

After all, the right virtual research environment?



West-Life, Computation
and Data Management
for Structural Biologist

O RLY?

#WestLifeSB

Table of Contents

Introduction	1.1
Data Management	1.2
Virtual Folder	1.3
User's guide	1.3.1
Settings	1.3.1.1
File Manager	1.3.1.2
File Picker	1.3.1.3
Installation guide	1.3.2
Cloud installation	1.3.2.1
Local installation	1.3.2.2
Integration guide	1.3.3
Select File or Dir from Virtual Folder	1.3.3.1
Working with WEBDAV	1.3.3.2
Embedding Virtual Folder Component	1.3.3.3
Adding component into Virtual Folder	1.3.3.4
Developer's guide	1.3.4
Backend	1.3.4.1
Frontend - Web Application	1.3.4.2
Metadata and API	1.3.4.3
Dataset metadata and API	1.3.4.3.1
File metadata and API	1.3.4.3.2
Virtual Machines	1.4
Repository	1.5
Installation guide	1.5.1

Prerequisites	1.5.1.1
Automatic installation	1.5.1.2
Manual installation from source codes	1.5.1.3
User's guide	1.5.2
Developer's guide	1.5.3
ARIA integration	1.5.3.1

Introduction

This documentation contains administrator, user's and developer's guide of several products produced by H2020 West-Life project.

Documentation is rendered as

- HTML version: <https://h2020-westlife-eu.gitbooks.io/virtual-folder-docs/content/>
- [Data Management](#) gives introduction and general overview of subsequent products and how they fit into data infrustrctre and data management ecosystem already available.
- [Virtual Folder](#) provides a unified access mechanism to files stored in a variety of locations including the local file system, and B2DROP and other cloud storage facilities.
- [Virtual Machines](#) Various images and configuration are prepared for production, development and testing purposes.
- [Repository](#) provides exemplar implementation of data management withing small facility who'd like to have web app support for data life cycle.

Data Management

Introduction

Research data is acquired, interpreted, published, reused, and sometimes eventually discarded. Structural biology has a strong tradition of data sharing, expressed by the founding of the Protein Data Bank (PDB) in 1971¹. In 2015, 9338 new structures were deposited in the Protein Data Bank, which is the result of approximately more than 25,000 experiments². All these experiments have together a combined data rate greater than that of the Large Hadron Collider.

One of the main obstacles to fully achieve a proper handling of the data life cycle in structure biology is managing the data, which will include datasets acquired in a range of different experimental facilities, some easy to transfer by email or USB stick, and some so large that it is only feasible to process them at source.

To address this obstacles, [Virtual Folder](#) provides common interface to access scattered data among data infrastructures offered by facilities and common data infrastructures offered by scientific community (EUDAT, Indico, and others now in upcoming (EOSC) European Open Science Cloud effort) or commercial vendors (Dropbox, Google Drive, ...). Virtual Folder is provided as public service without any other installation as well as configuration for [private deployment](#) is available for processing data using common software suites for structural biology in local workstation, cluster or user's preferred cloud computing infrastructure.

For experimental facilities that are newly embarking on data management, we provide a reference implementation of a [Repository](#) that supplies suitable metadata to the portal, follows common workflow of project as defined ARIA and allows deposit datasets via Virtual Folder to desired data infrastructure. Note that this reference installation expects to be ammended based on the feedback of

facility users thus technology choosen for backend and frontend implementation are described and source code provided in order to facilitate further customization and development.

¹. PDB, 1971. Protein Data Bank. Nature New Biology. ↩

². Assessment of the life cycle of structural data and comparison with other scientific data, West-Life Report, D3.1, www.structuralbiology.eu/upload/west-life/W-L_Deliverable_D3_1.pdf ↩

Virtual Folder

Introduction

"Virtual Folder" provides a unified access mechanism to files stored in a variety of locations including the local file system, and B2DROP and other cloud storage facilities. It is developed in work package WP6 by the [West-Life H2020 project](#), which is running from 2015 to 2018. It provides application level services usable for structural biology use cases and follows [the structural biology data lifecycle](#). Data management work package WP6 build on existing infrastructure for storing and accessing data to create a "Virtual Folder".

Usage

A public installation of the Virtual Folder is available at [West-Life portal](#). Follow [User's guide](#) for further details how to use it.

Installation

Virtual Folder can be installed: 1. from source codes 2. from binaries at cernvm-fs 3. from cloud template, contextualized to binaries at cernvm-fs

For all the installation options follow: [Installation guide](#).

Brief installation instruction from source codes

Recommended configuration for private deployment of virtual machine with Virtual Folder:

- Minimal: 1 CPU, 2 GB RAM, 50GB disk space.
- Recommended: 4 CPU, 8 GB RAM, 100 GB disk space
- Desktop, Server OS: Windows (tested on Windows 7, Windows 2012), Linux (tested on Ubuntu 14.04 LTS, Ubuntu 16.04 LTS), VirtualBox 5.1.6+, Vagrant 1.8.6+
- Cloud OS: OpenNebula, OpenStack, or any other supporting OVA images.

```
git clone https://github.com/h2020-westlife-eu/west-life-wp6  
  
cd west-life-wp6  
  
vagrant up
```

The new virtual machine can be accessed via web browser `http://[vm.ip.address]:[port]/` By default, `http://localhost:8080` Virtual folder provides WEBDAV API at `http://[vm.ip.address]:[port]/webdav` e.g. `http://localhost:8080/webdav`

Inside VM, the files of the current working directory of host are mounted into `/vagrant` and the repositories of virtual folder are accessible at `/home/vagrant/work`

Integration

The functionality of the Virtual Folder can be integrated into other portal or web application. The recommended method is to link the existing component using cross-document messaging mechanism - e.g. for picking a file from Virtual Folder by using the File Picker component.

Follow [Integration guide](#) for further details.

Development

In order to test, contribute to source codes and prepare virtual machine environment with source codes, follow [Virtual Machines](#) chapter.

Release Notes

- 14/05/2018 - update docs to new gitbook system - no PDF, updated links.
- 01/04/2018 - docs moved to westlife-docs repository, docs covering Virtual Folder, Virtual Machines and Repository
- 28/06/2017 - UI improvement, docs in sync with gitbook and /doc folder in dev-docs branch
- 23/06/2017 - added support for source code installation/binary installation
- 01/05/2017 - Added Dataset demo integrates some web PDB components to visualize features of PDB and UniProt entries
- 01/04/2017 - Added Upload Dir Picker component and updated integration guide
- 07/03/2017 - public deployment integrated with SSO authentication ARIA at www.structuralbiology.eu
- 25/11/2016 - Updated vagrant boxes to use uCernVM 2.7.7 bootloader, updated OVA images in <https://appdb.egi.eu/store/vappliance/d6.1.virtualfoldervm/vaversion/latest> and vagrant boxes, do "vagrant box update", bug fixes, consolidated initial web page and design, fixed/added background services
- 26/10/2016 - moved VagrantFile to new repository <https://github.com/h2020-westlife-eu/wp6-vm>, updated base box with uCernVM2.7.4 bootloader for CernVM 4 fixes security bug 'dirty COW' and aufs bug in kernel, <https://atlas.hashicorp.com/westlife-eu>,
- tested on Windows 7 64 bit, vagrant 1.8.6 + VirtualBox 5.1.6, vagrant 1.8.1, 1.8.4 + VirtualBox 5.0.26, note vagrant < 1.8.6 requires VirtualBox 5.0.x, doesn't require VirtualBox extension pack, download from https://www.virtualbox.org/wiki/Download_Old_Builds_5_0
- tested on Ubuntu 14.04 LTS, default vagrant 1.4.3 needs to be updated to

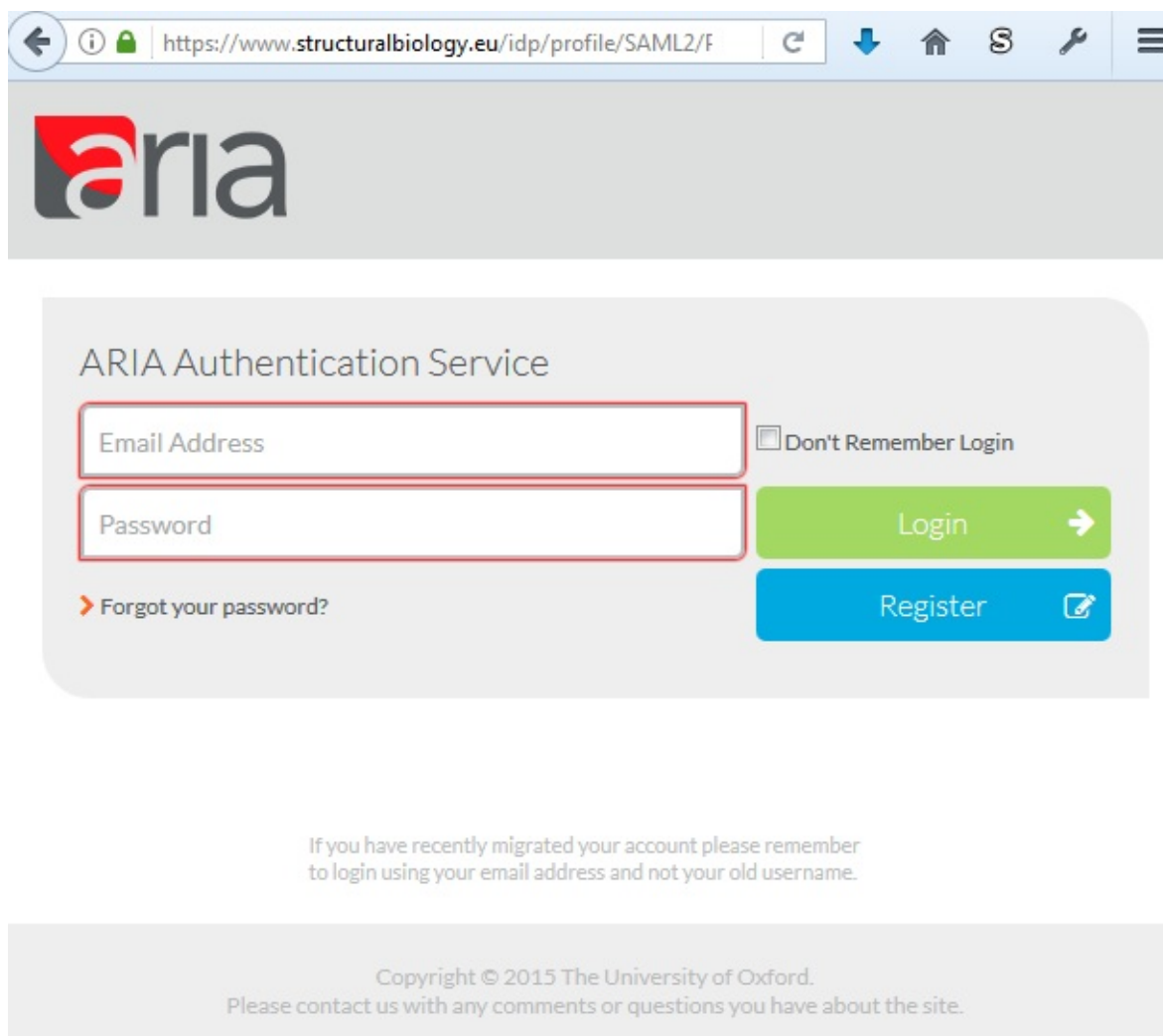
1.8.6), default VirtualBox 4.3.36 works

User's guide

The Virtual Folder is accessible from the West-life portal at <https://portal.west-life.eu>



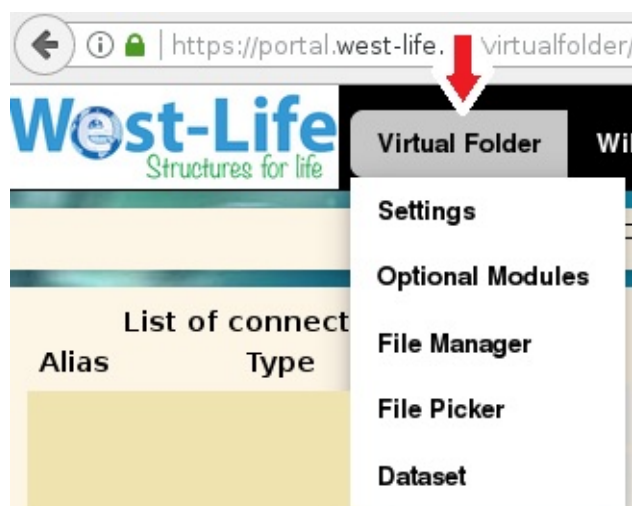
After you click the Login page, you will be redirected to the ARIA authentication service operated by Instruct. You can login or create a new account.



