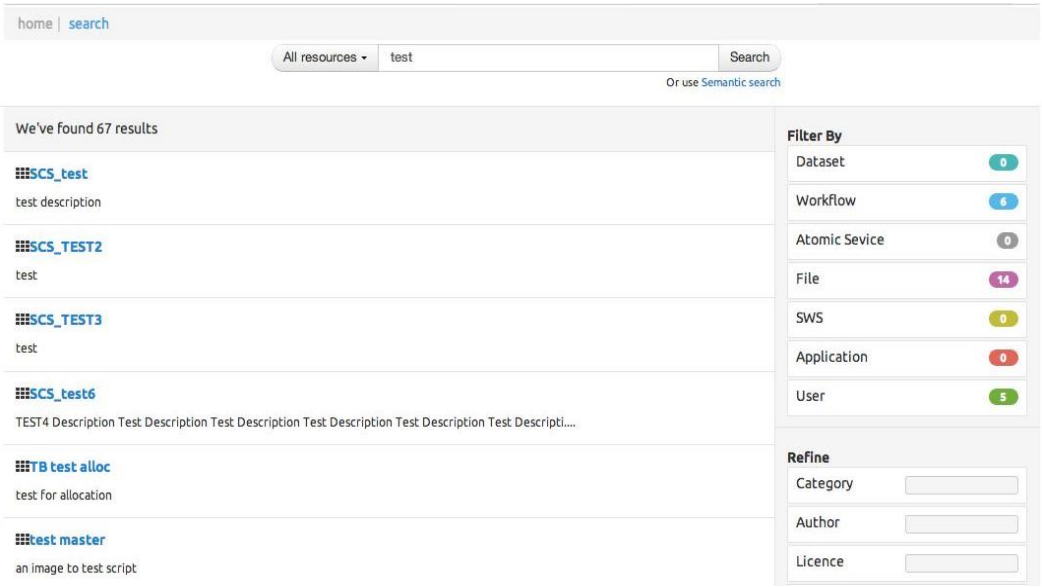


Figure 14. Plain search result page

On the right side of the page, there are the additional filters that can be applied to the results list. On the top, the '*filter by'* options filters based on the type of resource. The user can select more than one type at the time (the selected types are shown with a blue back colour). To deselect a type he/she has just to click on it again. As soon as a filter is applied the results list is automatically updated.

The '*Refine*' options can be used to add additional filtering to the data by adding extra information. The refinement is performed by inserting the text in the appropriate field and then pressing enter on the keyboard.

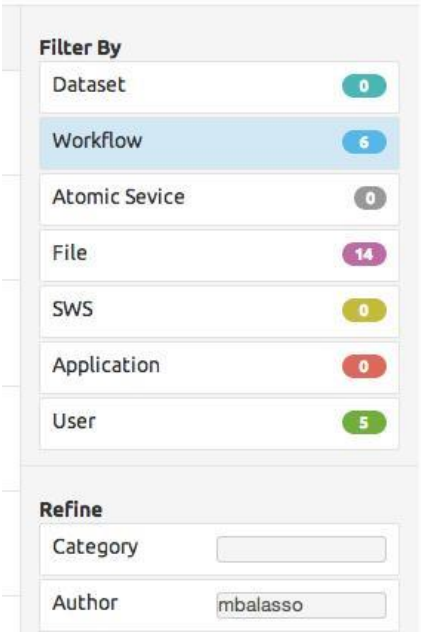


Figure 15. Filter and refine options in the plain search

2.3.3.2 Semantic Search

The link to the Semantic search (as user interface to the WP4 developed services) is available just below the search button of the global search and it can be used for a more detailed search with respect to the free text one provided by the Plain search described above.

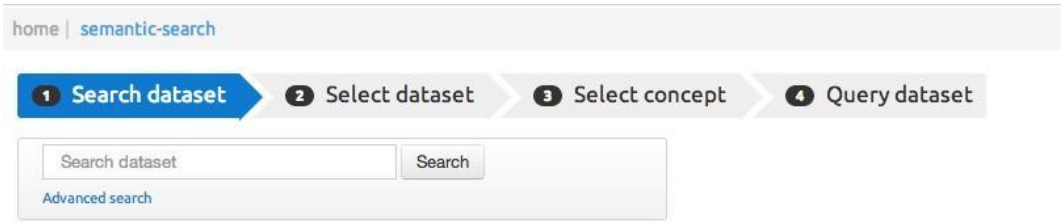


Figure 16. First step of the semantic search

The first step allows the user to enter the semantic term (or combination of terms). The output will be the list of resources where the specific term is present anywhere in the resource information.

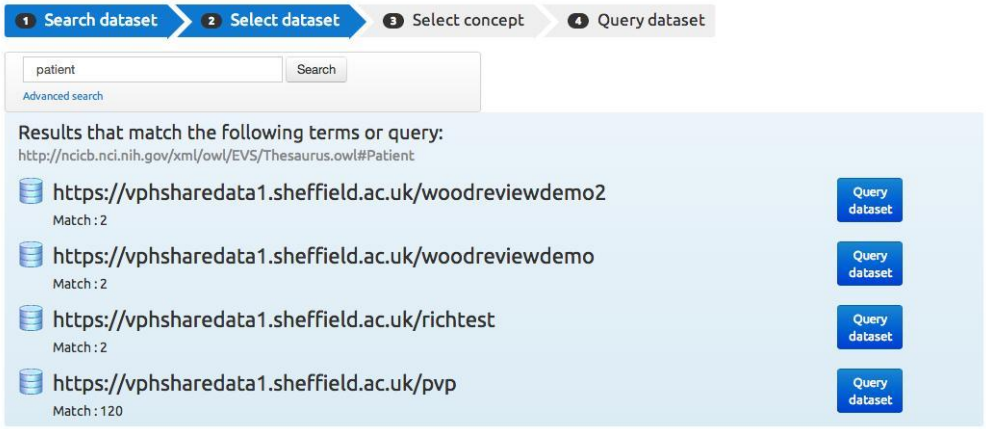


Figure 17. List of datasets corresponding to the semantic terms

From the list of the found resources, the user can choose which one to further query based on the concepts used during the annotation phase. Moreover, he/she can further refine the concepts or select one from the available list.

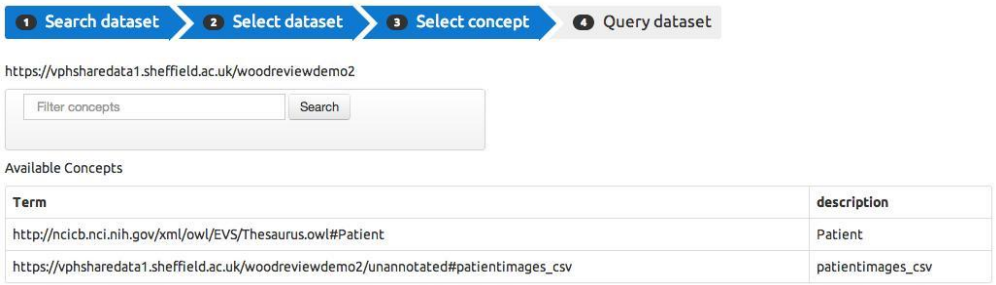


Figure 18. Dataset query

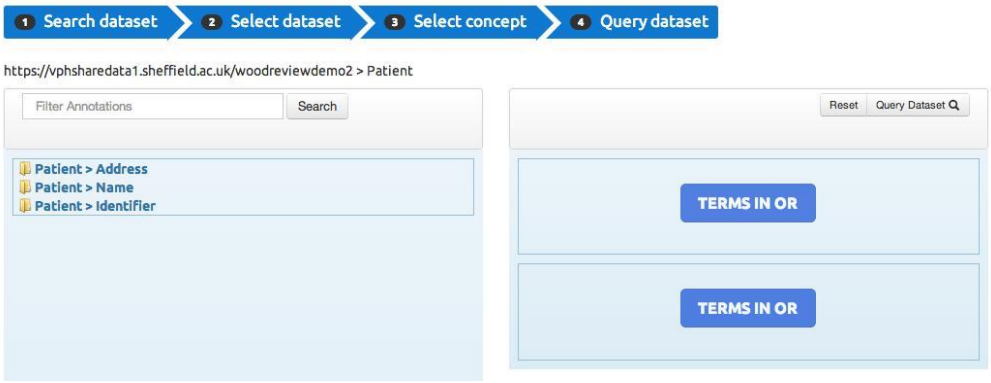


Figure 19. Dataset internal query

The terms can be chosen on the left box and composed with AND and OR operators by dragging them in the right boxes. The two right boxes are composed with the AND and the terms in each box are composed with an OR operator.

When the concept is dragged, a window is to set the value with exact match or inclusion criteria is shown.

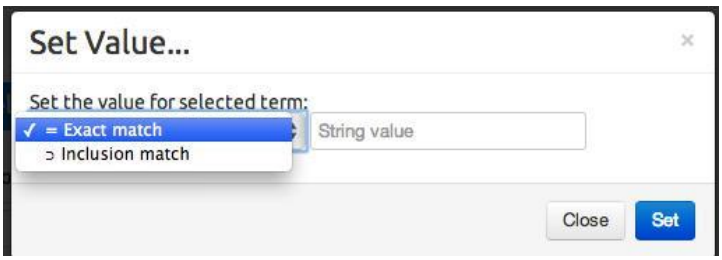


Figure 20. Set value for inclusion criteria

After the selection of the criteria, each one can be still edited or remove before pressing the *Query* dataset button.

2.3.3.3 Browsing

There is also the possibility for users to browse in each resource category all the available resources.

Data: for the data resources, the browsing can be achieved by domain category (i.e. cardiovascular, respiratory, etc.): a carousel of images for the different medical domains the data can belong to is presented and by selecting one, the user will be presented with data whose associated metadata are related to this category. An Alternative browsing is the alphabetical listing, the user can move from one initial to another using the top bar letters and click on the data name or icon to visualise the resource information and eventually access it.

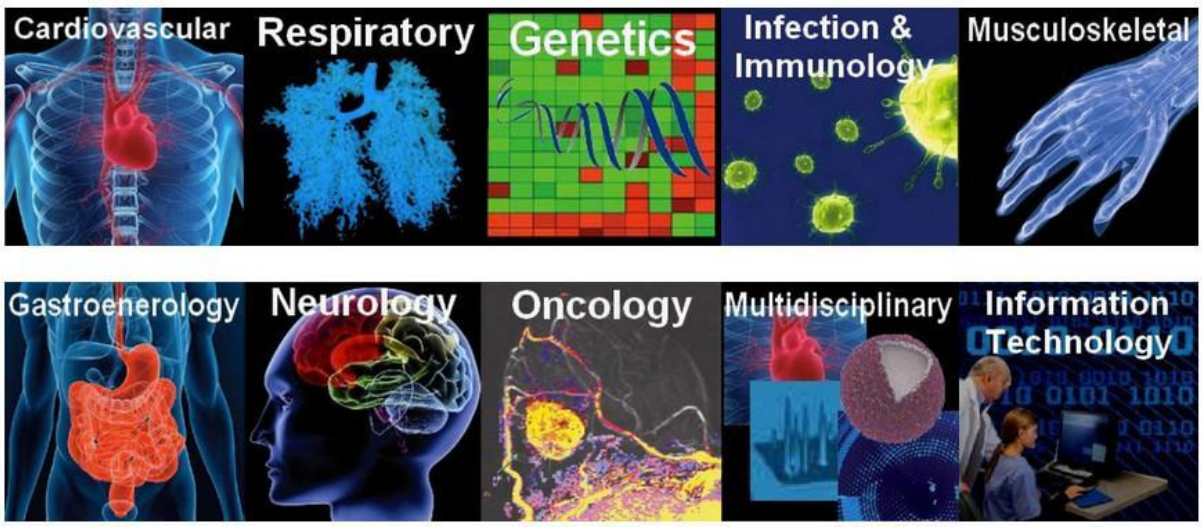


Figure 21. Data browsing per domain category

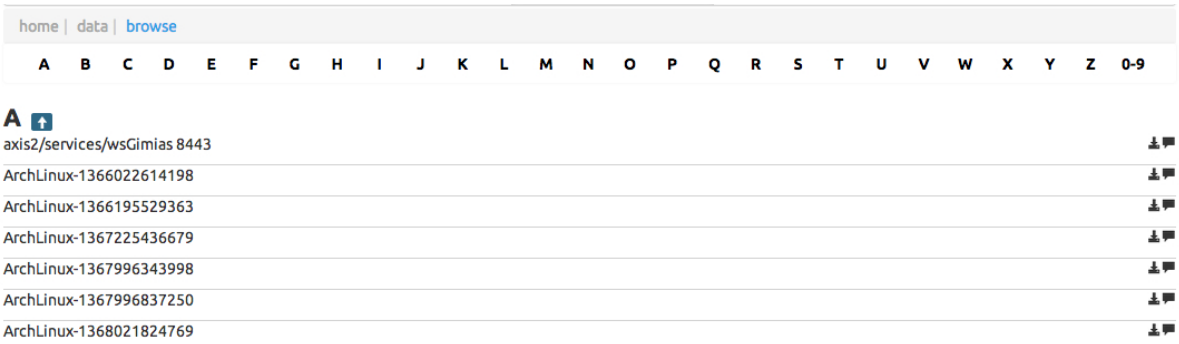


Figure 22. Alphabetical data browsing

Workflows: the list of all the available workflows appears in the screen. By clicking on the name the user can get information on the specific workflow, while if the user is owner/manager of a workflow an icon provides also access to the editing form.

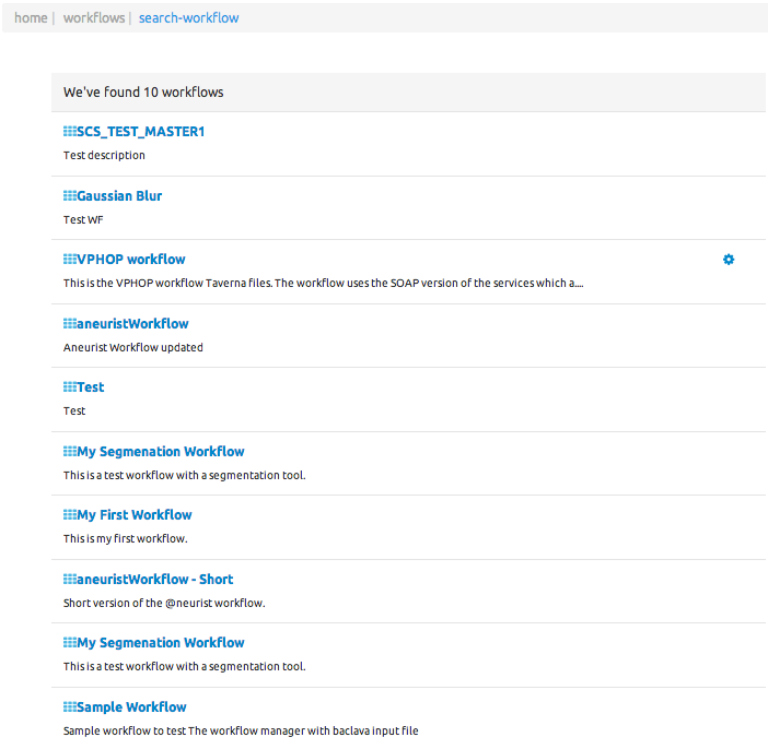


Figure 23. Workflow browsing

Applications: for the atomic services, when the *APPLICATIONS* in the top bar link is clicked the first page shows the already running Applications for the specific user, while clicking on the '*Start new application*' button provides the list of all the available tools in alphabetical order. Each Application has a description provided by the service owner and the link to start the service (if the user has the necessary permissions).

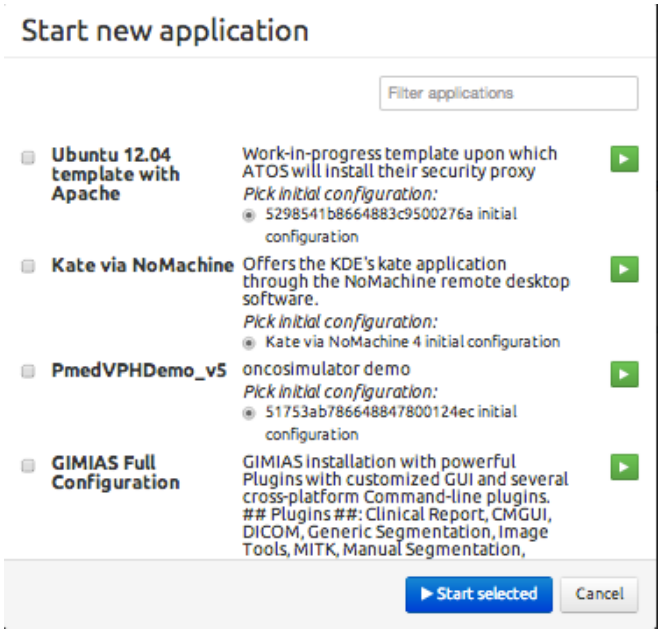


Figure 24. Appliances browsing

2.3.3.4 *Resources access*

When browsing the VPH-Share portal, the user finds a resource of interest and visualises the information associated to it as recovered from the WP4 metadata services (see D4.6 for more details).

