
DELIVERABLE

D6.2 Expansion of the SERIES database D6.3 Updated version of SERIES database

Work package	WP6 (NA4: Networking experimental seismic engineering databases (SERIES))
Lead	JRC, UPAT
Authors	Pierre Pegon, (JRC) Stathis Bousias, (UPAT) Ignacio Lamata Martinez, (ARHS Developments) Francisco Javier Molina, (JRC) Sakis Ntorumas, (UPAT) Christos Sintoris, (UPAT)
Reviewers	
Approval	[Management Board]
Status	Final
Dissemination level	Public
Delivery deadline	31.10.2018
Submission date	30.10.2018
Intranet path	DOCUMENTS/DELIVERABLES



Table of Contents

Summary	3
1 Introduction	3
2 The graphical user interface	3
3 Support for input of data.....	14
4 Distribution of the new tools.....	18
5 Modifications in DAP website and back-end	18
6 Annex I: Celestina Data Viewer Installation Manual	21
7 Annex II: Celestina Data Viewer User Manual	56

Summary

This document joins deliverables D6.2 "Expansion of the SERIES database" and D6.3 "Updated version of SERIES database" of the SERA project. Maintenance and expansion of the SERIES database is one of the two main objectives of SERA work package 6 "Networking experimental seismic engineering databases (SERIES)". The actions taken to achieve this objective are described in the following and include the creation of the graphical user interface (GUI) and its user manual, the development of tools that assist in the creation of experimental data, and the modifications in the Data Access Portal (DAP) website and back-end. The installation manual and the user manual are given in Annexes

1 Introduction

During the SERIES project (2009-2013), a common computing infrastructure was developed to address the lack of interoperability between the European earthquake engineering institutions. This infrastructure was implemented as a network of distributed data sources that allows access to the SERIES European laboratories' data through a single, centralised user interface, while enabling data integration between the different SERIES institutions. This automated integration of experimental results was one of the major achievements brought by SERIES in terms of experimental data management.

The distributed infrastructure with a centralised access served different purposes:

- Provide experimental results to the earthquake engineering community in a uniform manner. Thanks to this, end users can collect multi-source information from a single place. The distributed nature of the data is invisible to the end user, whose experience should be similar to accessing a single data repository.
- Enable the different SERIES institutions to work autonomously and be in control of their own data.
- Reduce the technological gap between each laboratory part of the SERIES network.

However, at the end of the SERIES project it was patent that the infrastructure lacked some features to take full advantage of the data management, especially those to facilitate the visualisation and acquisition of experimental data at the local nodes in a user-friendly way.

The SERA project intends to fulfil this lack by enhancing the current SERIES infrastructure with the creation of a user-friendly graphical interface and the development of IT tools to facilitate the input of experimental data. This document describes the results of these enhancements, which are based on targeted modifications of the local nodes.

The rest of the document is organised as follows:

- Section 2 presents the graphical user interface that allows the visualisation of experimental data in a user-friendly way.
- Section 3 describes the tools and features that assist in the creation of experimental data.
- Section 4 summarises the actions taken for the distribution of the new tools.
- Section 5 describes the modifications in the SERIES Data Access Portal.

2 The graphical user interface

The graphical user interface (GUI) enables end users to visualise data in a more appealing and efficient way. Instead of having to navigate technical tables of structured data in a database format, the GUI presents users with data in a more coherent manner, without the need for technical knowledge,

database management software specifics or data model implementation details. The GUI comes with the following documentation:

- An installation manual, that explains how to install and configure the whole system. This manual was sent to SERA partners on the 26th October 2018.
- An extensive user manual, that explains how to use the GUI as a user. This manual is quite detailed, and it is accessible from the GUI itself¹. Please, refer to it to obtain further information about the user interface and how to use it.

In the following Figures 1 - 11, some GUI screens are depicted.