The screenshot shows a web browser window with the URL <https://www.structuralbiology.eu/idp/profile/SAML2/F>. The page features the ARIA logo at the top. Below the logo, the text "ARIA Authentication Service" is displayed. There are two input fields: "Email Address" and "Password", both outlined in red. To the right of the "Email Address" field is a checkbox labeled "Don't Remember Login". Below the "Password" field is a green "Login" button with a right-pointing arrow. To the left of the "Login" button is a link that says "Forgot your password?". Below the "Login" button is a blue "Register" button with a right-pointing arrow. At the bottom of the page, there is a copyright notice: "Copyright © 2015 The University of Oxford. Please contact us with any comments or questions you have about the site."

After succesful login, you'll be redirected to your account in Virtual Folder at <https://portal.west-life.eu/virtualfolder/>.

User can navigate through Virtual Folder components by hovering the 'Virtual Folder' menu.

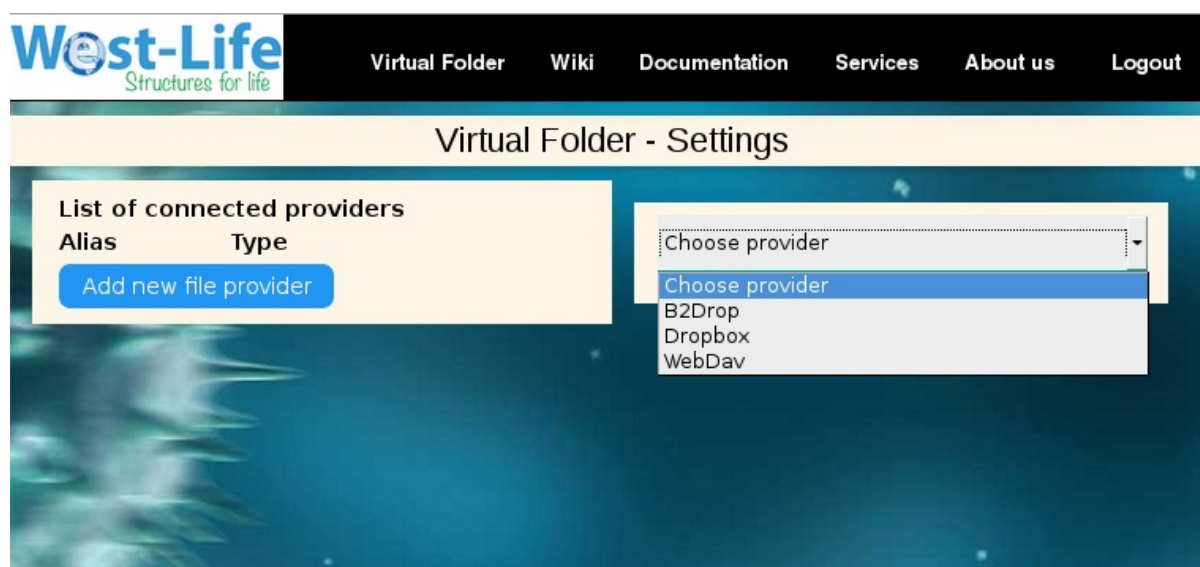


Settings

Connecting Scattered Data

In order to use the Virtual Folder, you need to connect some storage where your scattered data are accessible.

Add new file provider -> Choose provider



The B2DROP, Dropbox and any storage provider offering a [standard WebDAV connection](#) are supported.

B2DROP

[B2DROP](#) is a secure and trusted data exchange service for researchers and scientists. West-life portal uses B2DROP TO store, upload and download AND share the data files.

You need to create B2DROP account first at <https://b2drop.eudat.eu/pwm/public/NewUser?>

Select B2DROP -> Fill in the existing B2DROP username and password -> click Add

The screenshot shows the 'Virtual Folder - Settings' page. On the left, there is a 'List of connected providers' table with columns 'Alias' and 'Type'. Below the table is a blue button labeled 'Add new file provider'. On the right, there is a form for connecting a B2DROP provider. The form has a dropdown menu with 'B2Drop' selected. Below the dropdown, there is a text box for 'Username' containing 'john.smith@example.com', a text box for 'Password' with masked characters, and an optional text box for 'Alias'. A small note explains that the alias is a unique name for the folder. At the bottom right of the form is a black button labeled 'Add'.

Alias	Type
-------	------

[Add new file provider](#)

B2Drop

B2DROP is academic secure and trusted data exchange service provided by EUDAT. West-life portal uses B2DROP TO store, upload and download AND share the data files.

You need to create B2DROP account first at b2drop.eudat.eu/pwm/public/NewUser? Fill in the existing B2DROP username and password here:

Username:

Password:

Alias (optional):

Alias is a unique name of the 'folder' under which the provider will be 'mounted' and accessible.

[Add](#)

After clicking the Add button, and if everything works well, the connected B2DROP account should appear in the list:

The screenshot shows the 'Virtual Folder - Settings' page after the B2DROP account has been added. The 'List of connected providers' table now contains one entry: 'b2drop B2Drop'. To the right of the entry is a 'Browse content' button and a delete icon (an 'X' inside a circle). Below the table is a blue button labeled 'Add new file provider'.

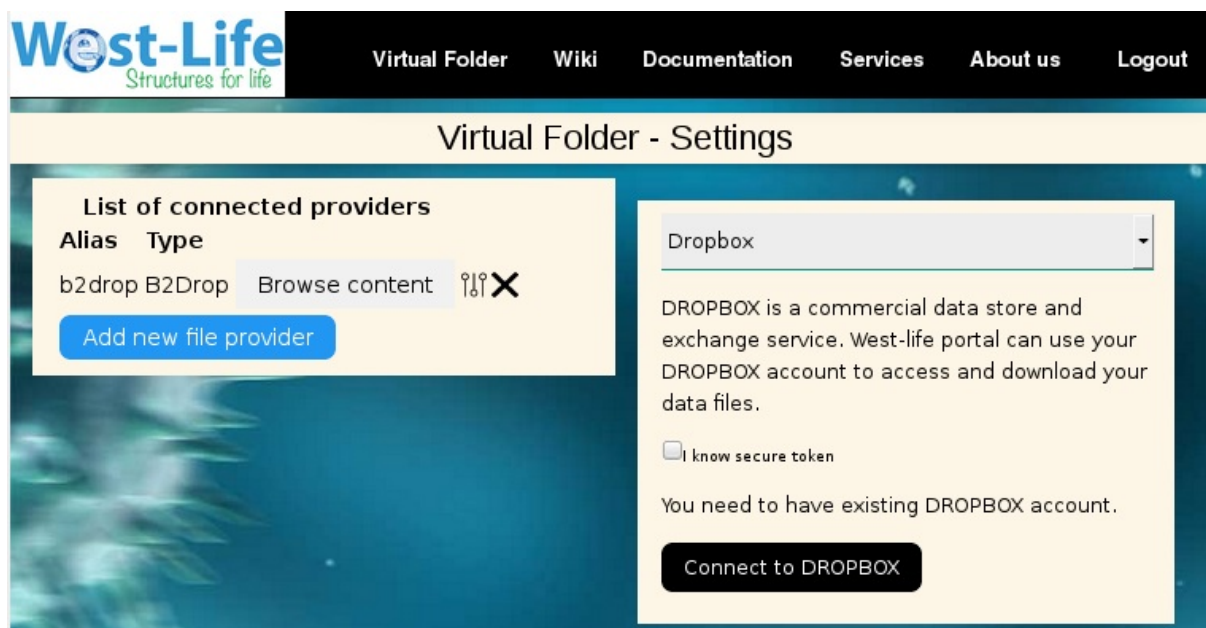
Alias	Type
b2drop B2Drop	Browse content

[Add new file provider](#)

DropBox

DROPBOX is a commercial data store and exchange service. West-life portal can use your DROPBOX account to access and download your data files.

Select DropBox -> click 'Connect to DROPBOX'



You will be redirected to sign in or create Dropbox account. After login, you'll be asked to allow Virtual Folder to access your data:

Click Allow



Virtual Folder would like access to the files and folders in your Dropbox. [Learn more](#)

Cancel

Allow

You'll be redirected back to Virtual Folder. You should see a unique secure token being filled into the secure token field. Do not change it. You can optionally fill the 'Alias'.

click Add

West-Life
Structures for life

Virtual Folder Wiki Documentation Services About us Logout

Virtual Folder - Settings

List of connected providers

Alias	Type
b2drop	B2Drop

Browse content

[Add new file provider](#)

Dropbox

DROPBOX is a commercial data store and exchange service. West-life portal can use your DROPBOX account to access and download your data files.

☒ I know secure token

Secure token:

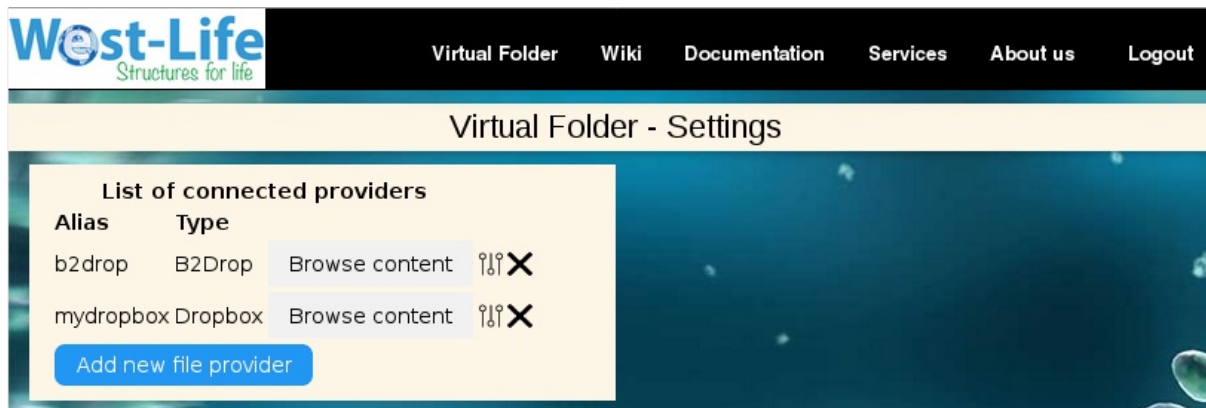
Xx1y

Alias (optional):

Alias is a unique name of the 'folder' under which the provider wil be 'mounted' and accessible.

[Add](#)

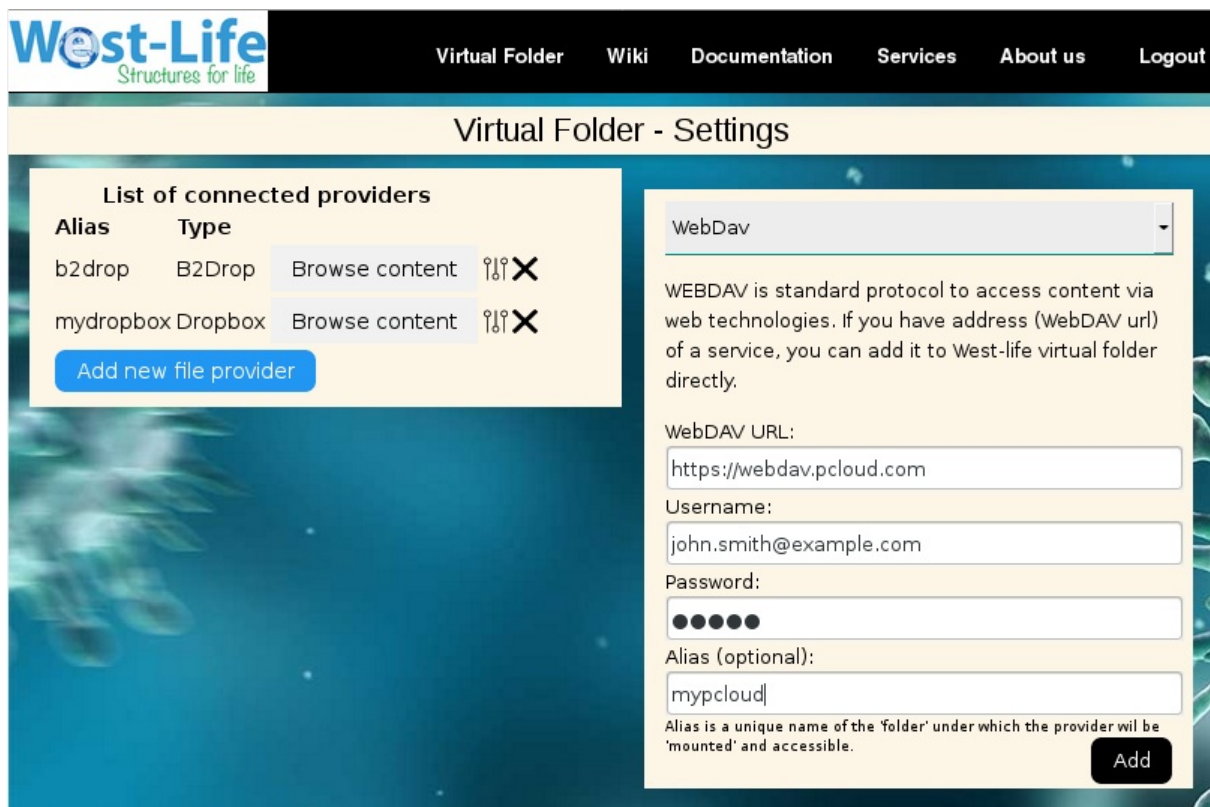
After clicking the Add button, and if everything works well, the connected DROPBOX account should appear in the list:



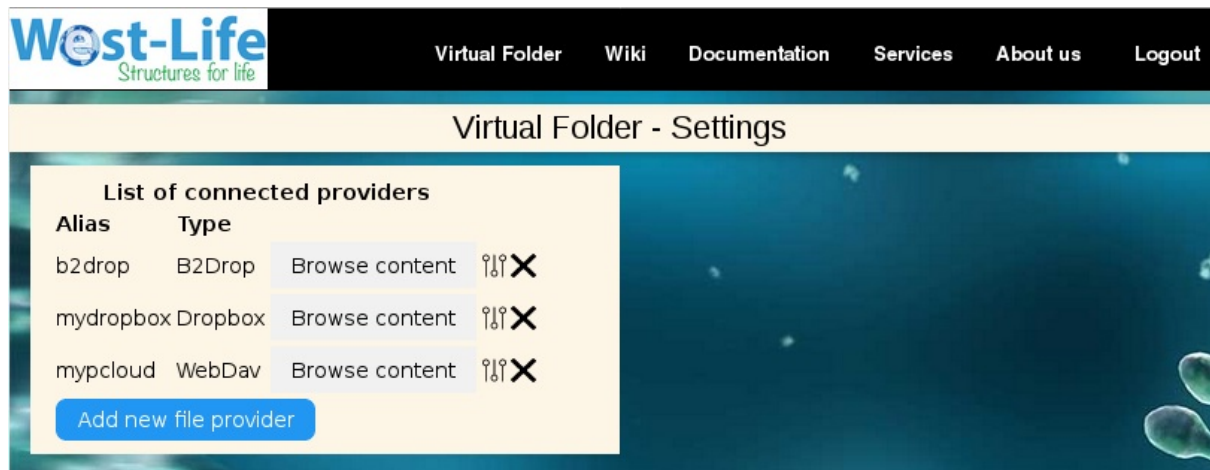
WEBDAV

If your data provider allows a WEBDAV protocol, this can be connected.

Add new file provider -> Select 'WEBDAV'



Fill in the WebDAV URL and account information -> click Add



The screenshot shows the 'Virtual Folder - Settings' page of the West-Life application. The page has a dark blue header with the West-Life logo and navigation links: Virtual Folder, Wiki, Documentation, Services, About us, and Logout. Below the header, the title 'Virtual Folder - Settings' is displayed. The main content area features a light blue box titled 'List of connected providers'. This box contains a table with three columns: 'Alias', 'Type', and 'Browse content'. There are three rows of connected providers, each with a 'Browse content' button and a delete icon (X). Below the table is a blue button labeled 'Add new file provider'.

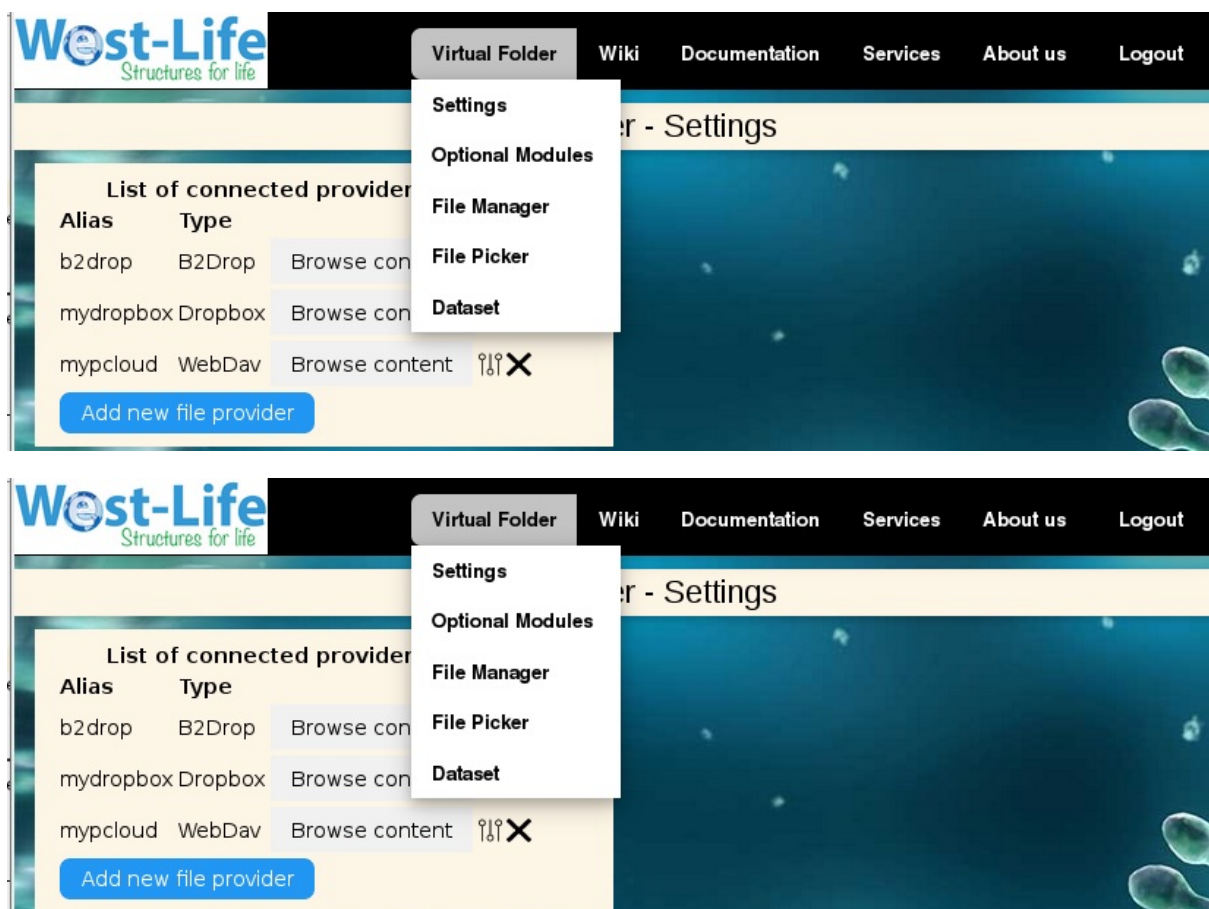
Alias	Type	Browse content	
b2drop	B2Drop	Browse content	X
mydropbox	Dropbox	Browse content	X
mycloud	WebDav	Browse content	X

[Add new file provider](#)

After clicking the Add button, and if everything works well, the connected WEBDAV provider should appear in the list.

File Manager

To use the File manager of the Virtual Folder, either go directly to <https://portal.west-life.eu/virtualfolder/filemanager.html> or click on the 'Browse content' button on any connected provider or hover over the "Virtual Folder" menu entry and click the 'File Manager' menu item.



File list view

The file manager consists of 2 panels, you can browse the directory structure on each panel. Clicking on directory shows details of the directory, clicking in '..' goes into parent directory:

The screenshot shows the 'Virtual Folder - File manager' interface. The top navigation bar includes 'Virtual Folder', 'Wiki', 'Documentation', 'Services', 'About us', and 'Logout'. The main content area has two panels, each with tabs for 'File List', 'View/Edit', 'Visualize', and 'Dataset'. The left panel shows a file list for '/mydropbox' containing 4 items. The right panel shows a file list containing 3 items.

name	size	date
..	UP DIR	
pdb	DIR	08/05/2017
Get Started with Dropbox.pdf	905827	16/12/2016
README.md	3640	16/12/2016
4xmm.pdb	2293110	12/01/2017

name	size	date
mydropbox	DIR	
mycloud	DIR	
b2drop	DIR	

Visualize

Clicking on any file with PDB extension will switch to the Visualize tab - (LiteMol component integrated from https://www.ebi.ac.uk/pdbe/pdb-component-library/doc.html#a_LiteMol)

The screenshot shows the 'Virtual Folder - File manager' interface with the 'Visualize' tab selected. The left panel displays the 'EMBL EBI PDB Viewer' with the file '4xmm.pdb' selected. The right panel shows a file list containing 3 items. The main area displays a 3D molecular structure visualization of the protein 4xmm.pdb. The bottom of the interface includes a 'Density' button and a text prompt: 'Click on a residue or an atom to view the data.'

View/Edit

Clicking on any file with extension other than PDB will switch to View/Edit -


The screenshot shows the West-Life File Manager interface. The top navigation bar includes links for Virtual Folder, Wiki, Documentation, Services, About us, and Logout. The main header is 'Virtual Folder - File manager'. Below this, there are two panes. The left pane, titled 'File List', shows a list of files with columns for name, size, and date. The right pane, titled 'Dataset', shows a list of files with columns for name, size, and date. The 'Dataset' pane is currently selected, showing a list of files including 'pdb', 'Get Started with Dropbox.pdf', 'README.md', '4xmm.pdb', and '4xmm.ent'.

File List	View/Edit	Visualize	Dataset
1	HEADER	TRANSPORT PROTEIN/IMMUNE SYSTEM	14-
2	TITLE	STRUCTURE OF THE YEAST COAT NUCLEOPORIN COM	
3	COMPND	MOL_ID: 1;	
4	COMPND	2 MOLECULE: PROTEIN TRANSPORT PROTEIN SEC13;	
5	COMPND	3 CHAIN: A;	
6	COMPND	4 ENGINEERED: YES;	
7	COMPND	5 MOL_ID: 2;	
8	COMPND	6 MOLECULE: NUCLEOPORIN NUP145;	
9	COMPND	7 CHAIN: B;	
10	COMPND	8 SYNONYM: NUCLEAR PORE PROTEIN NUP145;	
11	COMPND	9 EC: 3.4.21.-;	
12	COMPND	10 ENGINEERED: YES;	
13	COMPND	11 MOL_ID: 3;	

File List	View/Edit	Visualize	Dataset
/mydropbox contains 5 items. refresh			
name	size	date	
..	UP DIR		
pdb	DIR	08/05/2017	
Get Started with Dropbox.pdf	905827	16/12/2016	
README.md	3640	16/12/2016	
4xmm.pdb	2293110	12/01/2017	
4xmm.ent	2293110	08/05/2017	

Dataset

Clicking on the Dataset pane change the view to dataset definition dialog.



Virtual F

Virtual Folder






























































































































































































































































































































































































































































File List View/Edit Visualize **Dataset**

Dataset demo

PDB or related item to add:

PDB Prints

1r6a

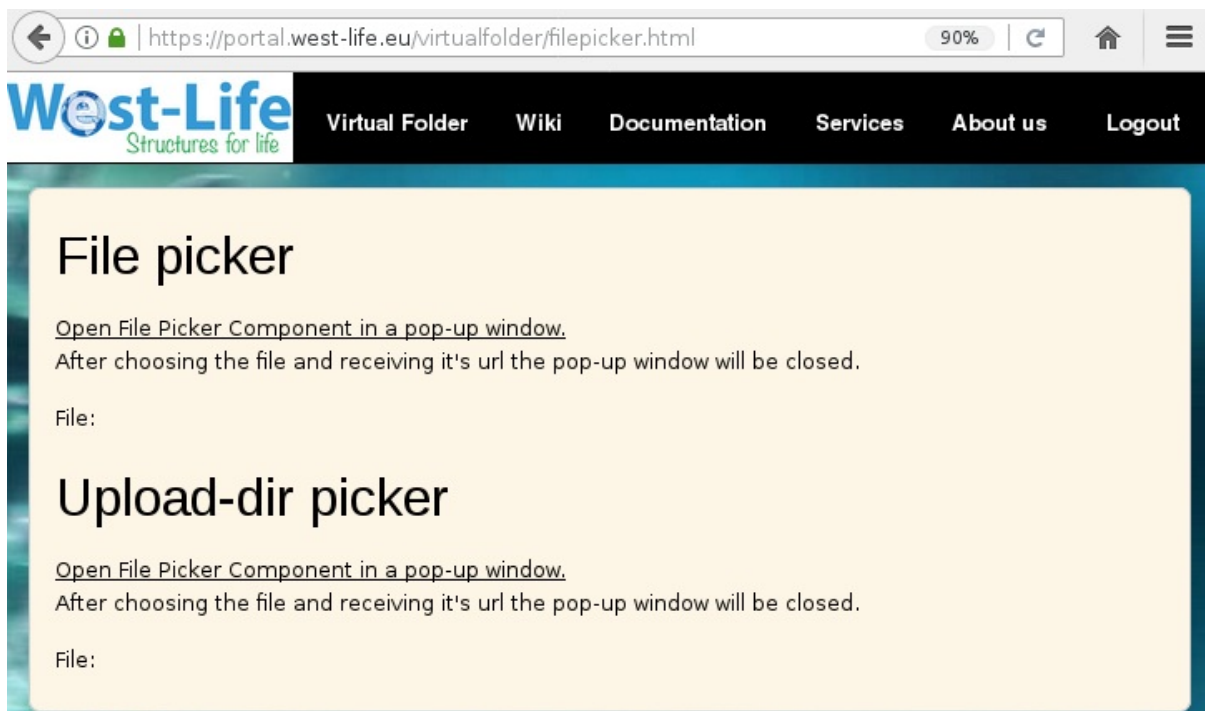
File Picker

File picker

You can use the file picker component to select a file from any storage repository which you have connected to the virtual folder and generate publicly accessible URL which can be used to download the file from virtual folder. In order to integrate File picking into your web application follow [Integration guide -> Select File or Dir from Virtual Folder](#).