All the resources (data, applications, or workflows) have the same presentation layout of the information, which relies on the common metadata model provided by WP4. Thus, we will here refer to a workflow as an example, but the same information would be found for the other resource types.

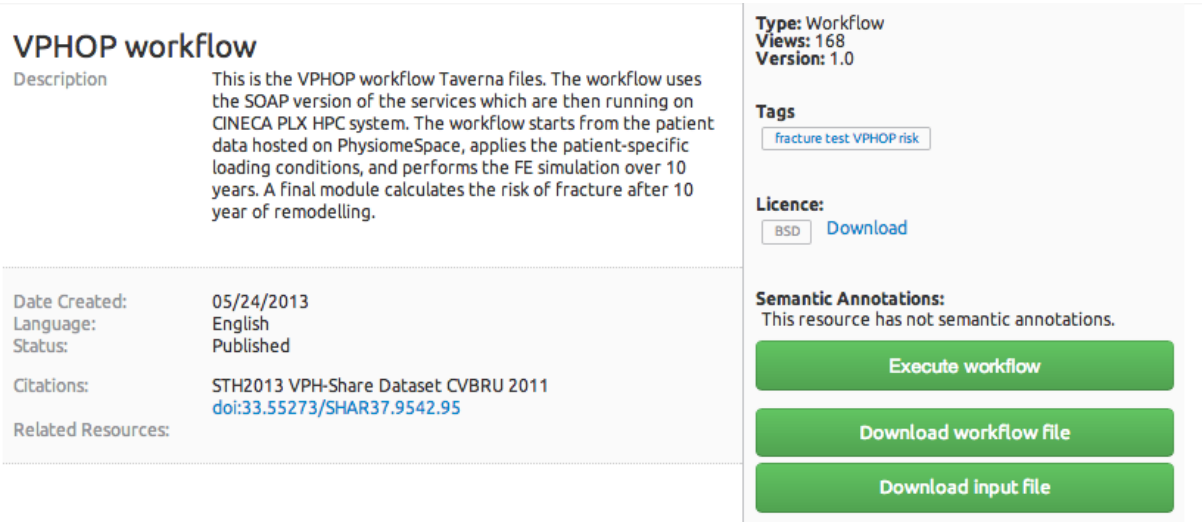


Figure 25. Resource information page

The information currently includes the metadata below; a revision of the metadata model is under finalisation and thus new metadata might be added in next versions of the system.

- the resource title
- the resource description
- the associated information: creation date, language, related citations, type (workflow, data, etc.) which are inserted at the upload time
- citations of publications related to the resource
- if there are other resources in some ways associated or related to the present one
- the views: a counter of the number of visits to this resource which is automatically updated
- the version number
- the tags, which can be used in the global search service
- the licence type and eventually the associated licence file available for download
- the semantic annotation URI

On the left side at the bottom, one or more buttons are present which allows you to access the resource.

The button will have a colour and an associated action depending on the permission the user has been granted on the specific resource.

If the user does not have the permissions to access the resource, the button will allow the sending of a request for sharing to the resource owner where a personalised message can be added to the request to specify the purpose of the resource need. When pressed, the user is notified by email that the request has been delivered. At the same time, the resource owner receives an email informing that there is a pending request in his/her dashboard waiting for approval together with the text of the message typed by the user.

Once the request for sharing is sent, the button will change state and the user will not be allowed to perform any further action until the resource owner approves or rejects the pending request.

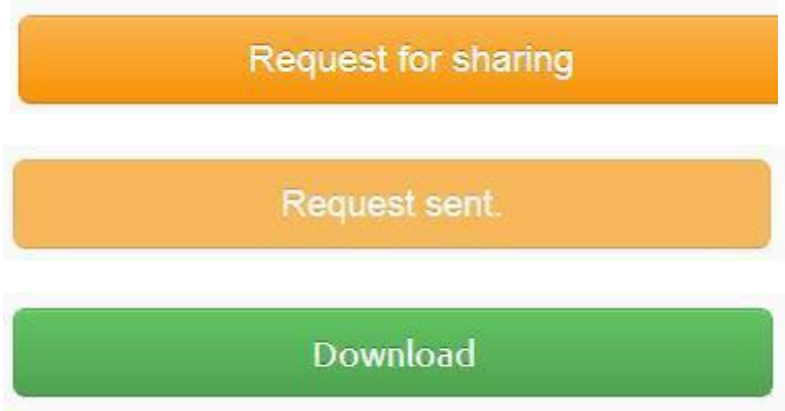


Figure 26. Resource access buttons

As soon as the resource owner approves/rejects the request, the user will be notified by email and the button will change state and colour. Also in this case the resource owner can motivate the rejection with a message or request for further details, which will be delivered to the user together with the email notification. At this point, if the access has been granted, the buttons can differ in type and number according to the resource type:

- file (unstructured data), a download button;
- dataset (structured data), a query button;
- workflow, a download button for the workflow file, a download button for the input files and an execute button;
- Application, an invoke service button.

Other specific action buttons will be added for specific applications end-points behaviour (i.e. for web services).

2.3.4 Resources: Data

2.3.4.1 Remote visualisation

For the unstructured data, for each file there is also a new icon (eye shape), which allows the user to have a remote preview of the data.

This remote visualisation solution has been implemented by integrating ParaviewWeb. Paraview[8] is an open-source multi-platform data analysis and visualisation application. It allows exploring and 3D visualisation of data or it allows creating batch script to process data. The software package supports also big data thanks to a distributed calculation mechanisms, which makes the system very efficient. From its version 4, Paraview is distributed together with ParaviewWeb, which makes available a series of tools for the rendering and sharing of 3D data and makes possible the integration of Paraview functionalities into a web browser (API Javascript).

In the first implementation available in this production release, the remote visualisation has been activated for a limited number of file types (i.e vtk) and for a limited type of data structures (i.e. structured grid, polydata, etc). More data formats and data structures will be added in year 4 according to the beta users requests.

When the user clicks in the preview icon, the data is loaded and a panel is opened in the top part of the page. The data are transferred to the visualisation server where ParaviewWeb performs the selected rendering, which is sent back to the MI panel where the user can interact with the visualisation.

[8] http://www.paraview.org/

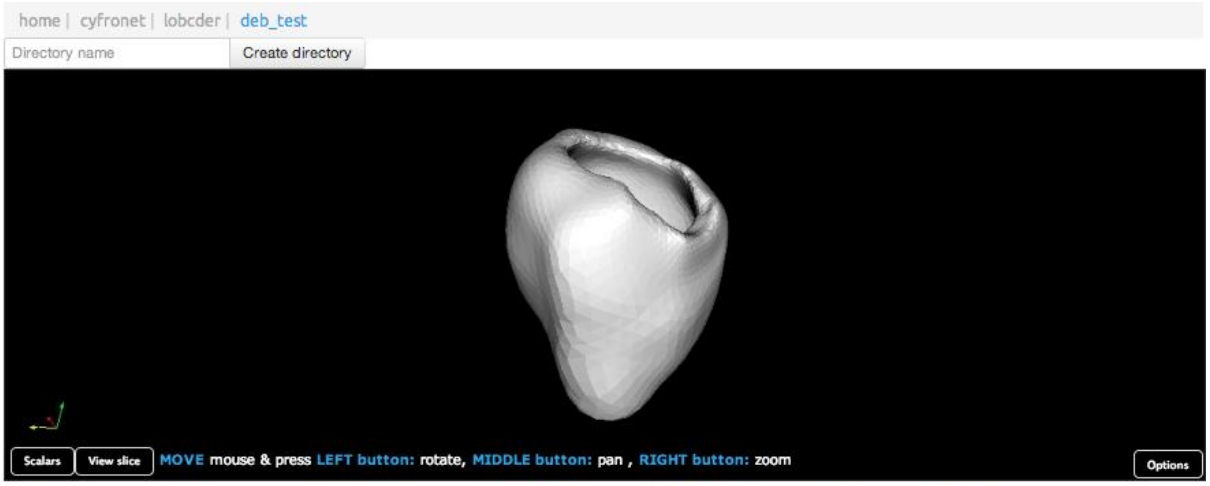


Figure 27. Preview of a polydata vtk file representing a human left ventricle

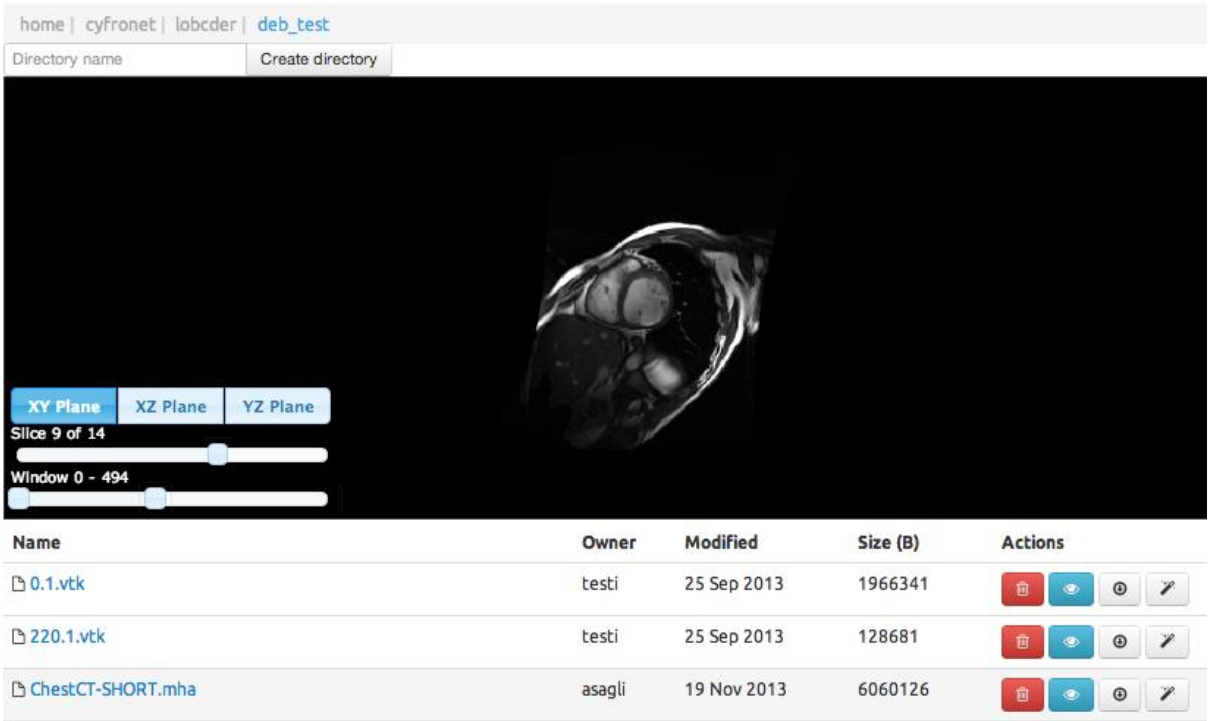


Figure 28. Preview of a 3D volume as slices along the coordinate axis. The user can control direction and position of the slice with the bottom left controls.

The user can switch the visualisation from 3D rendering to slice rendering according to the type of data. For the 3D rendering the user can also interact with the view by changing the point of view with the mouse (standard pan and zoom interaction), while for the slice rendering he/she can move the slice direction and position, and change the windowing moving the respective sliders. Much more visualisation modalities and options are available in ParaviewWeb for potential exposure into the MI; for the moment a limited number has been integrated for effort constraints and to wait for the user feedbacks on which might be the functionalities of interest for future implementation.

2.3.4.2 Upload

2.3.4.2.1 Structured data (DPS)

In VPH-Share we primarily refer to tabular, and potentially relational, data sources such as CSV extracts and relational databases, as Structured Data.

A desktop application, Data Publication Suite, has been developed as part of WP3 to support the process of publishing clinical or research data sets in a secure Internet accessible way. The data once published may be browsed using an RDF Browser or queried using SPARQL and an SQL type protocol from the OGSA-DAI project.

The general process for publication, although many of them are not mandatory, is as follows:

- Import a data source
- Define relationships between the tables if they exist and are not automatically detected
- Semantically annotate the data
- Create a destination container of the server
- Create a new destination based on a data source
- Define a de-identification profile for this destination
- Publish the data
- Manage the access list for the resource

More information on the DPS technical details and its use can be found in WP3 deliverables.

On the MI side, a page is available to users with a short description on the DPS, links to the documentation and to download the application and test data.

2.3.4.2.2 Unstructured data (LOBCDER)

In VPH-Share we refer to medical images and binary information as unstructured data. These are managed in the infostructure thanks to the LOBDCER service (see section 3.3 for more technical details).

In the MI, LOBDCER is presented similarly to a shared folder where sub-folders and files can be created.

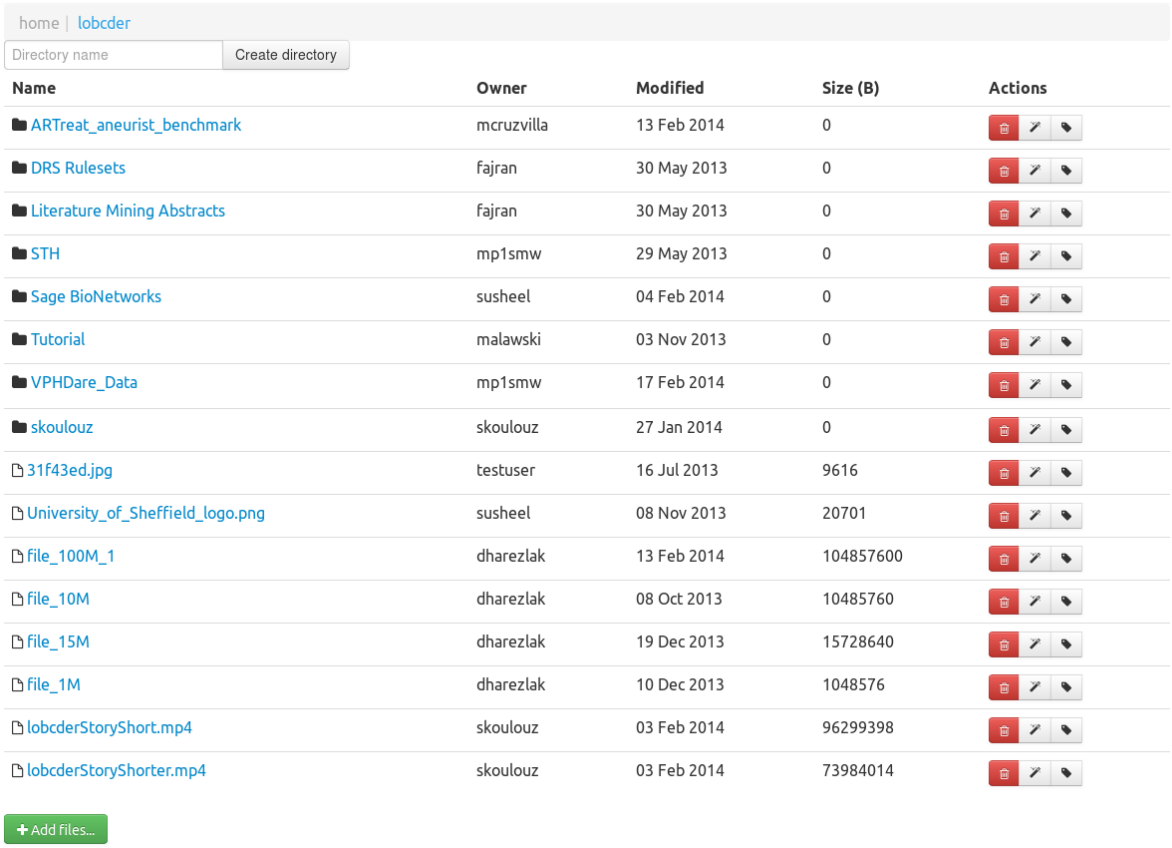


Figure 29. LOBDCER interface

In the LOBDCER, the user can browse the sub-folders and for each folder the available files. If the user has the permissions, he/she can download the file by clicking on the file name link.

The file owner can also delete or modify the file metadata respectively with the trash bin and pencil icons at the end of each row. If the data format supports it the preview icon is also available to obtain the file remote visualisation as described in section 2.3.4.1. The tag icon allows the user to copy the path of the file or folder in the computer's clipboard so that it can be pasted afterwards in any other application. This is especially useful, for instance, for the user to input paths to a workflow execution application, see section 4.

To add a new file the user can use the *Add files* green button or can create new directories with the top button *Create directory* inserting in the text box the name of the new older.

2.3.5 Resources: applications

2.3.5.1 Create a new Application

The creation of a new Appliance is part of WP2 and just exposed into the MI. Details on this can be found on WP2 deliverables and in section 3.2.1.