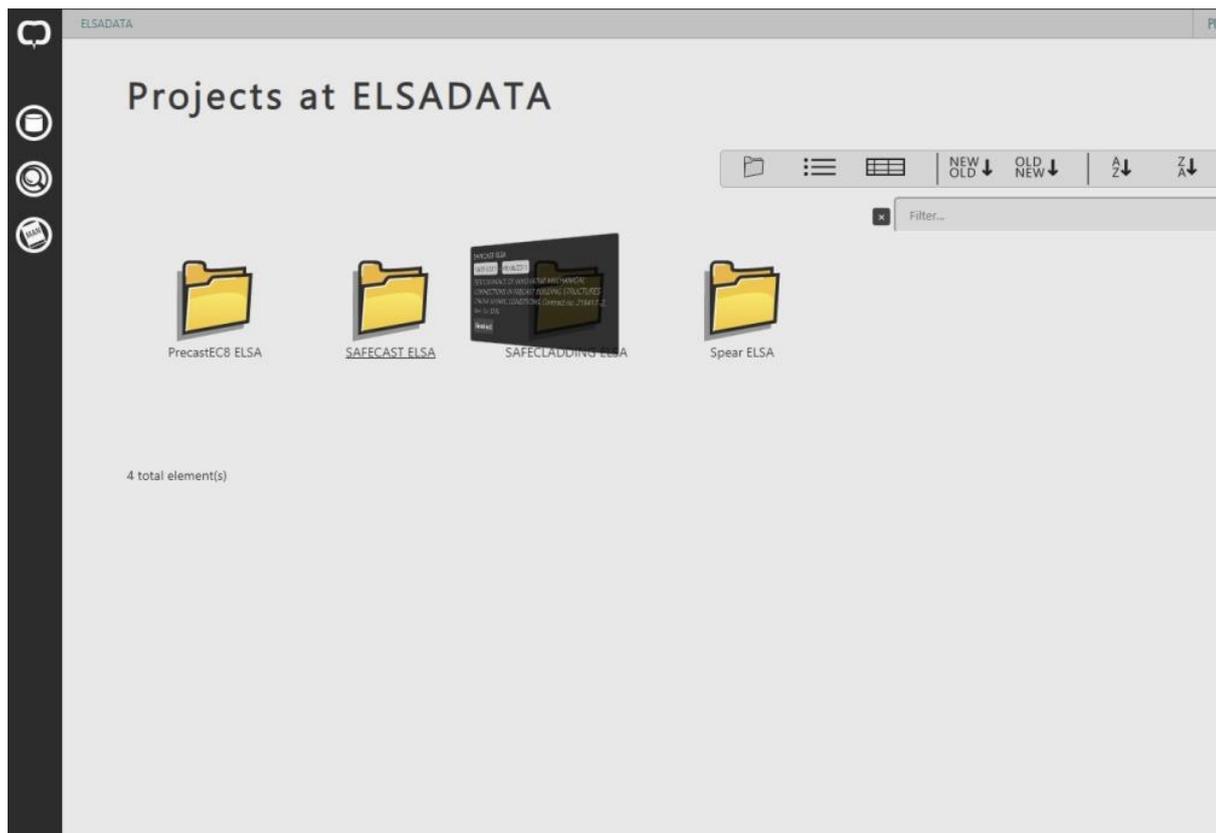


Figure 1: The list of projects available in a repository

¹ Accessible at https://YOUR_HOSTNAME/CelestinaDataViewer/manual.xhtml or from the menu icon:  Alternatively, you can check the manual in Annex II

ELSADATA > Spear ELSA PUBLIC

Project Spear ELSA

Spear ELSA
16/01/2004 - 16/03/2005 Finished

Abstract
Seismic Performance Assessment and Rehabilitation, G6RD-2001-00525 + HPRI-1999-00059

Keywords

reinforced concrete structure dynamic testing buildings fiber reinforced composite retrofitting cyclic testing

pseudodynamic bi-directional

Project map

The project map displays a network diagram. At the bottom center is a blue oval labeled 'Spear ELSA'. Two orange ovals, '1st Retrofit' and '2nd Retrofit', are connected to 'Spear ELSA'. '1st Retrofit' is further connected to five green circles labeled G13, G14, G15, G16, and G17. '2nd Retrofit' is connected to three green circles labeled G21, G22, and G23.