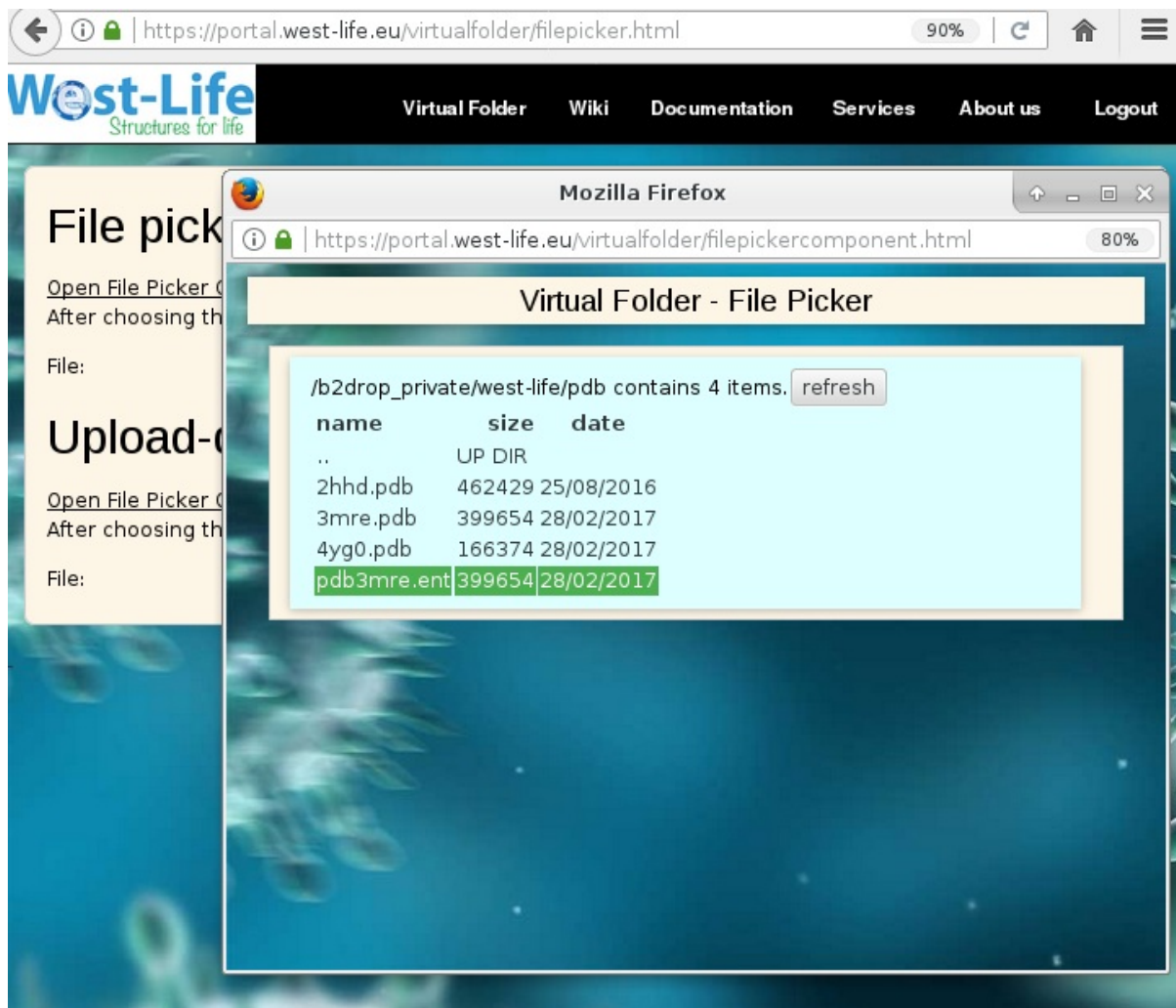
Exemplar file picker and upload dir picker web application is at

<https://portal.west-life.eu/virtualfolder/filepicker.html>

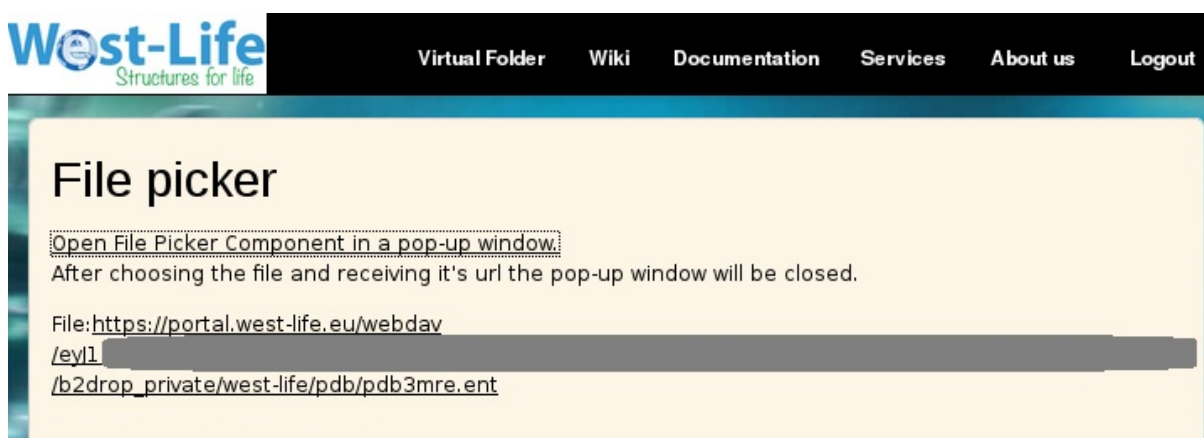


click the link 'Open File Picker Component'

Popup window appears where you can browser and choose the file to be picked.



After selecting a file the generated link can be used to access the file directly



Upload dir picker

To pick directory within Virtual Folder, where a results can be uploaded directly, the second component 'Upload Dir picker' can be used.

Installation guide

You can install Virtual Folder as an virtual machine instance, as an alternative to using it at <https://portal.west-life.eu/virtualfolder>.

There are 3 options

- [Cloud installation](#) - to install in OpenNebula, OpenStack or any other cloud computing provider.
- [Local installation](#) - using Virtual Box on local workstation or cluster for testing purposes or for private deployment.
- [Development installation](#) - same as "Local installation", but contains source code of Virtual Folder. This is covered in separate section "Virtual Machines"

Cloud installation

Images to deploy Virtual Folder are maintained and distributed via appdb.egi.eu.

1. <https://appdb.egi.eu/store/vappliance/d6.1.virtualfoldervm> - The standard OVA (open virtual appliance) image can be used to deploy West-Life VM into e.g. OpenNebula cloud environment.
2. <https://appdb.egi.eu/store/vappliance/west.life.vm> - RAW image can be used to deploy West-Life VM into OpenStack cloud environment

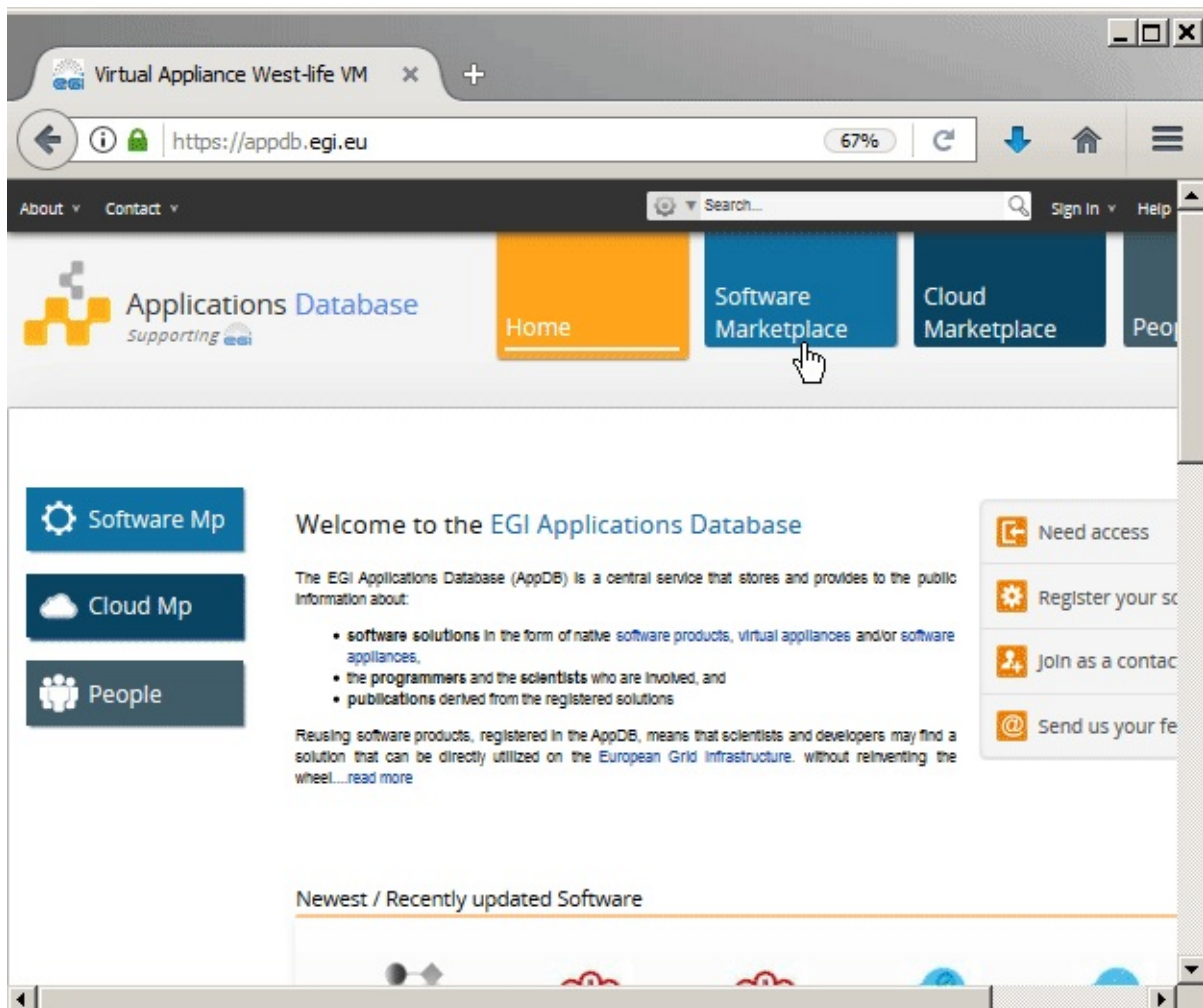
Both images are small (18 MB, 23 MB respectively) containing only CernVM 4 bootloader, which boots into standard Scientific Linux (currently version 7.3) and contextualizes it with West-Life specific software:

- Virtual Folder – private instance
- VRE – (optional) private instance delivering multiuser environment
- CCP4 – (optional available after license agreement) software suite containing software tools to process data produced by X-ray crystallography methods
- Scipion – image processing framework to process data produced mainly by electron microscopy methods.
- WeNMR – software tools to process data mainly of NMR methods

Downloading VM image

You may download the latest West-life VM in the OVA compatible format from

<https://appdb.egi.eu/store/vappliance/d6.1.virtualfoldervm>



Alternatively you may use the RAW image for deployment into OpenStack at <https://appdb.egi.eu/store/vappliance/west.life.vm>

Use the downloaded image to deploy it to preferred provider. You may deploy the image in EGI resources - follow EGI documentation.