In short, the user can create his/her instance of the Application VM and register it into the system. This can be done also relying on a number of templates created by WP2 for the

different Operating Systems with already configured all the necessary VPH-Share services (like the security proxy or the LODBCER data connection).

2.3.5.2 Run an existing Application

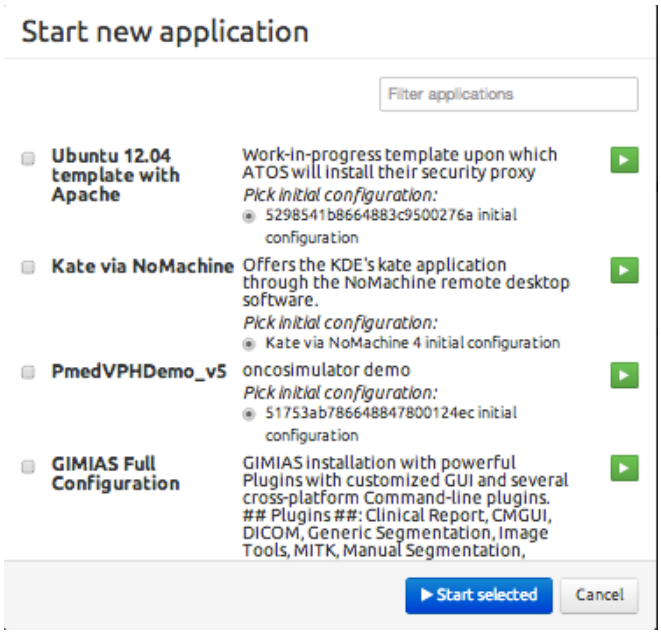
Applications that have been created and saved by developers may be used by any user of the platform. There are several ways in which the VPH-Share cloud platform enables use of Applications:

- By asking a specific service to be instantiated using the so-called Generic Invoker (see section 3.2.2 for more details).
- By authoring a Taverna workflow which makes use of the VPH-Share Taverna plugin to automatically instantiate the required services.
- By writing user-owned software, which communicates with the Cloud Façade interface of the Atmosphere platform and makes use of its API to instantiate and invoke services.

In this section we will focus on the Generic Invoker as a tool provided directly by the VPH-Share platform and targeted for end users rather than application developers.

The Generic Invoker is a facility provided as part of the VPH-Share Master Interface and enabling to use specific applications using a straightforward GUI, without worrying about the technicalities of service instantiation and invocation. Technically, the Generic Invoker is part of the Cloud manager portlet.

Once the user has logged in, he/she can get the list of the available Appliances (as reported in the Browsing section).



When the *Start* green arrow icon next to the selected service is pressed, Atmosphere will prepare that service for user interaction. The contents of the window will change to reflect the fact that a service is being spawned, and inform the user when the service becomes available.

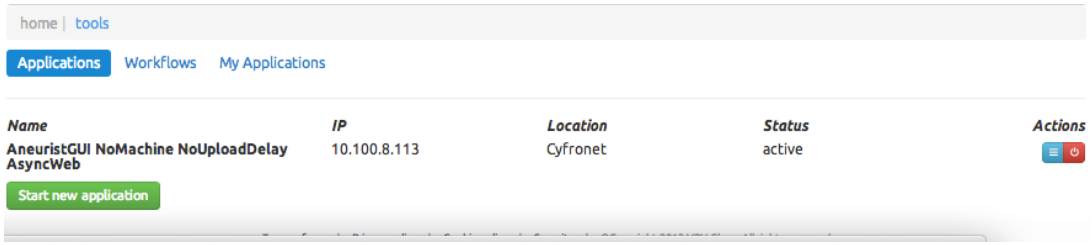


Figure 30. The application view after invocation

At this point the user may click on the details blue icon to learn about the possible ways in which he/she may interact with the Application Instance. The Cloud Management portlet will display a suitable dialog listing the interfaces provided by the instance:

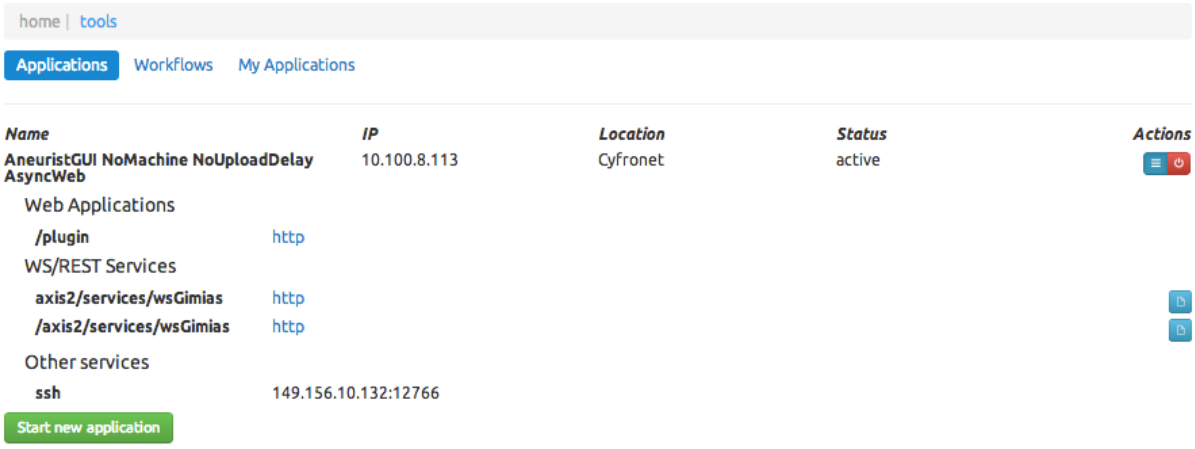


Figure 31. Invocation endpoints for an active application

In this case the sample service provides access through SSH (for administrative purposes) as well as an HTTP redirection upon which a Web Application endpoint has been configured. Clicking on the displayed link will take the user directly to the web application, enabling to start using the service.

If the service provides a non-Web interface - for example a remote desktop - the user will need a client appropriate for the type of interface in use (e.g. VNC/NoMachine). Some services only provide programmatic access via APIs such as SOAP or REST, in which case the service is intended to serve as a backend for other services. While it may still be useful to instantiate such services in the Generic Invoker, no end-user interface is available. In any case, the detail page will provide the user with details on how to access any interface endpoint provided by your instance.

2.3.6 Resources: workflows

2.3.6.1 Upload a new workflow

The user can use VPH-Share infostructure to upload and share new workflows.

To do so, from the Workflows page from the homepage, an upload section is available for a new workflow creation. When pressed, this will open the form, which allows inserting all the necessary information to completely define a workflow resource.

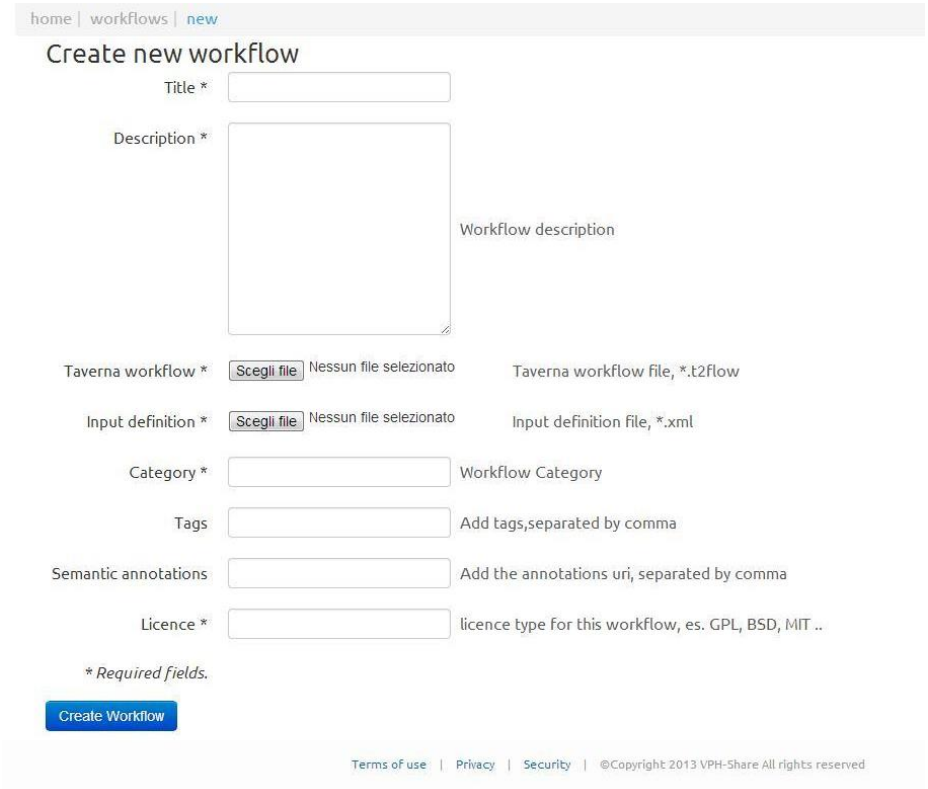


Figure 32. New workflow upload form

The mandatory fields are:

- Title: the name with which the workflow will appear in the available workflows list.
- Description: a summary of what the workflow does, its input, its output, and any other information useful for others to understand it.
- Taverna workflow: the workflow description file in Taverna Workbench format.
- Input definition: the input of the workflow as saved by Taverna Workbench format.
- Category: standard categories for workflows defined by the domain experts.
- Licence: the level of permission associated to the workflow access/use (i.e. GPL, BSD, etc.).

The user can eventually add also Tags and Semantic annotation URI (both separated by commas), which can be then used by search services to help user looking for specific services to retrieve the new workflow.

2.3.6.2 Run an existing workflow

A workflow is the choreography of a series of components with a certain research aim. Workflows can be created by composing Applications with Taverna Workbench, and uploaded and shared in the Master Interface as previously described.

If a user has got permissions to access a workflow, he/she can download the workflow description file and its inputs so to execute the workflow locally from Taverna Workbench. Otherwise he/she can use the *Execute workflow* button to start its run (connecting to the workflow manager technically described in section 5.4).

Once the button is pressed a window opens to allow the user to configure the start of the workflow. Some of the parameters can be used to run the workflow with custom input files or by using development mode Applications, while others (like the selection of the Taverna Server) are temporarily present for debugging. More work will be done in the future to provide an interface to customise the inputs of the workflow and in particular for the update/addition/deletion of parameters.

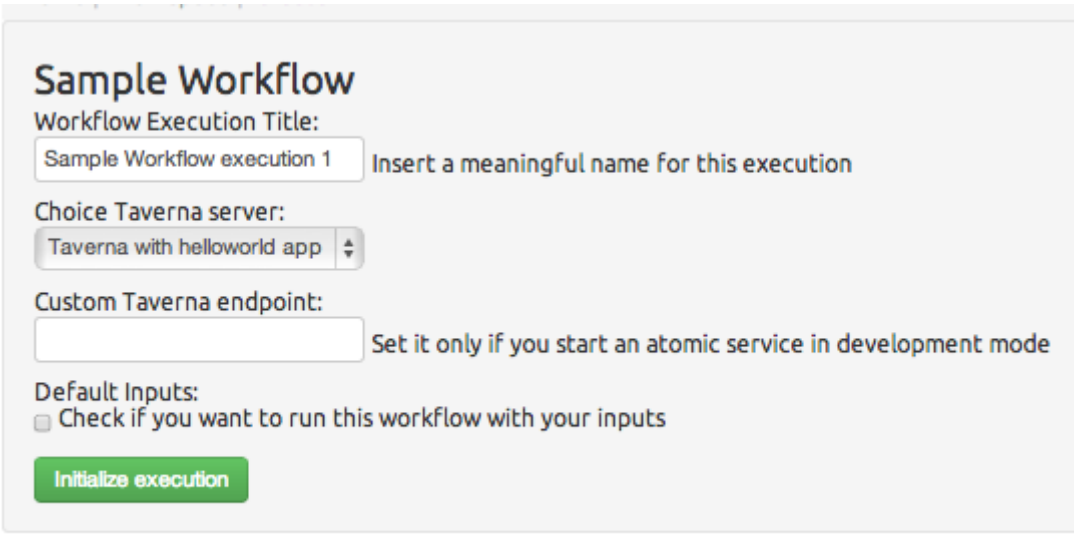


Figure 33. Workflow run configuration

When the *Initialise execution* is pressed, the workflow is added to the list of those ready for execution. Each workflow has three action icons, a green arrow to start the execution, a red cross to delete the workflow execution, and an orange icon to see the logs associated to the specific workflow execution (in green the actions successfully completed and in red the errors).

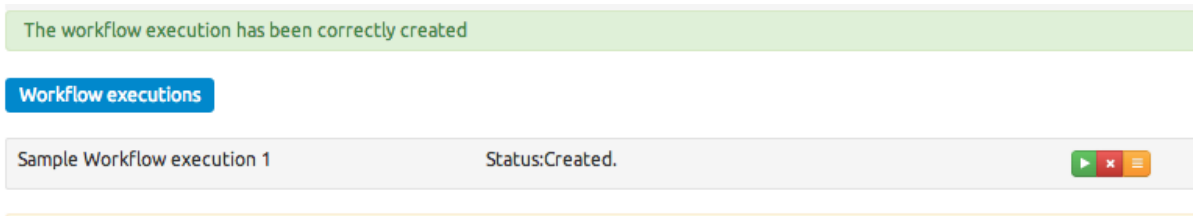


Figure 34. Workflow execution list

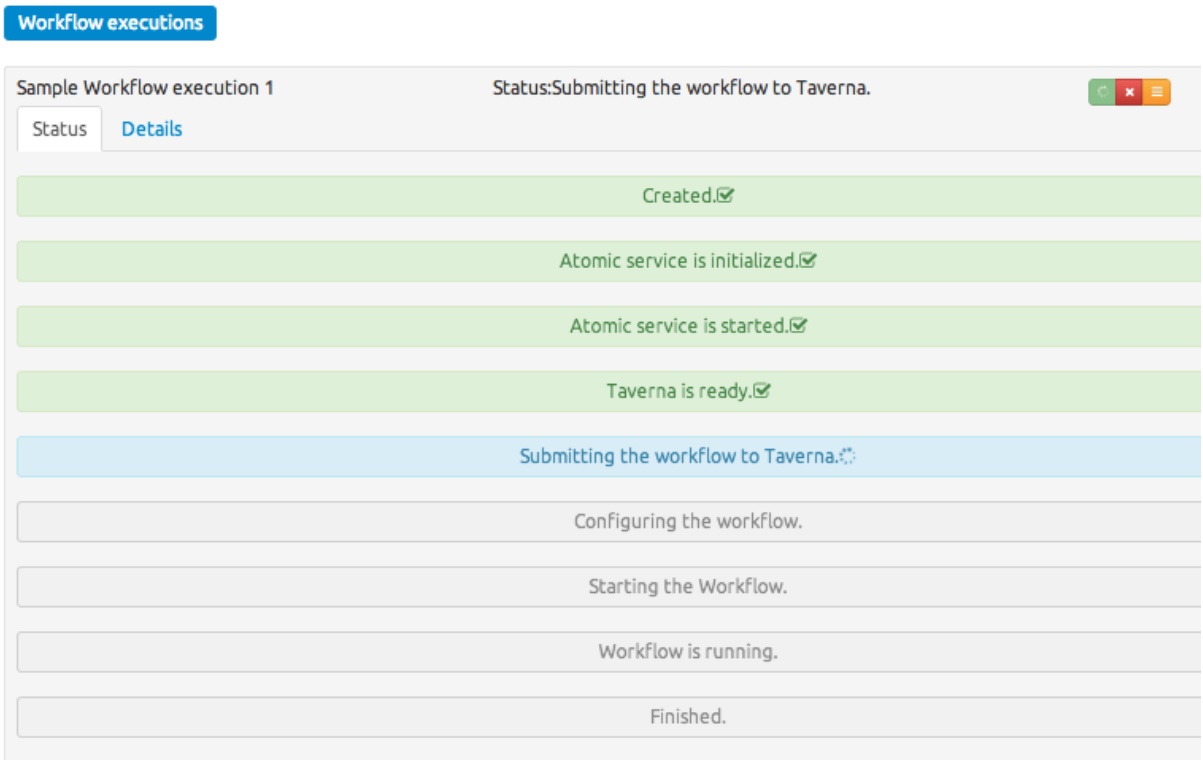


Figure 35.Workflow logs during execution

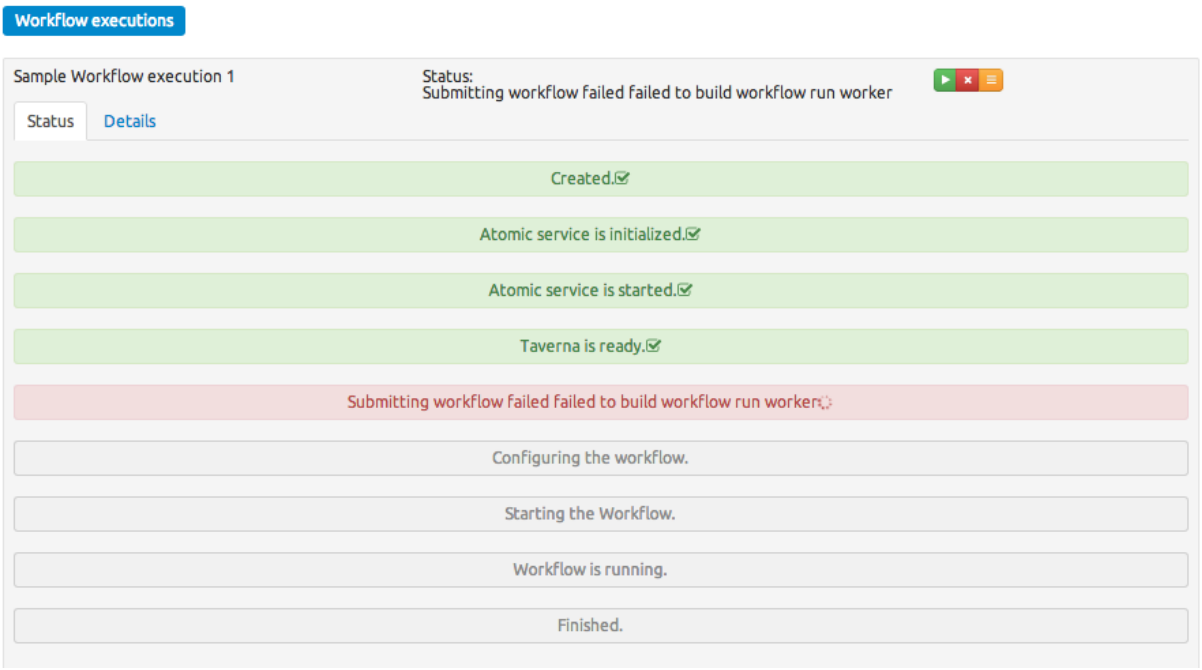


Figure 36. Workflow execution error reporting

If the workflow generates binary data during or after its execution the data can be found into the LOBDCER folder name with the same workflow Id.