Contextualization

OVA image for OpenNebula contains by default contextualization, RAW image for OpenStack contains reference to cloud-init file which needs to be provided in order to boot Virtual Folder and related software during first boot. By default the image contextualization is set to use direct connection to CernVM-FS repositories. If you have local squid proxy server, it is very recommended to configure it in contextualization script.

```
[ucernvm-begin]
cvmfs_http_proxy=http://<host>:<port>
[ucernvm-end]

[cernvm]
proxy = http://<host>:<port>;DIRECT
```

Further information about contextualization of CernVM can be found at <https://cernvm.cern.ch/portal/contextualisation>

Local installation

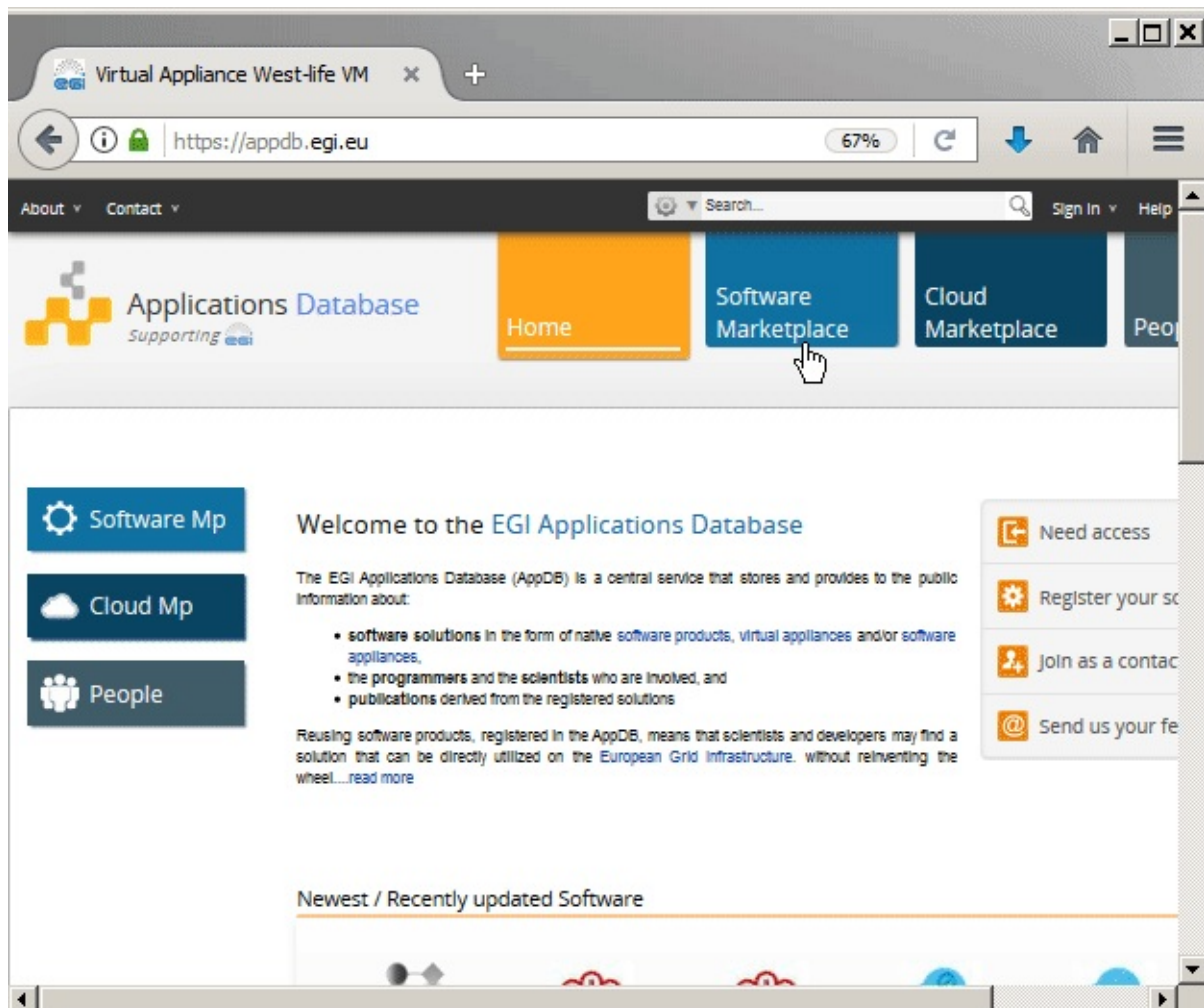
In order to install/launch legacy software and have access to a desktop you may install the virtual folder with selected software suites either on your cloud provider, or locally on your server, cluster, workstation or even laptop.

The standard procedure is to download virtual machine template image from appDB (18MB). If you would like to have testing environment set by Vagrant tool and VirtualBox, follow [Virtual Machines](#) chapter.

Downloading VM image

You may download the latest West-life VM in the OVA compatible format from

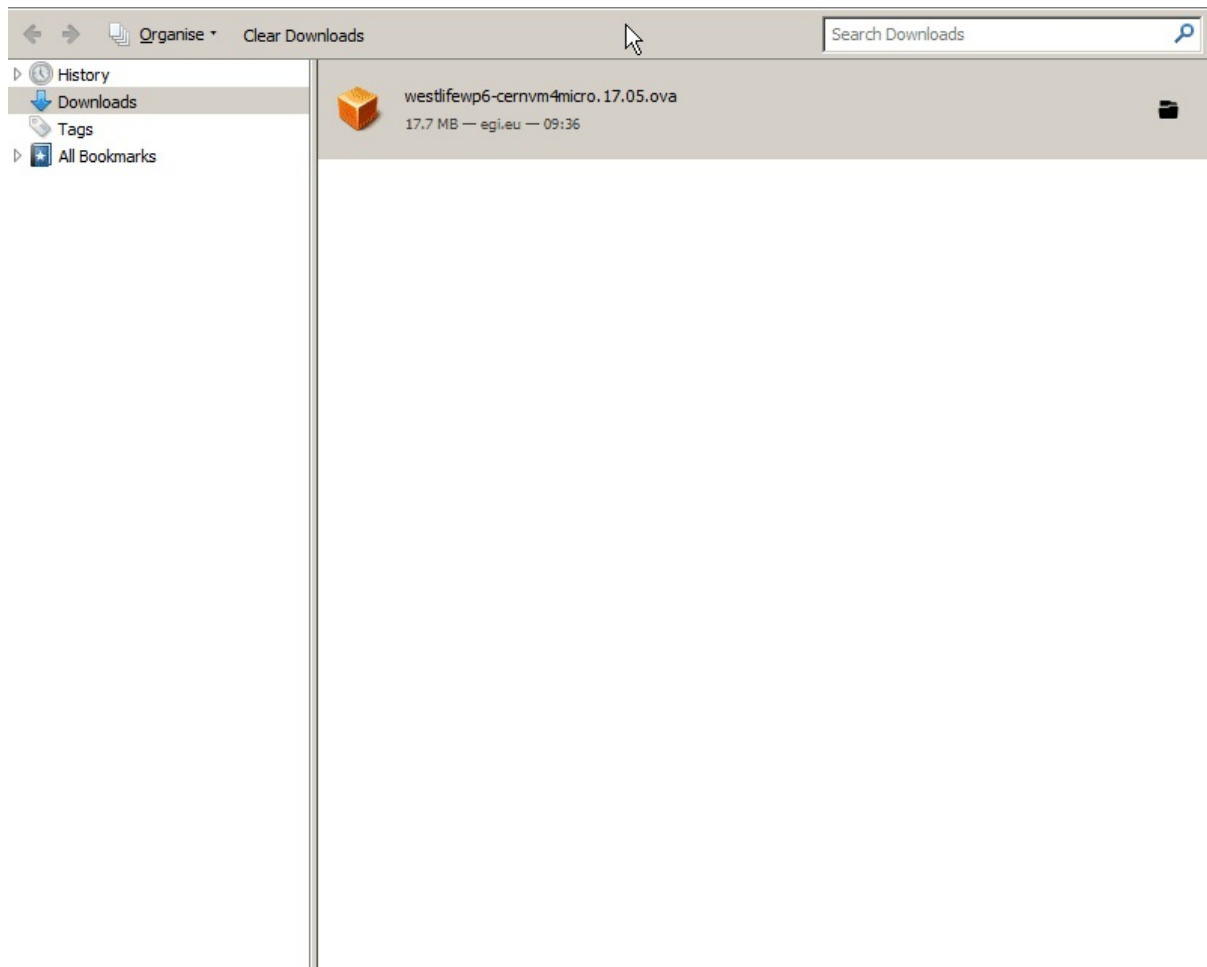
<https://appdb.egi.eu/store/vappliance/d6.1.virtualfoldervm>



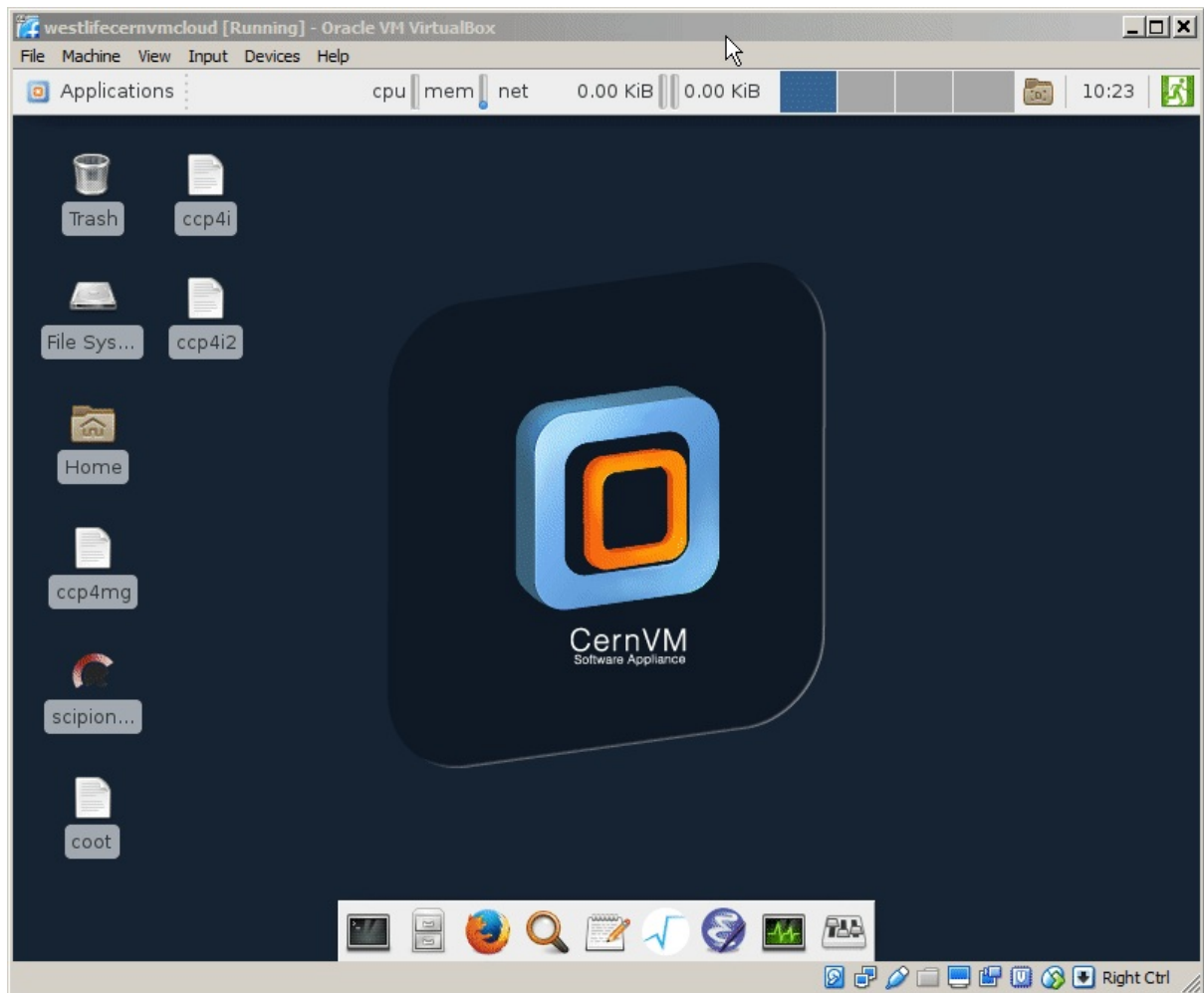
Alternatively you may use the RAW image for deployment into OpenStack at <https://appdb.egi.eu/store/vappliance/west.life.vm>

Deployment to local VirtualBox

For testing purposes you may import the image into local installation of VirtualBox.



The OVA image initially 18 MB bootstraps operating system and additional software by downloadin about 100-200 MB, the initial bootstrap can take about several minutes. You can access the desktop directly.



Integration guide

The public west-life portal is hosted at <https://portal.west-life.eu>. You can integrate Virtual Folder functionality by these main strategies:

- link/popup of west-life web page, see filepicker demo. The filepicker.html is hosted by your web site. The filepickercomponent.html is hosted by west-life. Continue with [Select File or Dir from VF](#)
- integrate using WEBDAV protocol using the WEBDAV endpoint URL obtained by filepicker component. Continue with [Working with WEBDAV](#).
- embedding west-life component - the web page is hosted by your web site. It links to some functionality hosted by west-life which is projected to the web page. Continue with [Embedding component](#)
- adding component - VF web component are maintained as open source, follow [Adding component](#)

Select File or Dir from Virtual Folder

Virtual Folder contains File Picker and Upload-Dir Picker components. They can send a message containing an URL of the selected File or Directory using cross-document messaging API. The integration is done by following:

- The web page (e.g. `myfilepicker.html`) is hosted by [yourweb]
- the following Javascript code in your web pages opens pop-up browser window with `filepickercomponent.html` and `uploaddirpickercomponent.html` hosted by West-Life portal.
- The `receiveMessage` Javascript function gets the url of picked file or dir via cross-document messaging API.

Shared Script Code

Add the following javascript code into your web page (`myfilepicker.html`):

```
<script>
var popup;
var target;
//opens popup window in defined location, sets the target element id.
function openwindow(_target,href) {
    target = _target;
    popup=window.open(href, 'newwindow', 'width=640, height=480');
}
//receives message from popup window, fills target element with the data received
function receiveMessage(event) {
    document.getElementById(target).innerHTML=event.data;
    document.getElementById(target).setAttribute("href",event.data);
}
window.addEventListener("message", receiveMessage, false);
</script>
```

File picker

The `filepickercomponent.html` is hosted by west-life portal and returns an URL pointing to the file. HTTP GET of that URL lead to download the file.

Add the following HTML snippet into your web page (`myfilepicker.html`). It will open West-life filepicker component in popup window:

```
<a href="enablejavascript"

    onclick="openwindow('fileid',
    'https://portal.west-life.eu/virtualfolder/filepickercomponent.html
    ',
    ); return false;">West-life File</a>

<p>File:<a id="fileid" href=""></a></p>
```

Note: `'fileid'` links to the element where the URL of chosen file will appear later.

Upload Dir Picker component

The `uploaddirpickercomponent.html` is hosted by west-life portal and returns an URL pointing to the directory. You can HTTP POST or use WEBDAV access to that directory to upload files.

Add the following HTML snippet into your web page (`myfilepicker.html`). It will open West-life upload dir picker component in popup window:

```
<a href="enablejavascript.html"
    onclick="openwindow('uploaddirid',
    'https://portal.west-life.eu/virtualfolder/uploaddirpickercomponent
    .html'
    ); return false;"> West-life Upload Dir..</a>
```

```
<p>Directory URL:<a id="uploadid" href=""></a></p>
</div>
```

Note: 'uploadid' links to the element where the URL of chosen file will appear later.

This reuses the same javascript as for filepicker component.

References

:Cross-document

Messaging:

<https://html.spec.whatwg.org/multipage/comms.html#web-messaging>

Working with WEBDAV

Working with WEBDAV

West-Life File picker and Upload-Dir picker components return URL capable of WEBDAV API which doesn't need any other type of authentication. Use this URL as confidential as possible. WEBDAV is HTTP extension, it supports basic HTTP VERBS to download file ('GET'), upload file ('PUT'), delete file ('DELETE') as well as extension to list resources ('PROPFIND') in a directory or directories. It is possible to use WEBDAV capable client application to connect to the selected folder or download selected file. Following sections contains samples how to download or upload file to Virtual Folder using WEBDAV API using scripting or compiled languages. Full samples can be downloaded

- bash:

<https://gist.github.com/TomasKulhanek/c94e148159a871ee688685828da82ebd>

- python:

<https://gist.github.com/TomasKulhanek/9d939350d234ec43ff1ffac8d1baa1f4>

Bash, CURL

Prerequisite: install curl using your OS package manager `yum install curl` or `apt-get install curl`. cURL is command-line tool for transferring data using various protocols available as standard package in most OS.

Bash script to UPLOAD file to the webdav URL

```
curl -X PUT [updirurl+filename] --data "any text content"
```

Bash script to DOWNLOAD file from the webdav URL

```
curl [updirurl+filename] --data "any text content"
```

[Download full sample script](#)

Python

Prerequisite:

- for Python 2.x install easywebdav using your favorite packaging system(`pip install easywebdav`)
- for Python 3.x download easywebdav from [here](#).

```
import easywebdav;

if sys.argv[1] == 'PUT':
    url = urlparse(sys.argv[2]);
    webdav = easywebdav.connect(url.hostname,protocol=url.scheme,path=
url.path);
    print("Uploading ", sys.argv[3], " to ", sys.argv[2]);
    webdav.upload(sys.argv[3], sys.argv[3]);
else:
    url = urlparse(sys.argv[1]);
    webdav = easywebdav.connect(url.hostname,protocol=url.scheme,path=
url.path);
    print("Downloading ",sys.argv[2]," from ", sys.argv[1]);
    webdav.download(sys.argv[2], sys.argv[2]);
```

[Download full sample scripts](#)

Javascript [Draft]

Not yet verified. The following code uses XMLHttpRequest API.


```
function ajaxrequest(method, url, callback,contentdivid,content) {
    var request;
    // compatible with IE7+, Firefox, Chrome, Opera, Safari
    request = new XMLHttpRequest();
    request.onreadystatechange = function(){
        if (request.readyState == 4 && request.status == 200){
            callback(request.responseText,contentdivid);
        } else {
            console.log(request.response)
            alert('There was an error. See console. ');
        }
    }
    request.open(method, url, true);
    request.send(content);
}

function downloadcallback(responsetext,contentdivid) {
    document.getElementById(contentdivid).innerHTML = responsetext;
}

function uploadcallback(responsetext,contentdivid) {
    document.getElementById(contentdivid).innerHTML = "Upload OK.";
}

function downloadwebdavfile(url,contentdivid) {
    ajaxrequest("GET",url, downloadcallback,contentdivid)
}

function uploadwebdavfile(url,contentdivid,content) {
    ajaxrequest("PUT",url,uploadcallback,content)
}
```

.NET [Draft]

Standard [HttpRequest](#) class use used

```
public static string Put(string url, string filename, string content)
{
    string log = "";
```

```
        try
        {
            // Create an HTTP request for the URL.
            HttpRequest httpPutRequest =
                (HttpRequest) WebRequest.Create(url+"/"+filename
e);

            httpPutRequest.PreAuthenticate = false;

            httpPutRequest.Method = @"PUT";
            httpPutRequest.Headers.Add(@"Overwrite", @"T");

            httpPutRequest.ContentLength = content.Length;

            httpPutRequest.SendChunked = true;
            Stream requestStream = httpPutRequest.GetRequestStream
();

            requestStream.Write(
                Encoding.UTF8.GetBytes((string) content),
                0, content.Length);

            requestStream.Close();

            HttpResponse httpPutResponse =
                (HttpResponse) httpPutRequest.GetResponse();

            log += @"PUT Response: " + httpPutResponse.StatusDescr
iption;

            return log;
        }
        catch (Exception e)
        {

            Console.WriteLine("PUT Response: Exception" + e.Message
e + " StackTrace:" + e.StackTrace);
            throw e;
        }
    }
}
```

Java [Draft]

In preparation.

```
//webdav client sample
```

References:

HTTP Extensions for Web Distributed Authoring and Versioning (WebDAV):
tools.ietf.org/html/rfc4918

XMLHttpRequest Living Standard: <https://xhr.spec.whatwg.org/>

easywebdav 1.2.0 A straight-forward WebDAV client, implemented using
Requests <https://pypi.python.org/pypi/easywebdav>

cURL, <https://curl.haxx.se/docs/>

Embedding Virtual Folder Component

In order to embed the Virtual Folder component into your web application you need to first declare the `virtualfolderbaseurl` variable pointing to the endpoint where virtualfolder services are available. In this case we will use public endpoint "<https://portal.west-life.eu>", but you may use endpoint of your private instance of Virtual Folder:

```
<head>
...
<script>
  var virtualfolderbaseurl="https://portal.west-life.eu";
</script>
</head>
```

Now within the body of your web page you can include a component of virtual folder by adding following `<div>` tag:

```
<div aurelia-app="[componentname]">
  <script src="https://portal.west-life.eu/virtualfolder/scripts/vendor-bundle.js"
    data-main="aurelia-bootstrapper"/>
  <script src="https://portal.west-life.eu/virtualfolder/scripts/app-bundle.js"/>
  Loading ...
</div>
```

The `[componentname]` could be one of the following:

- `filepicker/main` - File picker - allows to pick the file from Virtual Folder and it's WEBDAV URI is returned to managing page. See the demo at [filepicker.html](#) and [filepickercomponent.html](#).

- `filemanager2/main` - File manager - allows to browse files in defined file providers (demo view of PDB). See the demo at `filemanager.html`.
- `dataset/main` - Dataset definition page - allows to define list of entries - PDB and Uniprot entries can be refined with
- `virtualfoldersettings/main` - Virtual Folder Settings page - allows to manage file providers (B2DROP,Dropbox,Filesystem,...)

The complete example of embeded component in your web may look:

```
<html>
<head>
...
  <script>
    var virtualfolderbaseurl="https://portal.west-life.eu";
  </script>
</head>
<body>
<div aurelia-app="filemanager2/main">
  <script src="https://portal.west-life.eu/virtualfolder/scripts/vendor-bundle.js"
    data-main="aurelia-bootstrapper"/>
  <script src="https://portal.west-life.eu/virtualfolder/scripts/app-bundle.js"/>
  Loading ...
</div>
</body>
</html>
```

Flask integration example

Several web services provided within West-Life relies on [Flask python microframework](#). Below are few lines allowing to handle the URL sent by the VF component. In this implementation, we assume that the Flask-wtf form, instead of taking a file as input, will deal with a `str` that is the URL sent by the VF component. The implementation example can be found at this URL (only for

DEVELOPMENT purposes, do not advertise it!!!) -> <https://milou.science.uu.nl/cgi/servicesdevel/DISVIS/disvis/submit> The field is declared in the `forms.py` :

```
pdb_url = TextField(
    u'OR select fixed chain from your VRE',
    render_kw={'class': 'form-control', 'placeholder': 'File URL once
selected'},
    validators=[validators.Optional(), InvalidPDB_url()])
```

And in `submit.html` :

JavaScript

```
$(document).ready(function(){
    $('label[for="pdb_url"]').append('<br><a href="javascript:void(0)"
onclick="openwindow(); return false;"> Choose VRE file.</a>')
})
// This does nothing, assuming the window hasn't changed its location.
function openwindow() {
    target = event.target.parentNode.getAttribute('for');
    popup=window.open('https://portal.west-life.eu/virtualfolder/filep
ickercomponent.html', 'newwindow', 'width=640, height=480');
}

function receiveMessage(event)
{
    if (event.data != "") {
        document.getElementById(target).value=event.data;
        $('#'+target).trigger("change");
    }
    //popup.close(); //closed by child
}
window.addEventListener("message", receiveMessage, false);
```

HTML

```
<form action="{{ url_for('submit') }}" method="POST" enctype="multipar
t/form-data">
```

```
{{ render_field(form.pdb_url) }}  
</form>
```

Upon successful validation, we want to download the file to process them afterwards, in `views.py` :

```
import urllib2  
  
# Utility method to download a file locally from a URL  
def download_file(file_url, work_dir, fname):  
    logging.info("VF URL: {}".format(file_url))  
    try:  
        u = urllib2.urlopen(file_url)  
        f = open(safe_join(work_dir, fname), 'wb')  
        block_sz = 8192  
        while True:  
            buffer = u.read(block_sz)  
            if not buffer:  
                break  
            f.write(buffer)  
        f.close()  
    except HTTPError:  
        logging.error("Error during file download, aborting...")  
        raise  
  
@app.route('/submit', methods=['GET', 'POST'])  
def submit():  
    form = InputDataForm()  
    ... # Initialization of different variables..  
    ...  
    if form.validate_on_submit():  
        try:  
            # Get URL from the TextField  
            pdb_url = form['pdb_url'].data # Get  
            download_file(pdb_url, work_dir, fname)  
            print os.path.isfile(safe_join(work_dir, fname)) # Should return '  
True'
```


Adding component into Virtual Folder

- start component as a service inside Virtual Folder VM or container
- configure the proxy to redirect to the component
- Vf filesystem is at /home/vagrant/work so the external service can use it directly, webdav repositories are synchronized by davfs module, dropbox and ther are not synchronized.

Developer's guide

In order to have complete development environment follow either [Development installation]. Or just get source codes from <https://github.com/h2020-westlife-eu/west-life-wp6>

Git secrets

Within the public git repository, there are secrets to be used for demonstration purposes, these are encrypted using git-secret module.

Git-secret is installed within [wp6-vm/rep-standalone-src-sl7]. To manually install git-secret in different environment, launch

```
# git-secret require git version 2.0
yum -y install http://opensource.wandisco.com/centos/6/git/x86_64/wandisco-git-release-6-1.noarch.rpm
yum -y install git
# install git-secret
wget https://bintray.com/sobolevn/rpm/rpm -O bintray-sobolevn-rpm.repo
mv bintray-sobolevn-rpm.repo /etc/yum.repos.d/
yum -y install git-secret
```

Further details see <http://git-secret.io/>

New user:

```
#user needs a new gpg key pair, create
gpg2 --gen-key

#within his copy of git, replaces [mygpg@email.com] with your key alias
git secret tell [mygpg@email.com]
```

```
# git commit & git push
```

Trusted user:

```
# git pull
# reveal all secret file
git secret reveal
# encrypt with all keys including the new one
git secret hide
# git commit & git push
```

New user:

```
# git pull
# now reveal all secrets
git secret reveal
```

Cheat sheet:

```
# make sure you have gpg key pairs - to generate new
gpg2 --gen-key

# initialize - usually done once in git repository
git secret init

# add user among trusted users
git secret tell -m
# or
git secret tell mygpg@email.comg

# add file to the secrets
echo 'myfile.txt' >>.gitignore
git secret add myfile

# encrypt all secreted file
git secret hide

# decrypt all secreted file using trusted user's key
git secret reveal
```

Remove unwanted commits from git tree

Use it when accidental commit was done and to remove all sensitive files.