2.3.7 Manage owned resources

2.3.7.1 The dashboard

The dashboard is the place in the VPH-Share web portal where the user can access all the owned/managed resources: data, applications, and workflows. The user can be assigned to be manager of a resource (and thus have it in his/her dashboard) even if he/she is not the resource owner.

Figure 37. Dashboard

The page has two sections, one for each resource type, and each section can be expanded or collapsed with the right side arrow.

All the resources are presented in the same way, so only one description is provided here. For each of the resources, the user can see the title/name of the created resource, its author (with its username), and the date the resource was published.

Three actions are possible for each resource with the two blue and orange icons: visualise the resource information, edit the resource tags, and change the access permissions.

1. Resource information (blue icon), which provides the visualisation of the resource common metadata as presented in the Resource Access section 2.3.3.4;
2. Edit tags (orange icon, details tab), once this is clicked, a panel to edit the resource information is opened.

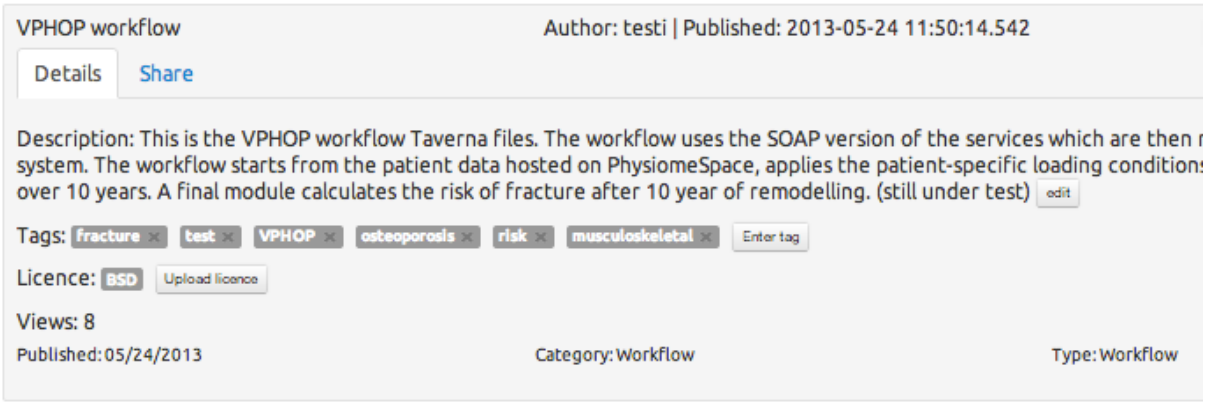


Figure 38. Edit tags

The description can be modified by clicking on the *edit* button. The Tags can be removed or new ones can be added by using the *Enter tag* button. Also the licence

associated to the resource can be changed using the *Upload licence* button. Other information like the title or the category of the resource cannot be modified.

3. Change access permissions (orange icon, Share tab), the permissions associated to each resource can be edited by the owner and the managers. If there are pending requests to be checked, the user will be warned by a red exclamation mark close to the icon.

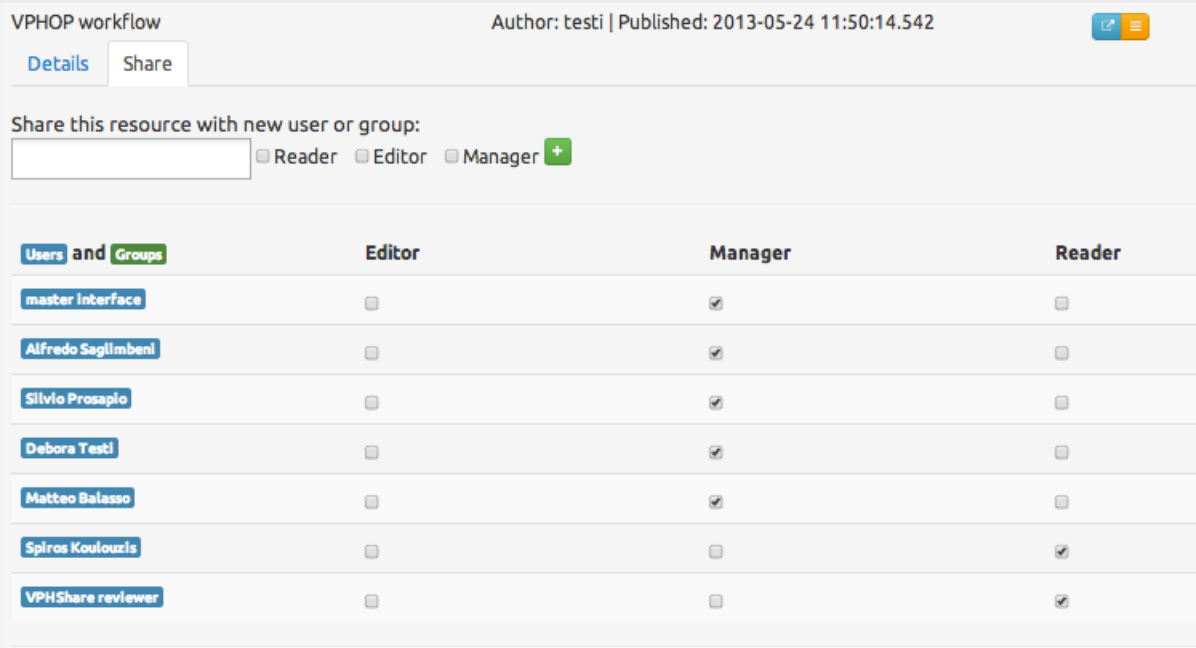


Figure 39. Manage requests

Using the available page, the resource owner can search for a user/group to be granted access in the search box. The list of matching users will appear at the bottom.

To add permissions, the resource owner has to click on the role he/she would like to assign (Editor, who can change the resource information, Manager who can also modify the permissions, Reader who can just download but nor modify the resource). If the user wants to revoke permissions, he/she simply has to click again on a respective box and the permissions will be revoked.

2.3.7.2 The workspace

The workspace is the place where the user can:

- Authoring and execute workflows
- Access to workflows outcomes
- Review history of workflow executions,
- Create and manage new workflow composition and execution thanks to the Taverna Online integration (see technical information in section 4.4.2).

Although the MI itself offer most of the components of the workspace, our collaboration with the IITP.ru to integrate web workflows composition services have bring us the opportunity to provide a more integrated workspace to VPH-Share users at the same time that a great use case to demonstrate how an external application can integrate VPH-Services and components. However this is still a work in progress that will be completed early in Year 4.

2.3.8 User care

Being WP6, the work package devoted to the user interfaces development; its activities in the last part of the project are mostly driven by the user's feedbacks and in synergy with WP8.

As engagement with end-users is particular relevant for VPH-Share as a whole we particularly care about supporting users and take into proper consideration feedbacks coming from the testing phases. As reported in WP8 beta user programme activities, so far the system has been opened to the project partners, but a public release is being launched soon.

For this reason, a number of tools to get in contact with beta users have been deployed and proper procedure to process the feedbacks put into a place.

After a first evaluation of the getSatisfaction tool, the consortium has decided that this was not completely satisfying the project needs as the direct interaction between the users and the development team was not straightforward. Thus, a new procedure has been put into place. For internal users a Redmine tracking system is in place, with two distinct sections for the users (exemplary workflows users) and the development team with the possibility to cross-link the issues and their tracking.

For external users, they are invited to write feedbacks or bug reporting to the support email; the email is processed by a single person (working on both WP6 and WP8) who takes care of clarifying unclear aspects, replicating the issue and discussing with the user further details. Once the issue is clear, the support person posts the issue on the Redmine system and assigns it to the proper member in the development team. As soon as a solution is provided to the issue, the user is notified by email.

A list of already notified/known issues will be also kept on the MI web pages (immediately after public release of the beta version to the general public) together with a release note with the last development exposed into the interface.

3 VPH-SHARE CLOUD SERVICES

3.1 Final architecture and API

Several views of the Master Interface, in particular associated to the Application resources, are cloud management graphical user interfaces (GUIs), which communicate with WP2 cloud components directly or through the Cloud Façade API module. In Figure 40 a dependency overview among the components is presented followed by a short summary of core functionalities.

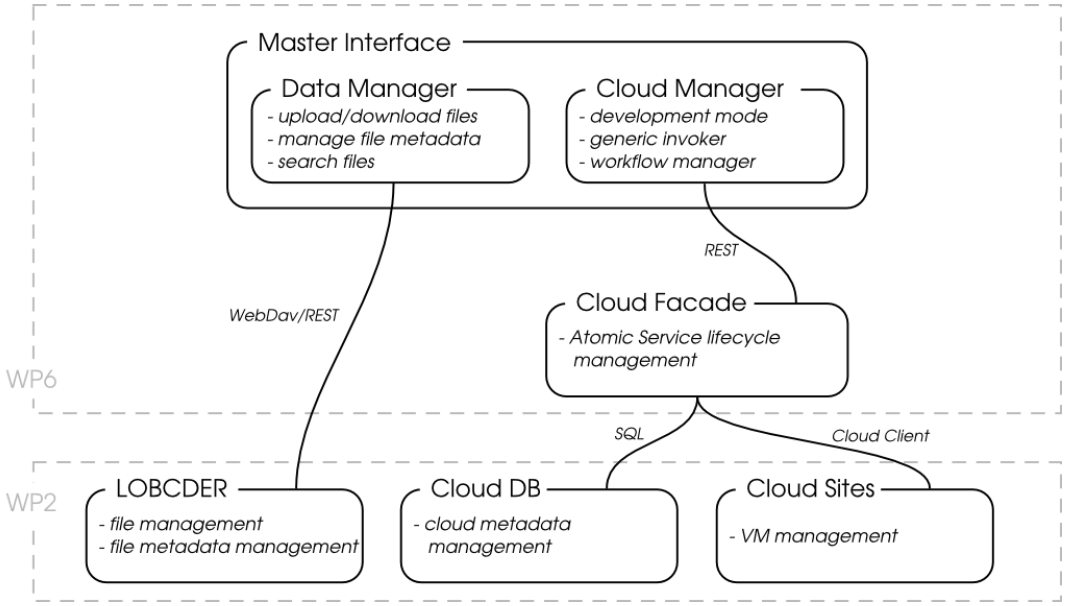


Figure 40. Architecture of cloud services and dependent components

Data Manager, which is used for managing file resources interfaces with the LOBCDER component directly by using a mixture of WebDAV[9] and REST[10] invocations. The user is able to view, upload and download files, as well as, edit their metadata. Because LOBCDER offers a standard WebDAV interface many available clients can be used to access its resources. The REST part of the interface is used to manage more complex metadata queries. Authentication is implemented by delegating a user token obtained from the Master Interface server with each request.

Cloud manager uses the API offered by the Cloud Façade component, which delivers a complete set of operations managing the life cycle of Appliances by utilising Cloud DB and Cloud Site components of the WP2 work package. The façade is also intended to be used by other clients such as workflow management systems to provision resources required for workflow execution. However, in the scope of the Cloud Manager it is used to enable users with the possibility to manage individual instances of Applications and test them in the development mode.

[9] http://www.webdav.org – WebDAV Resources
[10] http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm - Representational State Transfer

Deployment of cloud components is presented in Figure 41. All the cloud management views and Cloud Façade request handlers were encapsulated in a single Cloud Library run in the User Browser were necessary authentication credentials are obtained from the Token Store and delegated during direct communication with Cloud Façade. Such setup allows bypassing the MI Server for cloud requests and greatly improving the responsiveness of cloud management views. The Cloud Library itself is served by the MI server.

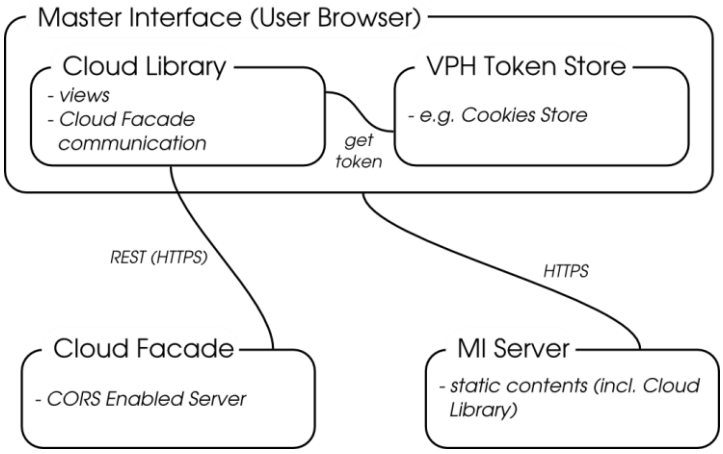


Figure 41. CORS-based cloud component deployment

In order for the direct communication between the Cloud Library and the Cloud Façade component to work a CORS (Cross-Origin Resource Sharing) mechanism had to be used as the MI Server and Cloud Façade servers are hosted by using different domains. Additionally, token authentication is used with each request to ensure proper resource access.

Cloud Façade component offers a REST interface for managing cloud resources. The full API description is available online. The API is divided into several sections, each managing different aspects of the cloud infrastructure. Here, for reference, the part responsible for obtaining information about compute sites is presented. The available operations include obtaining information about all compute sites or about a specific one:

Operation

```
GET /compute_sites
```

Response

```
{
  "compute_sites": [
    {
      "id": 1,
      "site_id":"cyfronet-folsom",
      "name":"Cyfronet",
      "location":"Cracow",
      "site_type":"private",
      "technology":"openstack",
      "config": ""
    }, {
      ...
    }
  ]
}
```

Operation `GET /compute_sites/{id}`

Response

```
{
  "compute_site":
  {
    "id": 1,
    "site_id":"cyfronet-folsom",
    "name":"Cyfronet",
    "location":"Cracow",
    "site_type":"private",
    "technology":"openstack",
    "config": ""
  }
}
```

For each of the operations description, URL and request/response bodies are given to easily implement clients. Authentication is done by passing a valid token obtained from the Master Interface (see section 3.1.6 for details). Other sections of the API include the following:

- Application Configuration Instances – concrete application configurations with property placeholders replaced and with contents which is injected into running machines at boot time,
- Application Configuration Templates – configuration templates holding property placeholders for a given application,
- Application Endpoints – provides a list of application types with their endpoints,
- Application Sets – groups applications for a given user, can be of type portal, workflow or development which changes the behaviour of optimisation mechanisms,
- Application Types – describes applications including properties such as visibility, sharing policy, scalability or preferred resources,
- Application – represents a list of running instances,
- Compute Sites – list of compute sites available in the infrastructure,
- Development Mode Property Sets – a structure with appliance type properties being a copy of the Application Type structure needed for instances run in the development mode,
- Endpoints – list of endpoints representing HTTP based applications offered by a given application type,
- HTTP Mappings – represents a list of HTTP redirections for applications running on private networks,
- Port Mapping Properties – list of port mapping properties allowing for setting attributes such as timeouts,
- Port Mapping Templates – a structure describing all necessary port mappings for a given application type,
- Port Mappings – concrete port mappings containing source and destination port numbers for a running application instance,
- Security Policies – repository for storing security policies used by the security proxy component while authorising requests,
- Security Proxies – holds configuration parameters for security proxies,
- User Keys – user public keys injected into running application instances,

- Users – list of users using the cloud infrastructure,
- Virtual Machines – list of virtual machines working for a given application.

3.2 Overview of Cloud Management User Interface

Cloud Management GUI offers graphical user interfaces to manage the life cycle of Application instances run on the cloud. It works in three different modes, which are shortly summarised below.

- *Development mode* – in this mode a developer can instantiate both starting templates and existing Applications to perform development tasks leading to the creation of new Applications, in this mode no optimisation is used and each instance of an Application has a corresponding physical machine running.
- *Generic invoker mode* – this mode is intended for users who would like to instantiate individual Application instances and utilise their functionality, in this mode platform optimisation is used which can result in instance reusing.
- *Workflow mode* – this mode is used to view all started Application instances (this does not include instances run in development and generic invoker modes) on behalf of a given user, which can be a result of using an external workflow management system.

The sections below contain screenshots with descriptions for each of the modes for completeness (some of these were already presented from the user perspective in section 3.2.5). The main view of the cloud management interface consists of three tabs corresponding to each mode as presented in Figure 42.

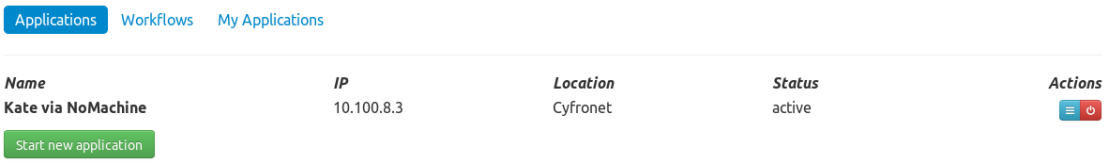


Figure 42. Main view of the cloud management GUI divided into three tabs corresponding to different working modes.

3.2.1 Development Mode

The development mode view (denoted as My Applications tab) is divided into two sections (see Figure 43). The first section contains all the applications owned by a given user and allows changing its properties and associated initial configurations. It is also possible to remove an Application, which will also remove the image snapshot stored in the cloud infrastructure. The set of properties for an Application include name, description shared and scalable flags and the visibility, which can be limited to the owner, to users granted the developer role or to regular users. Several initial configurations can be defined for a single Application and each can contain parameter placeholders, which are replaced by user provided values during booting up new instances.

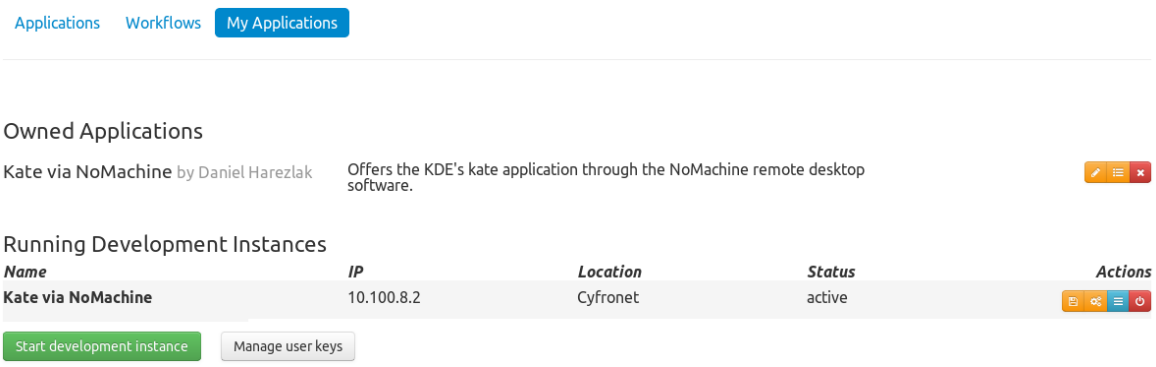


Figure 43. Development mode view containing a list of applications owned by a given user and a list of running development instances.

The second section is a list of running development instances with details about the location and current status. Also, redirections and available endpoints can be edited for each of the items in this list. After the item is expanded all the available redirections and endpoints for a given instance are presented which allows for testing the exposed services directly in development mode.

The development view also has two action buttons, which are used to spawn new development instances and managing user keys. Starting a new development instance ensures that always a fresh instance is run without any virtual machine reusing. If more than one user key is added during booting a new instance a choice dialog is presented to pick one of them.

To register applications exposed a developer has to specify any number of the following endpoint types:

- *Web Application Endpoint* – an endpoint pointing to a resource returning a web page,
- *Web Service Endpoint* – an endpoint pointing to a Web Service resource, for this endpoint a WSDL description should be provided, optionally a description of the service can be provided.
- *REST Endpoint* – an endpoint pointing to a REST resource, optionally a WADL descriptor can be provided.

Each of the endpoint should have a port number assigned, which is used when setting up redirections by the platform after an Application instance has started. The endpoints are defined in the development mode and therefore can be instantly tested by developers. According to the information provided in endpoint definitions the platform is able to produce valid URL and port mappings during runtime to target appropriate cloud instances.

3.2.2 Generic Invoker

In the generic invoker mode (Applications tab in the main view) regular users can instantiate and user available Applications. A sample view of this mode is presented in Figure 44.

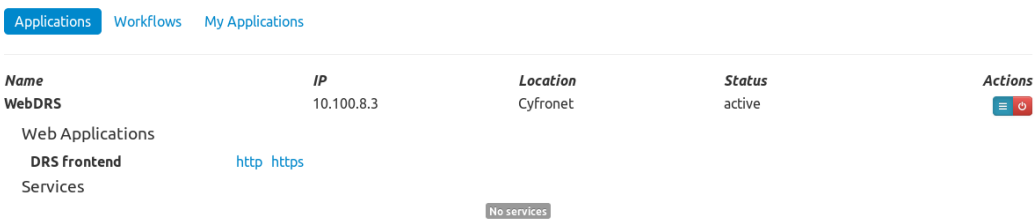


Figure 44. Generic invoker view with a list of Appliance instances.

The details for each instance show the site where the instance is running and its current state. After the instance becomes active all defined redirections are configured and the available endpoints are listed for easy access. In the presented example the WebDRS Application exposes a web application available through http or https channels. If the application is accessed by using the presented links inside the generic invoker view automatically a security token is attached to the request which is processed by the security proxy on the instance.

3.2.3 External Workflows

The third view, which corresponds to the workflow mode, allows for managing Application instances run from an external service (e.g. workflow management system such as Taverna). It is possible to see if any instances were run on behalf of a given user and if necessary remove them. The view also gives details about the instances similar to the generic invoker view.

3.3 LOBCDER repository

Files present in the cloud storage resources (WP2 LOBCDER component) can be managed from the Data Manager view. File upload and download is possible by using the WebDAV protocol (hidden from users by the data browser). The resources are structured into directories to resemble the standard file system structure. A sample view of the data browser is presented in Figure 29 and described in section 3.2.4.2.

Originally, a flat file structure view was supported by the Data Manager; however, a requirement for structured data emerged to deal with the amount of data which eventually will be stored. Structure of directories similar to standard file systems is used and the browser presents the contents of one of the directories at a time starting with the root location. The upload action always uploads a given file to the currently selected directory.

As in a standard file system removal of LOBCDER resources is possible. The removal action can be applied to both files and directories. In case of a directory all its contents (contained files and subdirectories) are recursively removed from the storage. The action respects the permissions set on files and directories so only resources owned by a given user can be removed. The user credentials are delegated down to the LOBCDER service with each request to be authorised.

Each of the files stored in LOBCDER can be annotated by a number of metadata properties. The metadata engine implemented within LOBCDER offers an extendible API to support new properties. The properties can be viewed and set (if they are writable) from the browser

by picking one of the files and going to the metadata view. Metadata are managed by a dedicated REST service exposed by the LOBCDER component, which is completely hidden from the end users by the browser. The service also enables for requesting search queries on the metadata. In such case the list of LOBCDER resources is filtered independently of the directory structure so all the files are always examined during searching.

3.4 Remote Desktop Access

To cover use cases where desktop-based GUI applications are part of workflow execution in VPH-Share, a mechanism to support remote desktop access was implemented. To make this even more convenient for end users a web-based client was used to handle such scenarios. The current implementation allows creating Applications, which offer accessing native applications from the browser by clicking a generated link. The communication is based on the SSH channel and the client is configured on the server side each time the user accesses the machine. As the remote desktop technology NX NoMachine software was used. The client is available for all major platforms and through the Web Companion web applet appropriate version is downloaded and executed. The end user experience is such that the remote application is run locally with a little slower response times (dependent on the network throughput). This is possible due to the fact that only a given application window is transferred over the network without the necessity to show the whole desktop as with other such technologies.

The main difficulty overcome by this integration effort was to handle the dynamic nature of redirections of instances run in the cloud. Each time an instance is started different redirection ports are assigned to the SSH channel (which by default is executed on port 22). The combination of using the Cloud Façade API and passing of the current properties through REST services deployed on the machine instance allowed properly configuring all the components.

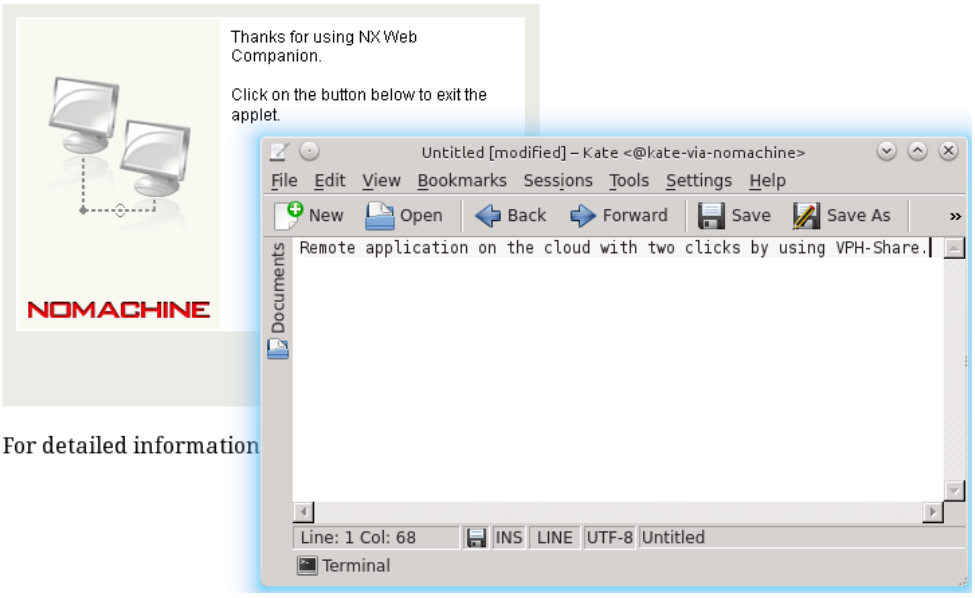


Figure 45. Sample remote application run on the cloud visible on the user computer as yet another local window

In Figure 45, a sample application was executed which presents itself as another locally executed window. If a file would be saved in this application it would be stored on the machine instance run in the cloud. This also makes it possible to save the results of running such application in LOBCDER (cloud storage component) given it is mounted on the cloud instance through the WebDAV driver. This use case makes it very easy to use cloud resources (both computation and storage) in a seamless way.

3.5 Web Service Catalogue

One of the services provided by the Cloud Façade component is a list of the endpoints provided by the available applications. It is possible to limit the number of applications types and endpoints returned by specifying the endpoint type or giving endpoint identification numbers. The specification of the operation is as follows:

Operation

```
GET /appliance_endpoints
GET /appliance_endpoints?endpoint_type=ws (rest or webapp)
GET /appliance_endpoints?endpoint_type=ws,rest
GET /appliance_endpoints?endpoint_id=1,3,7
```

Response

```
{
  "appliance_endpoints": [
    {
      "id": 1,
      "name": "Foobar Appliance Type",
      "description": "Foobar Appliance Type description",
      "endpoints": [
        {
          "id": 1,
          "name": "name of the endpoint",
          "description": "some descriptive text",
          "endpoint_type": "ws", ("rest" or "webapp")
          "url": "url_to_descriptor"
        }, ...
      ]
    }, {
      ...
    }
  ]
}
```

This operation is particularly used by the workflow composition tools Taverna Workbench and Taverna On-line, to present the user with a Web Service Catalogue, a user-friendly list of the available web services offered by VPH-Share applications.

In Taverna On-line this catalogue has been incorporated in the form of a dropdown list, see Figure 51. When the user presses the ‘Get Operations’ button, the list of web services is displayed and then the user can browse through the services and choose the one to be used for workflow composition.

In Taverna Workbench it is not possible to browse the catalogue, but when the user imports a VPH-Share service using the VPH-Share plugin, as explained in http://vph-share.eu/content/running-aneuristworkflow-short-workflow, the plugin accesses the catalogue

to retrieve the name of the applications that corresponds to the service being added, so as to show the application name instead of the URL of the service in Taverna Workbench's GUI. For this point on, the user can more readily identify the applications and web services in the GUI.

4 WORKFLOWS COMPOSITION, INTEGRATION AND EXECUTION

During the VPH-Share project several tools have been developed to compose, deploy and execute biomedical workflows. Most of the work has been done behind the scenes to provide the user with friendly interfaces and reliable tools that are able to run smoothly through the VPH-Share infostructure. The following sections describe the developed tools in detail.

4.1 Final architecture

This section provides a brief review of the architecture, leaving the more detailed description for the following sections.

The final architecture for workflow composition, integration and execution is composed of client-side and server-side components, see Figure 46.

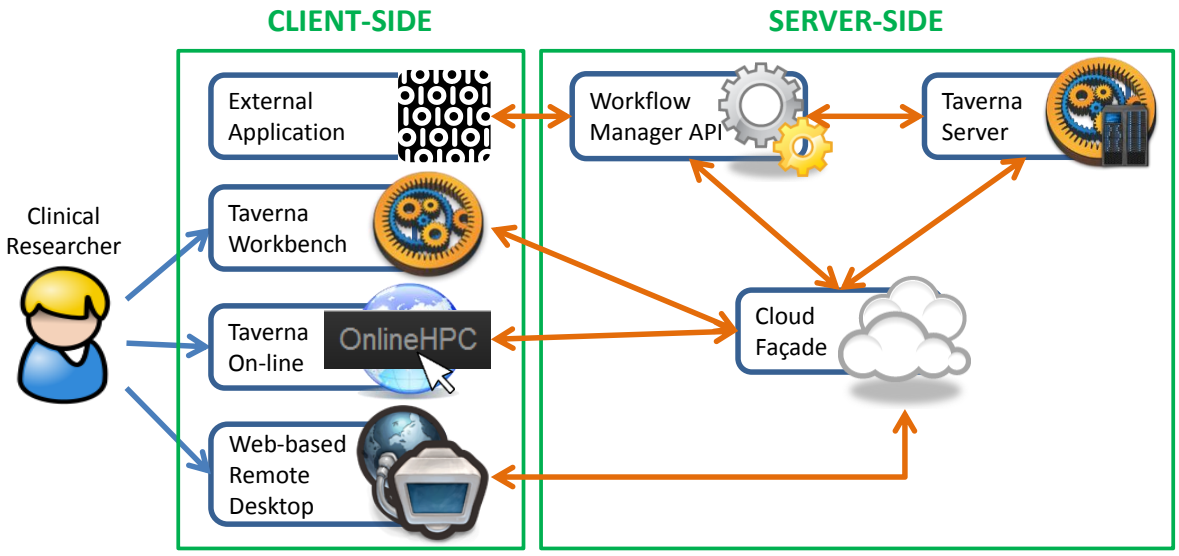


Figure 46. Workflow Management Architecture overview.

On the client-side, a biomedical workflow can be composed and executed by a clinical researcher using two different applications: Taverna Workbench and Taverna On-line. For this purpose, the VPH-Share plugin has been integrated into these two platforms. Taverna Workbench is used for desktop composition and Taverna On-line is used for web composition. If the executing workflow requires user interaction, the user is able to perform such interaction via NoMachine, a web-based remote desktop application, which connects to the executing workflow.

On the server-side, the Cloud Façade is the platform where the core of the biomedical workflow is executed. This execution is controlled by exchanging information with the VPH-Share plugin integrated into Taverna Workbench or Taverna On-line. If a biomedical workflow requires interaction, a server-side component inside the Cloud Façade is started to make the remote desktop communication possible.

Additionally, a biomedical workflow can be executed directly from the Master Interface using the Workflow Manager. In this case the Cloud Façade will start its own Taverna Server, with an integrated VPH-Share plugin, and then submit the biomedical workflow to the server for execution.