<https://help.github.com/articles/removing-sensitive-data-from-a-repository/>

Backend

Backend

Backend components consist of metadataservice (C# .NET), common portal (Python, Django) and Bash scripts and configuration files.

Database configuration

`/home/vagrant/.westlife` contains database configuration and key to access secured content. In order to backup or move configuration, it needs to be moved together.

Release Notes

- 25/11/2016 - Updated vagrant boxes to use uCernVM 2.7.7 bootloader, updated OVA images in <https://appdb.egi.eu/store/vappliance/d6.1.virtualfoldervm/vaversion/latest> and vagrant boxes, do "vagrant box update", bug fixes, consolidated initial web page and design, fixed/added background services
- 26/10/2016 - moved VagrantFile to new repository <https://github.com/h2020-westlife-eu/wp6-vm>, updated base box with uCernVM2.7.4 bootloader for CernVM 4 fixes security bug 'dirty COW' and aufs bug in kernel, <https://atlas.hashicorp.com/westlife-eu>,
 - tested on Windows 7 64 bit, vagrant 1.8.6 + VirtualBox 5.1.6, vagrant 1.8.1, 1.8.4, VirtualBox 5.0.26, note vagrant < 1.8.6 requires VirtualBox 5.0.x, doesn't require VirtualBox extension pack, download from https://www.virtualbox.org/wiki/Download_Old_Builds_5_0

- tested on Ubuntu 14.04 LTS, (default vagrant 1.4.3 needs to be updated to 1.8.6), default VirtualBox 4.3.36 works

Further doc

<http://internal-wiki.west-life.eu/w/index.php?title=D6.1>

Frontend - Web Application

The source codes of frontend components are at <https://github.com/h2020-westlife-eu/west-life-wp6/tree/master/wp6-virtualfolder/www>

Following Chapter 6. Section 1. Step 4. the sources will be in `/vagrant/west-life-wp6/wp6-virtualfolder/www`

Frontend application is based on HTML (5) and Javascript (EcmaScript version 6) uses aurelia framework to

- handle http communication
- bundle component into vendor-bundle.js and app-bundle.js and transpile it to ES5 - compatible Javascript implementation in most web browsers in the year 2017.
- maintain source code and use best practices/patterns to compose/configure the parts together

To use/integrate the components see [[Embedding components guide]]

To develop/maintain the components, the following procedure is recommended:

1. restore npm modules

```
npm install
```

2. install aurelia-cli - globally

```
sudo npm install aurelia-cli -g
```

3. run `au run --watch` , if some dependencies are not met - install the module using `npm install <module-name>`

Adding static content

To add static content into West-life VF web UI, commit it to directory `wp6-virtualfolder/www`. Apache server on the portal and on custom VM is by default configured with [SSI \(Server Side Include\)](#) module. The `header.html` and `footer.html` can be included into other html. To add new HTML page:

- add `your.html` and all relevant content (JS,CSS) into `wp6-virtualfolder/www` and it's subdirectories:
- edit `your.html` and

include `w3.css` styles and other relevant styles and scripts

```
<head>
...
<!--#include file="heads.html" -->
...
</head>
```

include header of the pages shared among all of them, add this line up to the html content of `your.html` :

```
<body>
<!--#include file="header.html" -->
...
</body>
```

(Optionaly) include footer, add this line bottom to the html content of `your.html` :

```
<!--#include file="footer.html" -->
</body>
```

- edit `header.html` to contain link into your new page.

add e.g. following row into desirable place of the menu

```
<li>  
<a href="your.html">Your Service</a></li>
```

virtualfoldersettings

- List of configured providers - component `aliastable`
- Configure new provider - component `genericcontrol` - lists available providers
- Three providers supported - `FileSystem`(mounted folder), `Dropbox`(Oauth or securetoken), `B2Drop`(username password), the same for backend web service
- After adding - the list of configured providers is updated
- If returned from dropbox auth url - the url contains `access_token` - reopens the dropbox add dialog with securetoken readonly.
- can add multiple instances of same provider: `Dropbox` (`dropbox_1`, `dropbox_2`) ...
- cannot add multiple instances of these providers: `B2Drop`, it is manageable by modifying shell scripts

filemanager

- Two panels
- List of files and directories
 - first level - list of aliases - virtual folders leading to the providers
 - second level - list of files under the root of the file provider, ...
- Click on directory
 - go into directory
 - go up (if directory is `..`)
- Click on file
 - PDB file, triggers action to view the pdb file on the opposite panel

- on (non-webdav provider) (Dropbox), the file is downloaded first to virtual folder server -

HTTP 302 redirect is returned to webdav pointing to the location of the file on the virtual folder server.

- if such file is clicked again in future - local copy on virtual folder server is viewed
- issue if the file is changed on Dropbox provider.
- on (webdav provider) (B2DROP), the file is processed by davfs - cached locally, cache is updated if the file is changed on B2DROP

Contact

H2020-westlife-wp6 (at) mailman.muni.cz

Design and analysis docs

<http://internal-wiki.west-life.eu/w/index.php?title=D6.2>

Metadata and API

Storage provider and file/directory are basic entities of virtual folder. Dataset is another optional entity which can be stored within virtual folder.

Dataset metadata and API

File metadata and API

Virtual Machines

The repository <https://github.com/h2020-westlife-eu/wp6-vm.git> contains various configuration for development and testing purposes. Prerequisites:

1. Vagrant - tool for automation of virtual machine deployment.
 - i. For MS Windows - Download and install vagrant from <https://www.vagrantup.com/> (tested on/recommended version vagrant 1.9.6, vagrant 2.0.3 on Windows requires updated Powershell, e.g. via Windows Management Framework WMF 5.1 <https://www.microsoft.com/en-us/download/details.aspx?id=54616>)
 - ii. For Linux - use preferred package management
 - i. Ubuntu: `apt install vagrant`
 - ii. Centos(RHEL): `yum install vagrant`
2. Virtualbox - VM stack.
 - i. For MS Windows - Download and install virtualbox <https://www.virtualbox.org/wiki/Downloads>

(tested on/recommended version Virtualbox 5.1.22, for vagrant 2.0.3 tested on VirtualBox 5.2.8)
 - ii. For Linux - use preferred package management.
 - i. Ubuntu: `apt install virtualbox`
 - ii. Centos(RHEL): `yum install virtualbox`

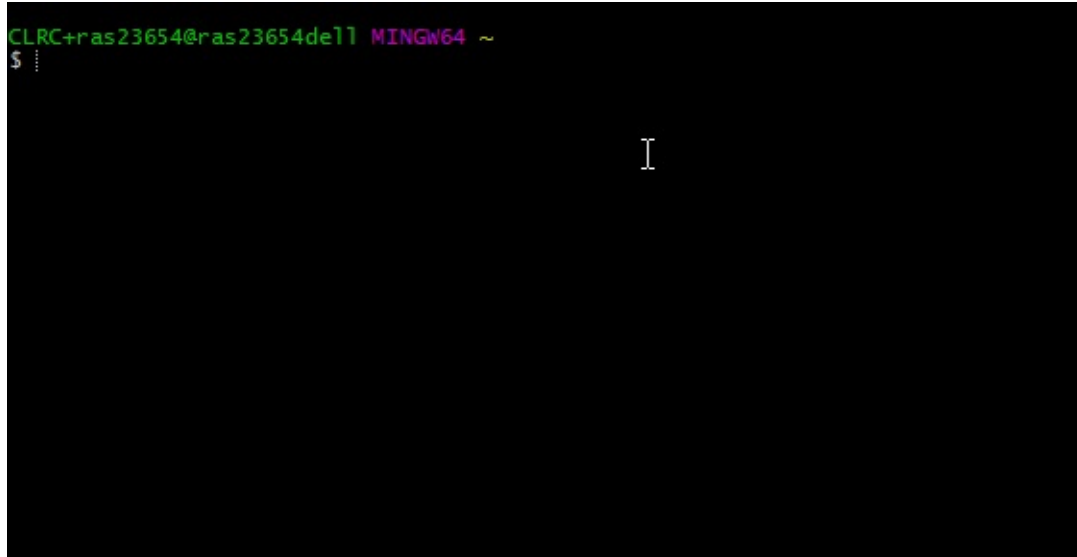
Brief instruction using Vagrant

The Vagrant tool configures and bootstraps virtual machine in Virtualbox. Brief instructions are:

```
git clone https://github.com/h2020-westlife-eu/wp6-vm.git

cd wp6-vm/[selected_config]

vagrant up
```



This clones special repository with various configuration. After successful `vagrant up`, there should be message `BOOSTRAP FINISHED, VM prepared to use` or similar, read the docs specific to the software/service.

If the application in virtual machine provides web application, you can access it with web browser using port 8080(check VagrantFile or vagrant log for exact port number being forwarded)

```
http://localhost:8080/
```

You can access the desktop of the VM by going into VirtualBox or you can log into the VM as user vagrant using

```
vagrant ssh
```

After you finish your work, you can destroy the VM and release resources by:

```
vagrant destroy
```

Detailed instruction

Download or clone metarepository [ZIP \(4kB\)](#) unzip it into some [wp6-vm directory] or clone the main repository <https://github.com/h2020-westlife-eu/wp6-vm.git>.

1. Open command-line (e.g. cmd, cygwin or terminal) and cd to directory where wp6-vm is unzipped/cloned

```
cd [wp6-vm directory]/[selected configuration]
```

These configurations are available:

Standalone from Source Codes (default)

This is based on CernVM 4.0 micro image which boots into Scientific Linux 7. Initial VM image size = 18MB, during boot and bootstrap downloads 658 MB. This is preferred option as CernVM distributomaibutes most updated SL7 with recent security updates, so either restart or `cernvm-update -a` is required occasionally.

```
cd wp6-vm/vf-standalone-src/  
vagrant up
```

Standalone from Binaries (distributed via cvmfs).

The same as above - but Virtual Folder is not compiled from sources -boots from `\cvmfs\` . This option is faster, the last stable release is distributed.

```
cd wp6-vm/vf-standalone-bin/  
vagrant up
```


Standalone from Source Codes on Scientific Linux 7 (~RHEL 7)

It is based on minimal Scientific Linux 7 - no dependency on online repositories at all. Initial VM image size = 665 MB, boot and bootstrap download 320 MB. Recommended for preparing off-line deployment.

```
cd wp6-vm/vf-standalone-src-sl7/  
vagrant up
```

Test configurations

These are currently in testing stage, not guaranteed to be working.

```
cd wp6-vm/test-...  
vagrant up
```

Base box update

Optionally, if you have used west-life VM before, you may remove previous VM and update the vagrant box cache

```
vagrant destroy  
vagrant box update
```

Deploy development branch

By default, the master branch from sources are cloned into VM and VM is booted. To change it, edit the bootstrapsources.sh file and uncomment/edit the following three lines (change 'dev' with a desired git branch):

```
# optional switch to branch
cd west-life-wp6
git checkout dev
cd ..
```

Virtual Folder enable multiuser environment

By default, Virtualfolder in VM will contain single user environment - no login is required. To enable multiuser environment with VRE, edit bootstrapsources.sh file and uncomment the following line. Default user for VF will then be vagrant/vagrant:

```
export PORTAL_DEPLOYMENT=1
```

Base Box

The following base boxes are used:

- Scientific Linux 7 with minimal GUI (~600MB), some additional packages are downloaded during bootstrap. Bootstrap takes about 4 mins.
- CernVM 4 (~18MB), after boot it will download additional 200-300 MB.

Usage

After 'vagrant up' finished, the new virtual machine can be accessed via web browser (port 8080 is by default forwarded to VM, check VagrantFile or vagrant log for exact port number)

```
http://localhost:8080/
```

Default login name to VRE is vagrant/vagrant.

Files of the current working directory of host are mounted into the guest
`/vagrant` .

You can access the guest by SSH (default port 2222 is forwarded to VM)

```
vagrant ssh
```

or access GUI in virtualbox (username/password: vagrant/vagrant).

Uninstallation - Cleaning

6. After testing you may, stop (halt) the VM:

```
vagrant halt
```

7. If you will not use the VM anymore, you can delete (destroy) the VM:

```
vagrant destroy
```

Custom installation

In order to install selected configuration to cloud environment, use bootstrap scripts from selected configuration above to install selected configuration into custom virtual machines. As the scripts above were tested on Scientific Linux 7, no or minimal changes is needed on any other RHEL 7 derivative (Centos 7.x, etc.). Some slight changes and manual steps need to be done on other OS.