Finally, an external application on the client-side can also start the execution of an already composed workflow. In such a case, the application communicates with the Cloud Façade through the Workflow Manager API. The Cloud Façade will then start its own Taverna Server, with an integrated VPH-Share plugin, and then submit the biomedical workflow to the server for execution.

In all cases, the VPH-Share plugin provides support for executing biomedical workflows with interactive and non-interactive services, as well as the execution of workflows in batch mode. The results of the execution are accessible though the LOBCDER repository, see Section 3.3.

With the development of all the aforementioned processes and tools, the VPH-Share project provides the Clinical Researcher with a very versatile platform for execution and composition of biomedical workflows.

4.2 GIMIAS WebServices plugin to make CLP tools available as Web Services

The main goal of the GIMIAS WebServices plugin is to offer a mechanism to expose external tools (command line tools) as web services (SOAP), facilitating the integration of these tools on research workflows, providing also an interface to support tools with long execution times.

The GIMIAS WebServices Plugin, part of GIMIAS's extensions, is able to expose as a web service any processor of any GIMIAS Plugin or Command Line Plugin (CLP). A list of available Plugins and CLPs is available at http://sourceforge.net/apps/mediawiki/gimias/index.php?title=Users, and instructions on how to create new CLPs can be found at http://sourceforge.net/apps/mediawiki/gimias/index.php?title=HowToAddCommandLinePlugin.

The user just needs to activate in GIMIAS those plugins that are to be exposed and then activate the WebServices Plugin. Then, the WSDL generated by the WebServices Plugin can be used to reach the exposed Web Services.

In the VPH-Share project GIMIAS is used as a server that acts as a Web Services provider. Those services are used to compose biomedical workflows in different fields (@neurIST,

euHeart, VPH-OP and Virolab flagship workflows). The composition and execution of these workflows can be done using Taverna Workbench or Taverna On-line.

Also, any CLP already exposed through GIMAS can take benefit of the improved WSDL interface to support execution with long times, in combination or not with the VPH-Share Taverna Plugin.

The list of Web Services related to the VPH-Share flagship workflows currently deployed includes:

- @neurIST services not requiring user interaction:
  - GAR segmentation
  - Geometric Descriptors computation
- @neurIST services requiring user interaction:
  - Bounding Box selection
  - Mesh editing
  - Ring cut
  - Neck selection

Other Web Services that are available (not directly related to any of the VPH-Share flagship workflows) include:

- Basic Data Visualisation Capabilities
- euHeart services:
  - Cardiac Initialisation
  - Cardiac Fitting
- Clinical Report Creation
- Other basic segmentation tools
  - Otsu Segmentation
  - Thresholding Segmentation
  - Region growing Segmentation

Currently GIMIAS provides a great interface to easily deploy any new external service on the VPH-Share infostructure. Interactive and non-interactive services can be provided using the VPH-Share Taverna Plugin, both locally and on-line, via the two aforementioned Workflow Management Systems.

4.3 Specification of services requiring user interaction

If an Appliance is to expose a service that requires user interaction, this needs to be advertised for the VPH-Share Taverna Plugin to be able to provide the support for such interaction. This can be easily configured by the Appliance developer through the Master Interface when adding the Web Service Endpoint of the Appliance in Development Mode.

The configuration consists on adding a string containing the list of interactive services to be exposed by the endpoint as part of the description field of the endpoint. The string must start with the 'INTERACTIVE_SERVICES=' (without quotes) and then the list of interactive services

names must follow, with each service name separated from the other by a coma. For instance, one of the appliances employed in the @neurIST workflow specifies its interactive services as 'INTERACTIVE_SERVICES=MeshEditing, NeckSelection, RingCut, BoundingBox' (without quotes), corresponding to the interactive services mentioned in the previous section.

Note that this process is only performed by the developer user that creates the Appliance in the VPH-Share portal, and it is only performed once, before the Appliance is saved. Any regular user of the Appliance does not have to perform any configuration task.

The INTERACTIVE_SERVICES string is automatically read by the VPH-Share plugin during workflow execution, and when an interactive service is to be executed, the plugin detects it and provides remote desktop access to it, as explained in the next section.

4.4 VPH-Share plugin

The main goal of the VPH-Share plugin is to facilitate integration of Web Services deployed on VPH-Share on scientific workflows, supporting for both composition and execution, enabling VPH-Share Web Services to be instantiated and released on demand.

In the VPH-Share project, a GIMIAS server is installed in a Virtual Machine (VM) that can be instantiated as often as needed and shutdown on demand. Each VM is called an Application and it is managed by the Cloud Façade. Many different Applications can be created exposing different sorts of services. In order to make the services available to several users at a time, several Appliances can be used at the same time.

A user can access the resources provided by the Cloud Façade by using the VPH-Share Plugin to create and/or execute a biomedical workflow. The VPH-Share Plugin integrates with Taverna Workbench for desktop workflow composition and execution; and with Taverna On-line for web-based workflow composition and execution.

When a user wants to use a service provided by the Cloud Façade in a workflow, a set of service definitions in WSDL format must be imported into the Workflow Management System being used. While normally the entries of the WSDL would contain references to a running Web Service server (endpoint), in the VPH-Share project the server is not yet running, for it corresponds to a VM. Instead, the WSDL contains an identifier that indicates the VPH-Share Plugin, which VM to instantiate and which service to execute on that machine. The URL for the WSDL of a given Appliance can be obtained from the Master Interface, using one of the resource access buttons.

When the user executes a biomedical workflow using the VPH-Share Plugin, all the complexity of the execution process is handled by the plugin in the background to make things easy for the user. The VPH-Share Plugin instantiates the needed Applications, waits for them to start up, redirects the Web Service calls to the correct Application, waits for each Application to finish its job and shuts it down when no longer needed, manages the authentication of the user in the Cloud Façade and handles possible errors during the whole process.

4.4.1 Desktop composition and execution tool

The architecture for desktop workflow composition and execution is presented in Figure 47. Currently, Taverna Workbench 2.4 is the main tool supported for this purpose; it is open-source and is licensed under GPL license version 2.1. This tool can be obtained from http://www.taverna.org.uk/download/workbench/2-4/.

The first step for a user to build a workflow using Taverna Workbench is to download and install the software. If the user wants to include services provided by the VPH-Share project, then the VPH-Share Taverna Plugin must be installed. For installing the plugin see Section 'Installing Taverna Plugin' at http://vph-share.eu/content/vph-share-taverna-plugin.

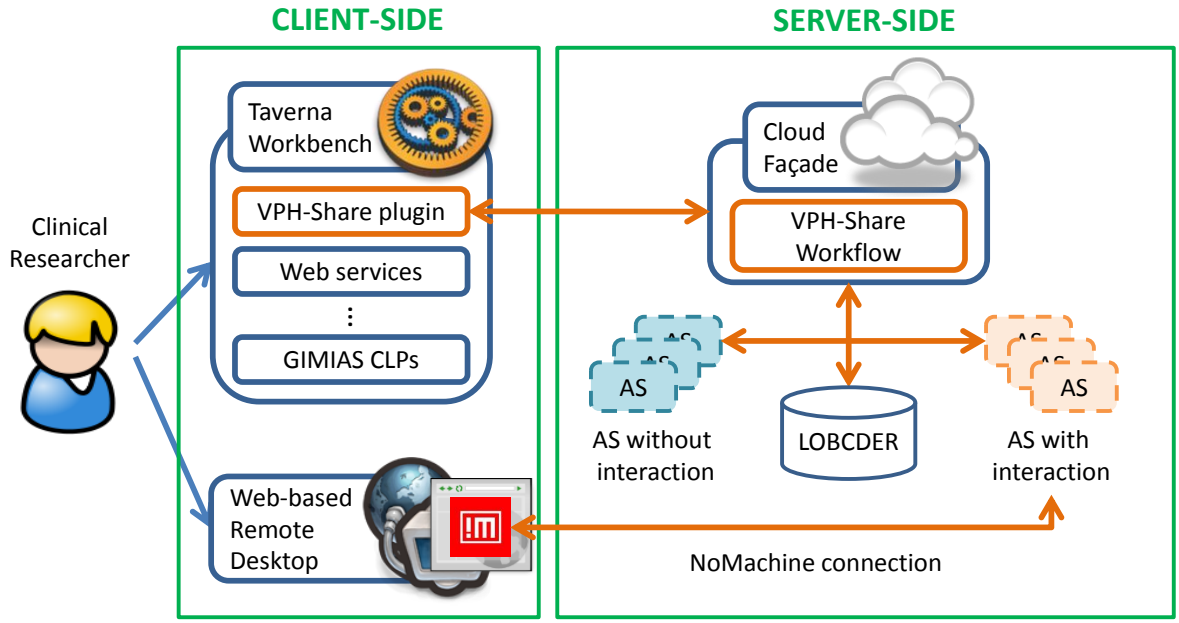


Figure 47. Desktop Workflow Management Architecture overview.

After this, the user can search the Master Interface for one or more Appliances that provide the required services. For each Appliance, the user can obtain the URL address of its WSDL, using one of the resource access buttons in the Master Interface. This WSDL address can then be used to import the services provided by the Appliance in Taverna Workbench, see Section 'Importing VPH-Share services' at http://vph-share.eu/content/vph-share-taverna-plugin.

As depicted in Figure 47, note that Taverna Workbench also supports other types of services, such as any generic Web Service, or the CLPs of a local GIMIAS installation. All these services can be combined by the user into one or more biomedical workflows. For a detailed explanation on how to build workflows with Taverna Workbench 2.4 see the user manual at http://dev.mygrid.org.uk/wiki/display/taverna/User+Manual.

When the user executes the workflow composed in Taverna, the VPH-Share plugin communicates with the Cloud Façade and creates a VPH-Share workflow, which includes all

the Applications needed to execute the Taverna workflow, see Figure 47. The execution of each service in the Taverna workflow is delayed until the plugin has made sure that the Application needed for this service is launched successfully. If two or more services from the same Application are being used, only one Application is created, and then the services share the Application, saving computation resources. The services are executed in the order specified by the Taverna workflow. The output of each service, as well as the final output of the Taverna workflow, is stored in the LOBCDER, see Figure 47.

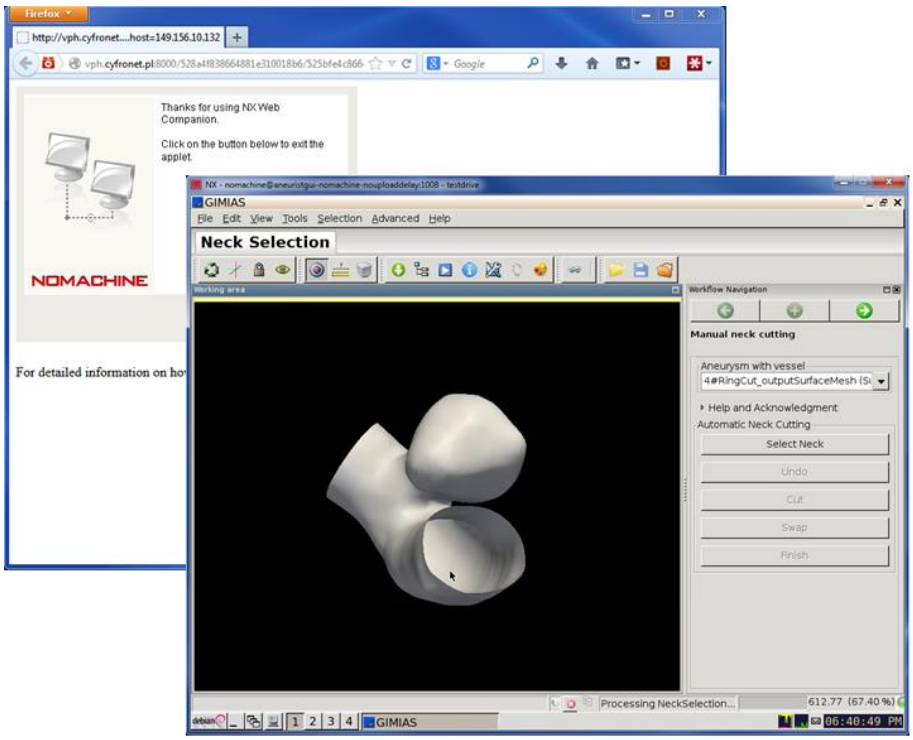


Figure 48. Web-based remote desktop connection via NX NoMachine.

If an Application requires user interaction, a web browser window will automatically open in the user's desktop when the service is executed, so that the user can perform the interaction. The browser will start a web-based remote desktop session via a NX NoMachine (https://www.nomachine.com/) client. The client is available for all major platforms. The NoMachine Web Companion java applet downloads and executes the appropriate version. The end user experience is such that the remote application is run locally with a little slower response times (dependent on the network throughput) than a normal PC, see Figure 48.

An explanation on how to run an example workflow can be found at http://vph-share.eu/content/running-aneuristworkflow-short-workflow.

4.4.2 Web composition and execution through Taverna Online

The architecture for web-based workflow composition and execution is presented in Figure 49. Currently, Taverna On-line is supported via the High Performance Computing Online (OnlineHPC) site. OnlineHPC's online scientific workflow editor is free-of-charge and it is available at http://onlinehpc.com.

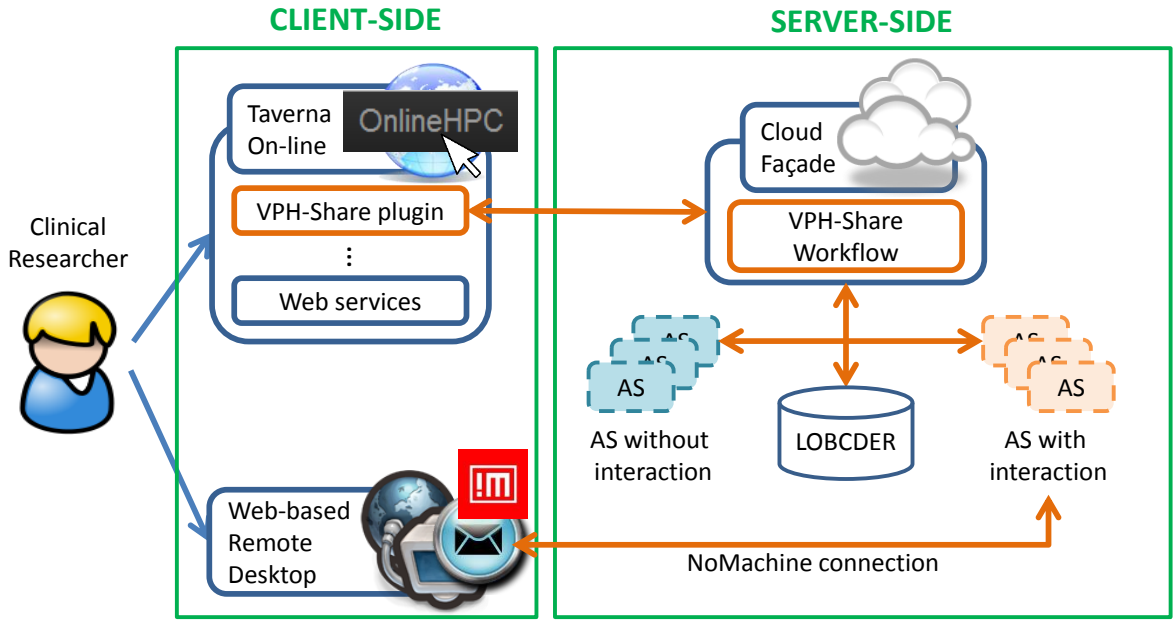


Figure 49. Web-based Workflow Management Architecture overview.

Once logged in OnlineHPC, the user just needs to press the 'New Workflow' button and name the new workflow. Then, the user will be presented with a working area as shown in Figure 50. If the user wants to include services provided by the VPH-Share project, it is only necessary to click on the 'VPHService' icon inside the 'Processors' box in the left side of the working area, and then drag-and-drop it in the working area. The result of this process is shown in Figure 50.

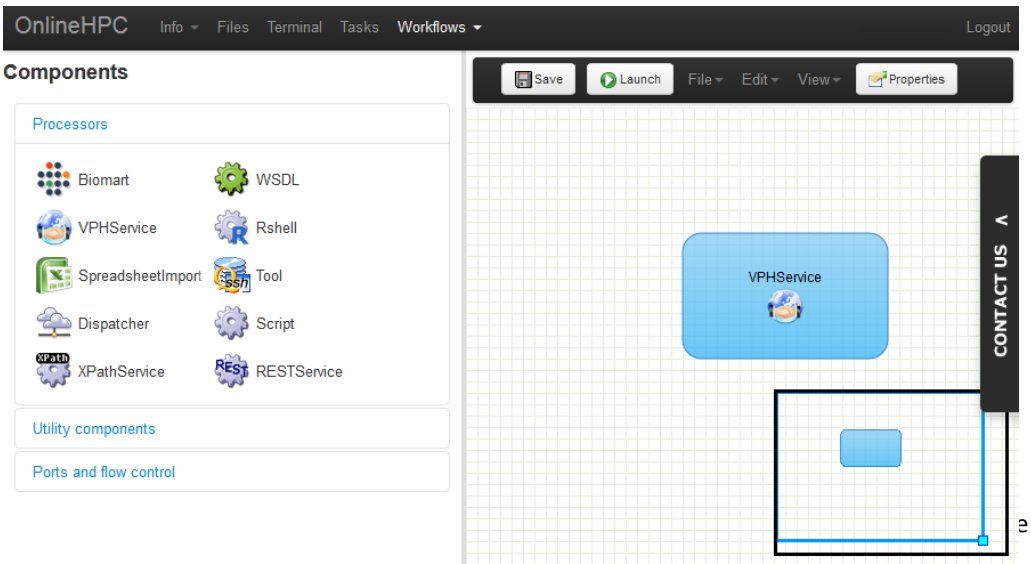
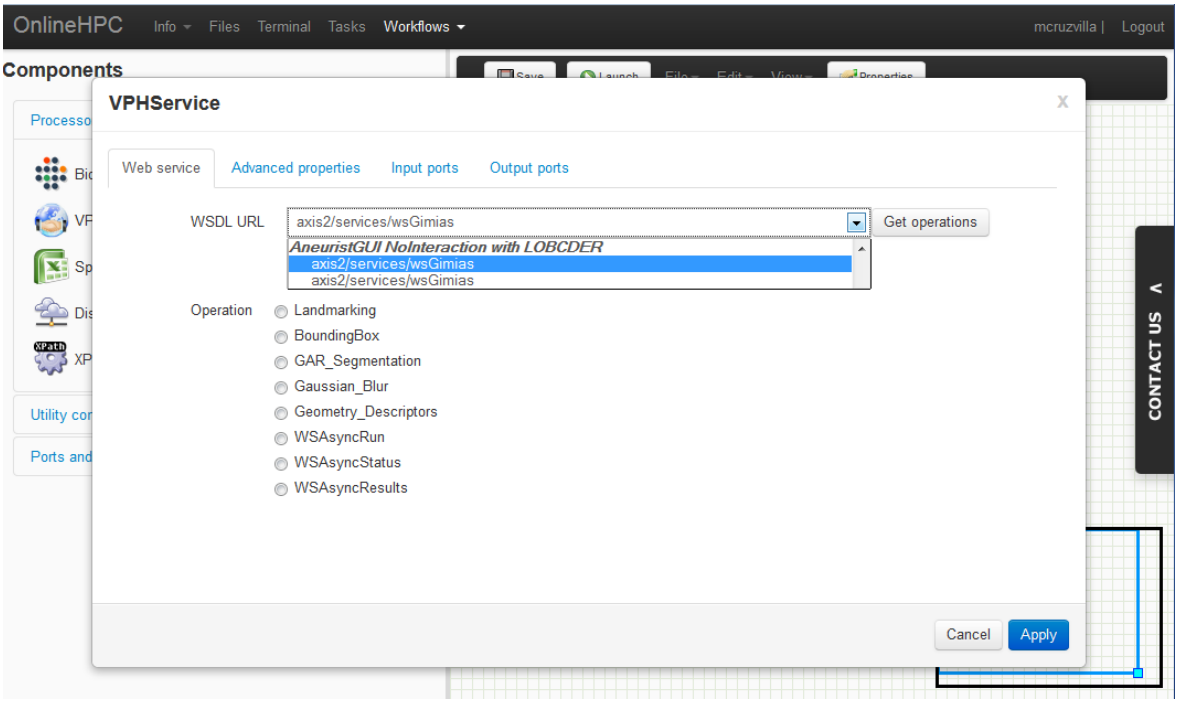


Figure 50. Taverna On-line working area.

Following this, the user must double-click on the VPHService blue box on the working area, and a configuration window will be presented. The first time this is done, the user must enter his Biomed Town credentials. Once this is done, the list of available Appliances will show up, and the user can choose an endpoint. Then, the user should press the 'Get Operations' button to retrieve the list of services available in the selected endpoint. Once the list of operations is displayed, the user can choose the service needed to build the workflow. See Figure 51 for an illustration of this process. For a guide on how to build workflows with OnlineHPC, see the video at http://www.youtube.com/watch?v=0n3YhJjPBy8.



When the user executes the workflow composed in Taverna On-line, the VPH-Share plugin communicates with the Cloud Façade and creates a VPH-Share workflow, which includes all the Applications needed to execute the Taverna workflow, see Figure 49. The execution of each service in the Taverna workflow is delayed until the Application needed for this service is launched successfully. If two or more services from the same Application are being used, only one Application is created, and then the services share it, saving computation resources. The services are executed in the order specified by the Taverna workflow. The output of each service as well as the final output of the Taverna workflow is stored in the LOBCDER, see Figure 49.

If an Application requires user interaction, the Master Interface notification service is used to send the user a link to the web-based NX NoMachine client, which can be used to open a remote desktop session to the Application. Notifications reach the user by e-mail as well as through the Master Interface's GUI, see Section 2.2.4. In the case of the later, the user can click on the link and then a new web browser tab will open, automatically starting the NX NoMachine client. Then the user can interact with the Application, see Figure 48.

A short video showing the web composition and execution process is available at https://dl.dropboxusercontent.com/u/5233146/tavernaOnlineIntegration_Sep13.avi.

4.4.3 Support for workflows with long execution times

By default, the execution of VPH-Share services from Taverna Workbench or Taverna On-line has a *blocking* behaviour. This means that when the VPH-Share plugin invokes a VPH-Share service, the plugin's execution is interrupted until the service returns a response. This is perfectly valid for services that return a response promptly, but it can pose a problem for those services with long execution times. The problem is that while the VPH-Share plugin is waiting for a response, the connection to the web service is not used, as no new information is communicated from either side. Then, the connection can be taken down by any intervening web proxies or firewalls, as the communication is flagged as *timed out*. This is the typical source of errors such as 'Bad gateway' or 'Timeout' when running VPH-Share services.

In order to avoid this, the VPH-Share Taverna plugin has been upgraded to support a *non-blocking* communication mechanism, in which the plugin starts the execution of the service in an asynchronous fashion, and then monitors the execution status of the service every few seconds. When the plugin detects that the service is finished, it collects the response and continues the execution of the workflow. This is implemented using the following GIMIAS CLPs:

- WSAsyncRun: This method can be used to start running a CLP asynchronously. The name of the CLP and its parameters must be specified. The method returns the identification number of the process that corresponds to the running CLP.
- WSAsyncStatus: This method can be used to inquire the execution status of a CLP previously started with WSAsyncRun. The process identification number of the CLP must be specified. Possible statuses are: 'STATE_PENDING', 'STATE_ACTIVE' and 'STATE_FINISHED'.
- WSAsyncResults: This method can be used to obtain the results of a CLP previously started with WSAsyncRun, once the WSAsyncStatus method returns 'STATE_FINISHED'. The process identification number of the CLP must be specified. The results will be returned in the form of a string.

In this way the VPH-Share Taverna plugin makes short and simple requests to the server every few seconds, avoiding leaving the communication up for long times, and therefore avoiding any timeouts.

This new mechanism can be easily activated in a per-service base, by selecting a service and then displaying its details. In the 'Details' tab, a 'Configure' button will show up. By pressing this button, the configuration dialog will appear, in which the user can activate the non-blocking behaviour by clicking on the check-box next to 'Execute service in non-blocking mode', see Figure 52. Moreover, this behaviour is automatically available for any CLP published using the GIMAS WebServices Plugin.

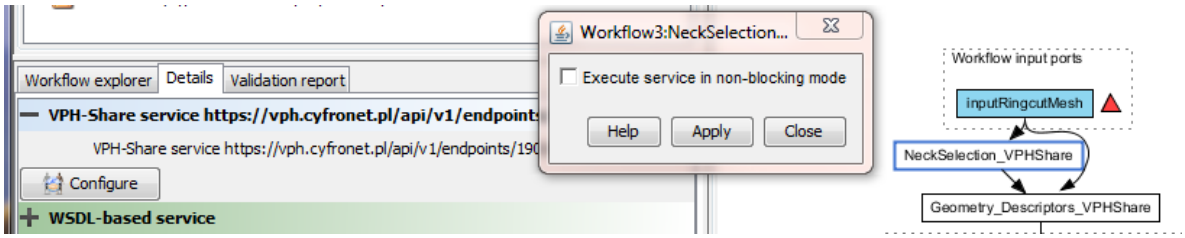


Figure 52. Configuration dialog for the NeckSelection VPH-Share service.

4.5 Workflow execution through the MI

Users can upload their composed Workflows to the MI, and share them with other users. Any user with access to a workflow can download it and load it into Taverna Workbench or Taverna On-line for editing and/or execution (see section 3.2.6 for details on the user interface). However, if the user does not need to edit the workflow but only wishes to execute it using inputs stored in LOBCDER, then the MI provides the *Execute workflow* button, which can be used to execute the workflow in the MI. The technology behind this button corresponds to the *Workflow Manager* (WM).

Once the user enters the parameters for the execution and presses the *Initialise execution* button, see Figure 33, the WM is activated behind the scenes, see Figure 53. The WM communicates with the Cloud Façade to start a new Application that runs the Taverna Server specified by the user, it waits for the server to be active and then submits to it the workflow selected by the user.

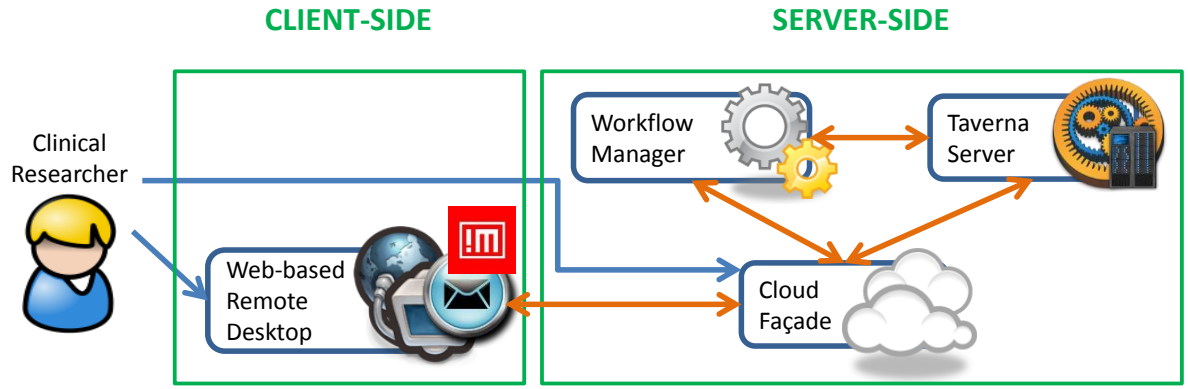


Figure 53. MI Workflow Execution Architecture overview.

If the submission of the workflow is successful, the WM proceeds to configure the workflow execution. This configuration process consist of specifying all necessary security parameters/certificates for the workflow to be allowed to run in the Cloud Façade, specifying the version and location of the VPH-Share plugin that is going to be used during workflow execution, specifying which services in the workflow require interaction and finally specifying what will be the input for the workflow, in the form of a *baclava* file. See

http://dev.mygrid.org.uk/wiki/display/taverna/DataViewer+Tool for more details about *baclava* files.

In addition, when the workflow is submitted to the server, it is given a unique identification string. Before executing the workflow, the WM creates a new output folder in the LOBCDER using this identification string. Then, the WM copies into this folder all input files required by the workflow. The outputs of the workflow will also be copied in this folder. In this way, the user can easily locate the files that were produced by the execution of the workflow that he/she chose.

Once the workflow is configured, the WM indicates the Taverna Server to start running it. The WM then request the server the status of the workflow executing every 5 seconds. During this time, if the user decides to run another workflow using the same server. The WM will detect that a Taverna Server is already running and will reuse it, meaning that it will submit the new workflow to the same server, so as to save computational resources.

Once the server indicates that a workflow has finished its execution, the WM will delete that workflow from the server, releasing all resources allocated by it. By this time, the outputs of the workflow will be already copied in the output folder. If the server is not running any other workflow, then the WM will also shut down the appliance that is running the Taverna Server, then again saving computational resources.

If an Application within a workflow requires user interaction, the Master Interface notification service is used to send the user a link to the web-based NX NoMachine client, which can be used to open a remote desktop session to the Appliance. Notifications reach the user by e-mail as well as through the Master Interface's GUI, see Section 2.2.4. In the case of the later, the user can click on the link and then a new web browser tab will open, automatically starting the NX NoMachine client. Then the user can interact with the AS, see Figure 48.

4.6 Batch execution

With the tools created in the VPH-Share project, the user can also execute the same workflow multiple times without manual intervention. This could be used for batch execution of the same workflow with a series of input data, which is very useful for running tests on multiple subjects, performing the same experiment multiple times and other common situations in the career of clinical researchers.

All Workflow Management Systems developed by Taverna have a built-in support for dealing with lists of data values. This means that, automatically, VPH-Share services can be input lists of values instead of single values. This later translates into what Taverna calls *implicit iterations*, see http://dev.mygrid.org.uk/wiki/display/taverna/Implicit+iteration. This is due to the fact that normally VPH-Share services have inputs of depth 0 (single values), and if the user feeds a workflow with an input of depth 1 (a list), Taverna will automatically apply the *implicit iterations* approach.

Normally, if the Taverna workflow has a single input port, this means that Taverna will perform as many *iterations* of the workflow as the number of values in the input list. That is, Taverna will execute the workflow as many times as the number of items in the input list. Each iteration takes as input one value from the input list. However, for a more detailed explanation on how 'implicit iterations' are performed, especially for workflows with multiple input ports, see http://dev.mygrid.org.uk/wiki/display/taverna/Implicit+iteration.

Batch execution of workflows can be performed either in the researcher's PC or on-line, as follows.

4.6.1 Desktop batch execution

For activating batch execution of a VPH-Share workflow in Taverna Workbench, the user simply has to edit the input port of the workflow. In the 'Workflow explorer', select the input port and press the right mouse button to open the pop-up menu. In that menu select 'Edit workflow input port'. In the dialog box that pops up select 'List of depth' and enter a depth of 1, see Figure 54 for an example. Press 'OK' to finish editing.

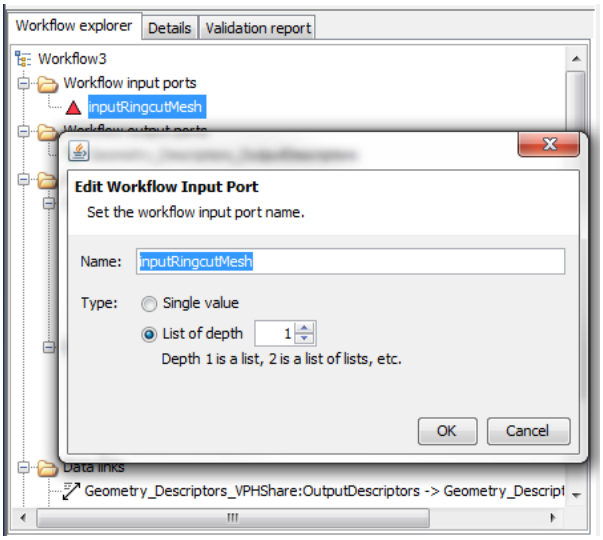


Figure 54. Taverna Workbench's edit input port dialog.

After this, before the user starts running the Taverna workflow, a list of values will be required as input in the Run Workflow dialog, see Figure 55. The user can enter the list manually or the 'Load previous values' button can be used to load a *baclava* file specifying the list of values. See http://dev.mygrid.org.uk/wiki/display/taverna/DataViewer+Tool for more details about *baclava* files.

After clicking on 'Run Workflow', Taverna will start running the workflow using the *implicit iterations* approach. During execution, in the 'Graph' tab, Taverna will show progress bars and iterations numbers on each service that is performing *implicit iteration*. Similar information is shown in the 'Progress Report' tab. For more details, see the 'Pipelining' section at http://dev.mygrid.org.uk/wiki/display/taverna/Implicit+iteration. In addition, notice

that Taverna will automatically perform *implicit parallelisation* to help increase data throughput, see http://taverna.knowledgeblog.org/2010/12/13/parallel-service-invocations/ for more details.