Custom installation was tested on Google Cloud Compute Engine using Centos 7 and Amazon AWS using Centos 7 and in academic cloud infrastructure OpenStack and OpenNebula environment using CernVM 4 template registered at

appdb.egi.eu <https://appdb.egi.eu/store/vappliance/west.life.vm> and
<https://appdb.egi.eu/store/vappliance/d6.1.virtualfoldervm>

Release Notes

- 27/06/2017 - added variation of Vagrant scripts for different deployment type
- 03/05/2017 - works with Vagrant version 1.9.3 and below or 1.9.5 and above. Vagrant version 1.9.3 needs to use different Vagrantfile, Vagrant version 1.9.4 has bug preventing to bootstrap the VM.
- 25/11/2016 - Updated vagrant boxes to use uCernVM 2.7.7 bootloader, updated OVA images in <https://appdb.egi.eu/store/vappliance/d6.1.virtualfoldervm/vaversion/latest> and vagrant boxes, do "vagrant box update", bug fixes, consolidated initial web page and design, fixed/added background services
- 26/10/2016 - moved VagrantFile to new repository <https://github.com/h2020-westlife-eu/wp6-vm>, updated base box with uCernVM2.7.4 bootloader for CernVM 4 fixes security bug 'dirty COW' and aufs bug in kernel, <https://atlas.hashicorp.com/westlife-eu>,
 - tested on Windows 7 64 bit, vagrant 1.8.6 + VirtualBox 5.1.6, vagrant 1.8.1, 1.8.4, VirtualBox 5.0.26, note vagrant < 1.8.6 requires VirtualBox 5.0.x, doesn't require VirtualBox extension pack, download from https://www.virtualbox.org/wiki/Download_Old_Builds_5_0
 - tested on Ubuntu 14.04 LTS, (default vagrant 1.4.3 needs to be updated to 1.8.6), default VirtualBox 4.3.36 works

Repository

Reference implementation of a repository that supplies suitable metadata to the portal. This is developed within work package WP6 by the [West-Life H2020 project](#) running from 2015 to 2018. It provides application level services usable for structural biology use cases and follows [the structural biology data lifecycle](#).

Data management work package WP6 build on existing infrastructure for storing and accessing data. Full documentation is rendered in [HTML docs](#) or [PDF docs](#)

Installation guide

This guide contains information, how to install Repository into various environment.

- External registration is needed with West-Life SSO and ARIA, follow [Prerequisites](#).
- To prepare development/testing environment follow [Automatic Installation](#).
- To prepare production environment in public domain, follow [Manual installation](#).

Prerequisites

External registration is needed with West-Life SSO and ARIA for integrating single sign on and project proposal import. It allows to login into Visiting Scientist space and allows import project proposals from Instruct's database at <https://www.structuralbiology.eu>

Demo registration package can be provided upon request.

In order to integrate with West-Life SSO, you need your `sp-metadata`, `idp-metadata`, `sp_key` and `sp_cert` files.

If you have it from previous installation, then put it next to the VagrantFile - these will be reused.

If you don't have, Edit the bootstrap.sh file and change values of variables to your hostname and registered identification:

```
SP_IDENTIFICATION=http://[your public domain of repository]
```

```
SP_ENDPOINT=http://[your public domain of repository]/mellon
```

After first `vagrant up` or `bootstrap.sh`, find the generated file `sp-metadata.xml` and register it with West-Life SSO - send it to `westlife-aai@ics.muni.cz`

In order to have working ARIA integration, you need your URL of repository registered with Instruct. If you have it from previous installation, place `clientIds.php` next to the VagrantFile - this will be reused. Otherwise send a request for `client_id` and `client_secret` for the URL in the form `http://[your public domain of repository]/repository` to

<https://www.structuralbiology.eu/contact-us/>. You can continue with installation `vagrant up`. When you receive `client_id` and `client_secret`, create `frontend/ariademo/clientIds.php`.

```
<?php
// Unique client ID that lets ARIA know where your request is coming from
$client_id = 'xxx';
// keep this one on the server for security (used for generating access tokens)
$client_secret = 'yyy';
?>
```


Automatic installation

The following procedure will prepare environment in VirtualBox configured with port forwarding and shared folders automatically using Vagrant tool. It's appropriate for testing and or development installation on local workstation.

[Vagrant tool](#) and [Virtualbox](#) is required for installing Repository from source codes. Tested on Vagrant (1.9.6 and 2.0.3) and VirtualBox (5.1.30 and 5.2.8).

External registration is needed with West-Life SSO and ARIA for integrating single sign on and project proposal import. Follow prerequisites at [Prerequisites](#).

Execute following in your command line:

```
git clone https://github.com/h2020-westlife-eu/wp6-repository.git
cd wp6-repository
# (optionally unzip sp files from westlife sso and clientIds from ARIA
)
# unzip westlifespfiles.zip
vagrant up
```

After successful finish. The VM is prepared, the port 8080 is automatically forwarded to VM's port 80. To connect to VM using SSH

```
vagrant ssh
```

The backend DB is using randomly generated access credentials at `/etc/westlife/repository.key`

You may source this file by using `source /etc/westlife/repository.key`

You may restart the backend service by `service westlife-repository restart`

Manual installation from source codes

This procedure is appropriate for deploying into existing server or machine. E.g. within dedicated cloud infrastructure. It is recommended to use Scientific Linux 7.x, Centos 7.x or other RHEL 7.x derivatives.

External registration is needed with West-Life SSO and ARIA for integrating single sign on and project proposal import. Follow prerequisites at [Prerequisites](#).

Log in into the server as root and do following:

```
git clone https://github.com/h2020-westlife-eu/wp6-repository.git
cd wp6-repository
bootstrap.sh
```

Follow bootstrap.sh messages for intermittent errors. Once bootstrap.sh finishes succesfully, the server is prepared to be used. The web server is configured at <http://localhost> or at your FQDN.

User's guide

The Repository user's guide

Developer's guide

This guide covers parts suitable to maintain and implement parts of Repository. Source codes are available at <https://github.com/h2020-westlife-eu/wp6-repository>.

ARIA integration

Aria integration follows docs at <http://aria.structuralbiology.eu/docs.html>

with slight modification explained in this type of info box

The following steps needs to be performed in order to get proposals from ARIA database: 1. your application needs first generate special link with encrypted client_id and secret_id 2. when user clicks the link, first it is redirected to Instuct site to authenticate and authorize access to the Repository web app. When redirected back, it is redirected with access code and state in url parameters. 3. with access code, web application requests access token. 4. with access token, web application requests list of proposals 5. user can initiate to request proposal detail, it is obtained using access token and proposal ID (pid).

Step 1 Obtain link

Your application is hosted at [http://\[yourweb\]/index.html](http://[yourweb]/index.html). And contains e.g. following link which should initiate importing data on user request.

index.html :

```
...  
<a href="">Import data</a>  
...
```

The link is empty and needs to be filled, e.g. using the Fetch API: `ariaapi.js` :

```
...  
return this.httpClient.fetch('/accessToken.php')
```

```
//expecting JSON with {'url':'http://structuralbiology.eu/
//  authorize?client_id=123&state=456&response_type=code'}
.then(response => response.json())
.then(data => {
    return // use data.url to add it into <a href="${data.url}">ge
t access token</a>
})
...
```

In order not to reveal `client_id` and `client_secret` , the following code reside on `server/backend` side only of your application at `http: [yourweb]/accessToken.php`:

`accessToken.php` part 1 :

```
<?php
//$client_id = 'abc123...';
//$client_secret = 'ABC123...';
$headerHost = $_SERVER['HTTP_HOST'];
$OAservice = 'https://www.structuralbiology.eu/ws/oauth/';
// define a unique state string for security
session_start();
if(!isset($_SESSION['OAsstate'])) {
    $_SESSION['OAsstate'] = rand(1000000, 9999999);
}
error_log("session state:".$_SESSION['OAsstate']);
// if we have code, then retrieve token and set it to cookie
if(isset($_GET['code']) && isset($_GET['state'])) {
    ...
}
//otherwise create link
else {
    header("Status: 200 OK");
    header('Content-type: application/json');
    echo '{"url":"' . $OAservice . 'authorize?client_id=' . $client_id . '&state
=' . $_SESSION['OAsstate'] . '&response_type=code"}';
    exit;
}
?>
```

This returns link which can be bind to the HTML by your preferred Javascript framework (Aurelia,Angular,React,Jquery,clean Javascript using DOM api only). So the link is now prepared `index.html` :

```
...  
<a href="http://structuralbiology.eu/authorize?client_id=123&state=456  
&response_type=code">Import data</a>  
...
```

Step 2 Obtain Access Code

When the link is clicked by the user, it redirects to Instruct site and user is asked to authorize access to his data to your application at [yourweb]. If succesfull, Instruct site will redirect user's browser back to your web using parameters in URL, e.g. [http://\[yourweb\]/index.html?code=789&state=012](http://[yourweb]/index.html?code=789&state=012) Therefore your web application needs to detect parameters in url, e.g. by following code `index.js`

```
//  
const getParams = query => {  
  if (!query) {  
    return {};  
  }  
  
  return (/^[?#]/.test(query) ? query.slice(1) : query)  
    .split('&')  
    .reduce((params, param) => {  
      let [key, value] = param.split('=');  
      params[key] = value ? decodeURIComponent(value.replace(/\+/g, ' ')) : '';  
    }) : '';  
  return params;  
}, {});  
...  
this.params=getParams(window.location.search.substring(1));  
if (this.params.code && this.params.state)  
  this.ariaapi.getAccessToken(this.params.code,this.params.state);
```


Step 3 Obtain Access Token

The parameters `code` and `state` are used to obtain access token from ARIA proxied by the `accessToken.php` service `ariaapi.js`

```
getAccessToken(code,state){
    return this.httpClient.fetch("/accessToken.php?code="+code+"&state="+state)
        .then(data => {
            this.accesstoken=JSON.parse(data.response);
            return this.accesstoken;
        })
        ...
}
```

PHP7 does not allow credentials in the body of a request. Thus client id and secret is sent as HTTP headers `curl_setopt($ch, CURLOPT_USERPWD, $client_id.':'.$client_secret);`.

`accessToken.php` part 2

```
...
if(isset($_GET['code']) && isset($_GET['state'])){
    ...
    // manually build OAuth packet - this can be replaced for an off-she
    lf solution
    $OApacket = array(
        'grant_type' => 'authorization_code',
        'code' => urlencode($code)
    );
    // start curl connection
    $ch = curl_init();
    ...
    // add OAuth packet and define HTTP POST
    curl_setopt( $ch, CURLOPT_POST, true );
    curl_setopt( $ch, CURLOPT_POSTFIELDS, http_build_query($OApacket) );
    curl_setopt( $ch, CURLOPT_USERPWD, $client_id.':'.$client_secret )
```

```

;
// run curl and capture output
if(!$content = curl_exec( $ch )) { ...
    exit('Could not fetch access token');
}
curl_close ( $ch );
// convert JSON into object
if(!$authorization = json_decode($content))
    exit('JSON parsing error');
if($authorization->access_token){
    // we will store access tokens as cookies for this demo
    setcookie('access_token', $authorization->access_token, time()+$au
thorization->expires_in);
    // this refresh token actually has an expiry of 28 days, however f
or the demo it's useful to expire within a single day
    setcookie('refresh_token', $authorization->refresh_token, time()+60
*60*24*1);
    $_COOKIE['access_token'] = $authorization->access_token;
    //now exit - cookie set - no other content is needed
    header("Status: 200 OK");
    header('Content-type: application/json');
    echo ($content);
} else {
    header("Status: 404 Not Found");
    header('Content-type: application/json');
    echo ($content);
}
exit;
...
?>

```

Step 4 Get Proposal list

As the web app now has the access_token, it can use it to request ARIA directly (not via accessToken.php). To get user's proposallist and proposal details:

index.js

```
this.ariaapi.getProposalList(accesstoken).then(list =>{ //show list})
```

`ariaapi.js`

```
...
  getProposalList(accesstoken) {
    return this.httpClient.fetch("https://www.structuralbiology.eu/w
s/oauth/proposallist?access_token="
+this.accesstoken.access_token)
    .then(response => response.json())
    .then(data => { return data })
  }
...
```

Note, the HTTP GET request is used to obtain the proposallist

Step 5 Get Proposal list

`index.js`

```
this.ariaapi.getProposal(pid,accesstoken).then(list =>{ //show list})
```

`ariaapi.js`

```
...
  getProposal(pid,accesstoken) {
    return this.httpClient.fetch("https://www.structuralbiology.eu/w
s/oauth/proposal?pid="+ pid+"&access_token="
+accesstoken.access_token)
    .then(response => response.json())
    .then(data => { return data })
  }
...
```

Note, the HTTP GET request is used to obtain the proposal detail

Full working code

Working sample code is in github.com/h2020-westlife-eu/wp6-repository/tree/master/frontend