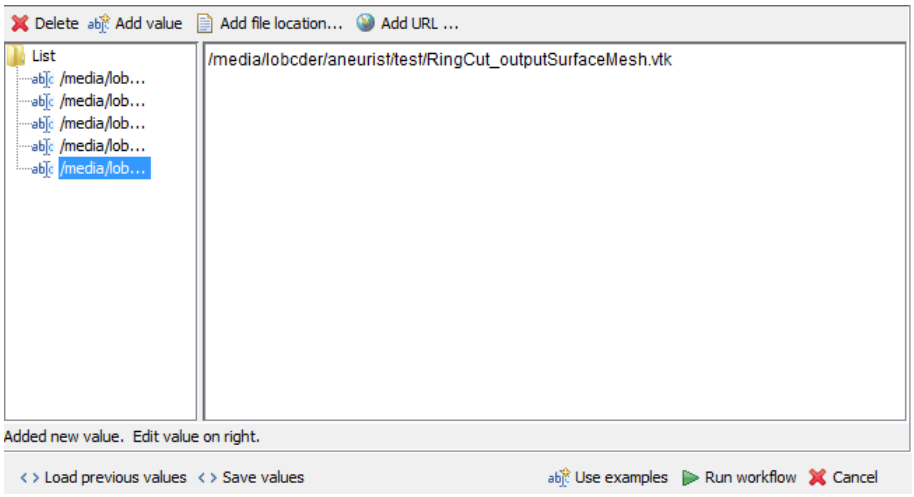


Figure 55. Taverna Workbench's Run Workflow dialog with input list.

For the first iteration, all the necessary Applications will be created by the VPH-Share Taverna Plugin. All following iterations will re-use the same Applications therefore saving the time and resources of continuously shutting down and re-starting the same set of Applications.

If an Application requires user interaction, a web browser window will automatically open in the user's desktop when the service is executed, so that the user can perform the interaction. The browser will start a web-based remote desktop session via a NX NoMachine client, see Figure 48. However, since in batch execution mode services are executed several times, the web-client will open only once per Application. The user must be careful not to close the browser tab, although in such a case it could be easily recovered using the browser's history. It is important to emphasise that the user must handle all the activations of the interactive service throughout all iterations, for the workflow to finish successfully.

The outputs of each iteration (intermediate and final workflow outputs) are stored in the LOBCDER, see Figure 47. However, it is important to notice that Taverna Workbench will not do any automatic renaming of the output files on each iteration. Therefore, the user must take care to input files on different locations of the LOBCDER, as otherwise the output files will be overwritten on each iteration. If the user does not want to worry about this, then the web execution tool can be used, for it will automatically create separate output folders in the LOBCDER. This is explained in the following section.

4.6.2 Web batch execution

Web batch execution can also be accomplished, by using the Workflow Manager (WM) available through the MI. From the point of view of the user interface, the process is exactly the same as executing a normal workflow using the *Execute workflow* button. However, two preconditions are necessary to accomplish web batch execution.

The first precondition is that the workflow chosen by the user through the MI must be already prepared to accept lists of depth 1 for input, as explained in the previous section. The second precondition is that the input *baclava* file must specify a list of input values, not just one single value. The user can upload to the MI such input *baclava* and workflow definition files and run the workflow directly in the VPH-Share portal.

In addition, if the workflow is expecting a list as input, and a *baclava* file with only one input value is used, this input will be automatically converted into a list with only one item. This mean that input *baclava* files for single execution can still be used with batch execution workflows.

Another difference between single and batch web execution relates to the output folder. In single execution the WM will create one output folder in the LOBCDER, using the unique identification string of the workflow, and copy the output files directly into that folder. However, in batch execution multiple output files with the same name will be produced and this could pose a problem because the output files would be overwritten at the end of each iteration of the workflow. In order to avoid this, the WM creates subfolders within the workflow output folder. Each subfolder is named after an iteration number, and so the output files produced by each iteration will be saved inside the subfolder that corresponds to the iteration that produced them.

4.7 Workflow Manager API

There is also available a XML-RPC API for the Workflow Manager, which can be used to start, monitor and stop workflows in the MI using a python script. The most relevant methods available are:

- execute_workflow: This method is able to start the execution of a workflow in a particular Taverna Server instance. It takes as input the workflow definition file, the input definition file, the user credentials, the workflow title and the details of the Taverna Server to which the workflow is going to be submitted. It produces as output, among other things, the identification of the submitted workflow.
- stopWorkflow: This method stops a specific workflow, previously started with execute_workflow. It takes as input the user credentials and the identification of the workflow to be stopped.
- getWorkflowInformation: This method can be used to monitor the execution of a workflow previously started with execute_workflow. It returns information such as the status of the workflow execution, the starting execution time, the creation time, any errors

or warning messages triggered by the Taverna Server, etc. This information is also kept inside the MI for monitoring purposes.

- deleteExecution: This method stops a workflow and clears the information about the workflow stored in the MI.

With these methods it is possible to create a simple python script to execute workflows. An example pseudo-algorithm for such a script would be:

```
import wfmng

wfDefinition = open('SampleWorkflow.t2flow', 'r').read()
inputDefinition = open('SampleWorkflowInputs.xml', 'r').read()
ret = execute_workflow(userCredentials, wfTitle,  tavernaServer, wfDefinition, inputDefinition)

info = getWorkflowInformation(ret['wfId'] , userCredentials)
while info and info['status'] != 'Finished' and info['error'] != True:
   time.sleep(5)
   info = getWorkflowInformation(ret['wfId'] , userCredentials)

deleteExecution(ret['wfId'] , userCredentials)
```

4.8 Data Provenance and Semantic

Currently, the Taverna VPH-Share plugin also incorporates functionality for creating metadata and provenance information for every single file that is produced in LOBCDER by the plugin. That is, for every file produced in LOBCDER an entry is automatically created in the Metadata Catalogue developed by WP4, which can then be used to publish the file and perform any intelligent data search that includes the file, see WP4 deliverables for more detail. An example entry for file 'Geometry Descriptors_OutputDescriptors.xml' stored in LOBCDER is shown below. If the file already exists, the metadata information is just updated.

```
<resource_metadata>
  <file>
    <author>ecoto</author>
    <category>GenericMetadata</category>
    <creationDate>2014-02-24 15:11:42.288</creationDate>
    <description>Taverna workflow output</description>
    <globalID>df7197ce-6ae1-442b-bbb9-7a7f2e0ac530</globalID>
    <linkedTo/>
    <localID>7962</localID>
    <metadataCreationDate>2014-02-24 15:11:42.288</metadataCreationDate>
    <metadataUpdateDate>2014-02-24 15:11:42.287</metadataUpdateDate>
    <name>Geometry Descriptors_OutputDescriptors</name>
    <provenance>
      <prov:document xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:prov="http://www.w3.org/ns/prov#"
            xmlns:share="http://www.vph-share.eu/ns/share#">
        <!-- Person -->
        <prov:person prov:id="ecoto">
```


```
            <prov:role>executor</prov:role>
         </prov:person>
         <!-- Webservice -->
         <prov:entity id="e3c17fe0-1657-4911-97d9-5148d78d5b74">
            <prov:label>Geometry_Descriptors</prov:label>
            <prov:location> https://vph.cyfronet.pl/api/v1/endpoints/188/descriptor
            </prov:location>
         </prov:entity>
         <!-- Input Data -->
         <prov:entity id="Dbee5df8-8c4c-46e0-a3c6-11597478d81b">
            <prov:label>inputFileNameDome</prov:label>
            <prov:location> lobcder:/aneurist/NeckSelection_outputSurfaceMesh.vtk
            </prov:location>
         </prov:entity>
         <!-- Input Data -->
         <prov:entity id="5b5a9dd2-be74-40a7-82d8-42cbc9368754">
            <prov:label>inputFileNameAneurysm</prov:label>
            <prov:location> lobcder:/aneurist/RingCut_outputSurfaceMesh.vtk
            </prov:location>
         </prov:entity>
         <!--  Entities used in generation -->
         <prov:wasGeneratedBy>
           <prov:entity prov:ref="ecoto"/>
           <prov:entity prov:ref="e3c17fe0-1657-4911-97d9-5148d78d5b74 "/>
           <prov:time>2014-02-24 15:49:57</prov:time>
         </prov:wasGeneratedBy>
         <!-- Output Data -->
         <prov:entity id="df7197ce-6ae1-442b-bbb9-7a7f2e0ac530">
           <prov:label>OutputDescriptors</prov:label>
           <prov:location>
             lobcder:/aneurist/Geometry Descriptors_OutputDescriptors.xml
           </prov:location>
         </prov:entity>
         <!--  Entities used in derivation -->
         <prov:wasDerivedFrom>
           <prov:generatedEntity prov:ref="df7197ce-6ae1-442b-bbb9-7a7f2e0ac530"/>
           <prov:usedEntity prov:ref="Dbee5df8-8c4c-46e0-a3c6-11597478d81b"/>
           <prov:usedEntity prov:ref="5b5a9dd2-be74-40a7-82d8-42cbc9368754"/>
           <prov:time>2014-02-24 15:49:57</prov:time>
         </prov:wasDerivedFrom>
       </prov:document>
    </provenance>
    <rating>0</rating>
    <relatedResources/>
    <semanticAnnotations/>
    <status>active</status>
    <type>File</type>
    <updateDate>2014-02-24 15:11:42.288</updateDate>
    <views>0</views>
    <fileType>XML</fileType>
    <format>XML</format>
    <size>429</size>
    <subjectID/>
  </file>
</resource_metadata>
```

Notice the highlighted field <provenance>, which contains a provenance document following the PROV-XML schema, see http://www.w3.org/TR/prov-xml/. The generated provenance document specifies the location of the file, its owner, its ID in the Catalogue, and all the

entities that were involved in the production of the file, such as the web service that produced it and the files that were used as input to the web service. With this information, the user could re-produce the file, if needed. In year 4, the user will be able to visualise this metadata and provenance information in the Master Interface.

4.9 Workflow Monitoring

The first attempts towards creating a Workflow Monitoring service have been carried out, by deploying a Nagios Core engine for the project. This is a free and open source solution monitoring system, see http://www.nagios.com/products/nagioscore. The system supports the installation of specific purpose plugins, from which the WebInject plugin (http://www.webinject.org/plugin.html) has been chosen for Workflow Monitoring. The plugin will allow us to write a script with a series of steps for testing the workflow execution through the MI. Each step in the script will use the REST interface of the Workflow Manager (WM), which will be available in year 4, to execute a method that will test the functionality and availability of the WM.

At the moment, three test services have been setup. See the three services under hosts 'VPH-Share' in Figure 56. The first one just checks that the portal.vph-share.eu returns a valid HTTP response, which indicates that this web page is up and reachable. The second and third services use the WebInject plugin, so each one runs a test script using operations. The script of the second service has been set to fail on purpose, so as to test the reaction of the server to this failure. Notice how the failed service is highlighted and its status is set to 'CRITICAL'. The system administrator received an alert e-mail when the service failed. The script of the third service has been set to be successful. Notice how the service's status is set to 'OK'.

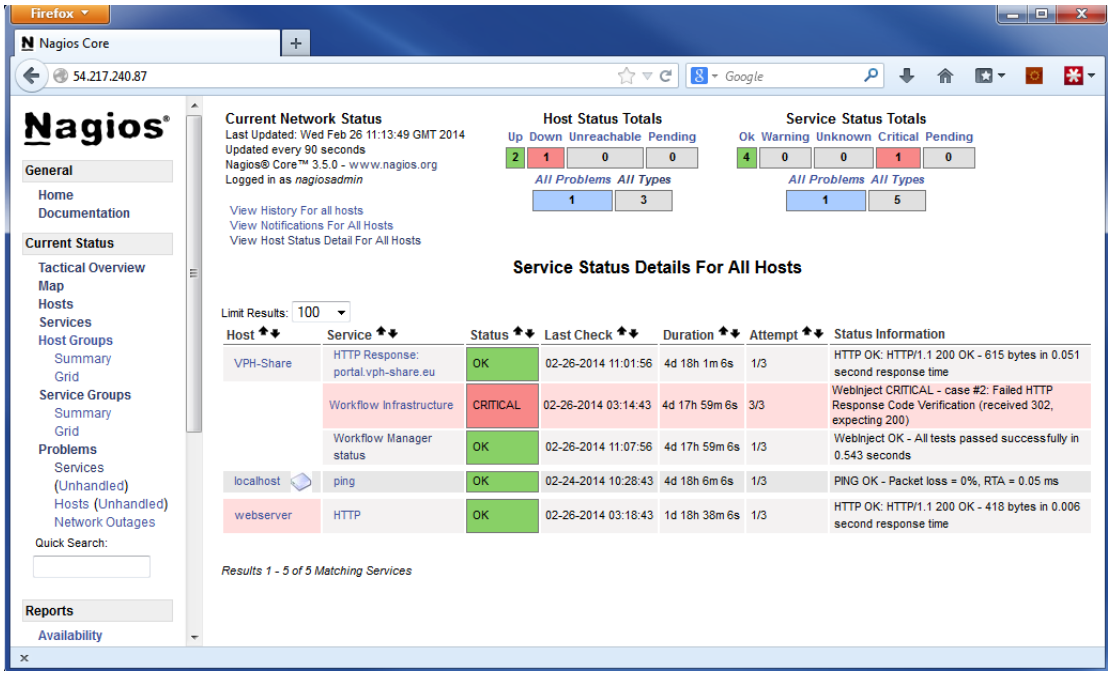


Figure 56. Nagios Core web interface showing VPH-Share service monitoring

5 YEAR 3 OUTCOMES

WP6 Year 3 Achievements	**WP6 Implementation Team Roles: O- Owner C- Contributor**			
Results delivered	**CYF**	**CINECA**	**IITP**	**USFD**
Remote visualisation services based on Paraview	C	O		C
LOBCDER web client	O	C		C
GIMIAS Webservice wrapper extended to run long lasting requests	C	C		O
VPH-Share Taverna plugin extended to run long lasting request	C	C		O
Workflow Manager	C	C		O
Web Workflow composition services	C	C	O	C
Workflow execution services integration on the MI	C	O		C
Integration of new Atmosphere 2.0 API	O	C		C
Integration of new Metadata Catalogue	C	C		C
User cases and external projects support (with bug fixing)	C	O		C
New cloud management interface	O	C		C
Initial implementation of services monitoring	C	C		O
Taverna server deployment 2.4.1, 2.5.2,2.5.3	C			O
Initial implementation of provenance	C			O
VPH-Share webservice catalogue	O	C	C	C
Messaging tool integration		O		
User registration tool		O		
Institutions and groups management in the MI		O		
MI UI improvements (i.e. dashboard, sharing)	C	O		C

Table 1. WP6 Year 3 Achievements

6 WORK PLANNED FOR YEAR 4

WP6 Year 4 High-level plan	**WP6 Implementation Team Roles: O- Owner C- Contributor**			
Results to be delivered (list will be extended following requests from users and developments from other WPs)	**CYF**	**CINECA**	**IITP**	**USFD**
User cases support	C	O		C
External projects support	C	O		C
Full integration with Taverna Online (LOBCDER & workflow repository)	C	C	O	C
Full implementation of provenance (Visualisation)	C	C		O
REST interface to execute VPH-Share workflows	C	C		O
Extended user interface to execute workflows From MI	C	O		C
Cloud Monitoring integration	C	C	O	C
Cloud Billing Model	C	O		C
New home page (facelift)		O		C
Data browsing improvements		O		C
Bug fixing and maintenance of the UI	C	O		C
Workflow queue manager		O		C

Table 2. WP6 Year 4 Plan

LIST OF KEY WORDS/ABBREVIATIONS

3DRA	3-Dimensional Rotational Angiography
AIR	Adversarial information retrieval
AMS	Application Management System
AS	Appliance
API	Application Programming Interface - Infrastructure services for cloud metadata management
AWS	Infrastructure services for cloud site management
BSD	Berkeley Software Distribution
CF	Cloud Façade
CLP	Command Line Plugin
CRUD	Create, Read, Update and Delete operation
CSV	Comma Separated Values
DB	Database
DPS	Data Publication Suite
GAR	Geodesic Active Regions
GIMIAS	Graphical Interface for Medical Image Analysis and Simulation
GPL	GNU General Public License
GUI	Graphical User Interface
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
IITP	Institute for Information Transmission Problems
JQUERY	multi-browser JavaScript library designed to simplify the client-side scripting of HTM

JSON	JavaScript Object Notation
LOBCDER	Large OBject Cloud Dataf storagE fedeRation
MI	Master Interface
NX	NX technology is a computer program that handles remote X Window System connections
OGSI-DAI	Open Service Gateway Initiative - Data Access and Integration
PKI	Public-Key Infrastructure
RDF	Resource Description Framework
REST	REpresentational State Transfer
SOAP	Simple Object Access Protocol
SPARQL	SPARQL Protocol and RDF Query Language
SQL	Structured Query Language
SSH	Secure SHell
SSL	Secure Socket Layout
UAS	User Access System
UI	User Interface
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
VM	Virtual Machine
VMT	Virtual Machine Template
VNC	Virtual Network Computing (protocol)
VTK	Visualisation Toolkit (www.vtk.org)
WADL	Web Application Description Language
WebDAV	Web Distributed Authoring and Versioning

WebDRS	Web Drug Ranking System (a service from ViroLab)
WP	Workpackage
WSDL	Web Service Description Language
WS	Web Service
XML	eXtensible Markup Language
XML-RPC	XML encoded Remote Procedure Call (protocol)

This page was intentionally left blank