Figure 2: Details of a specific project

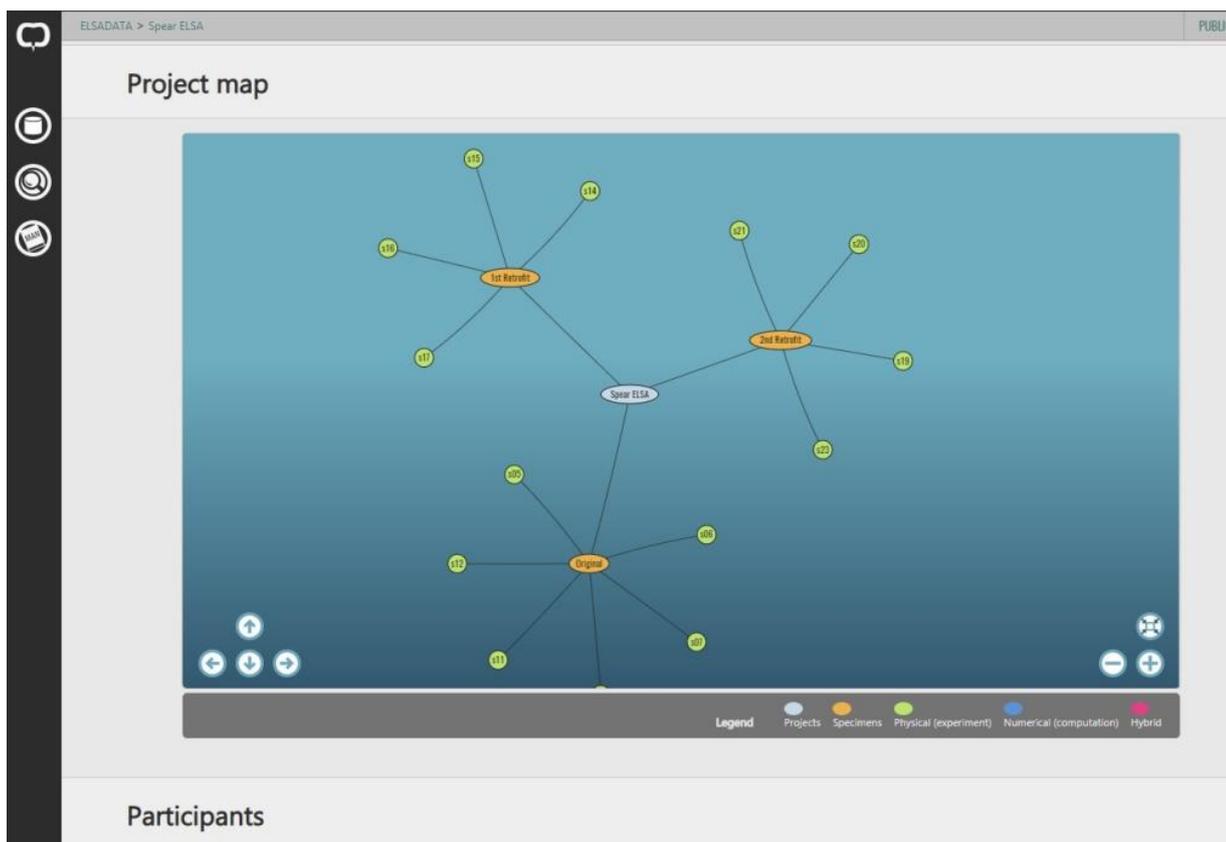


Figure 3: Visual representation of the relations between a project, the specimens tested and the experiments conducted

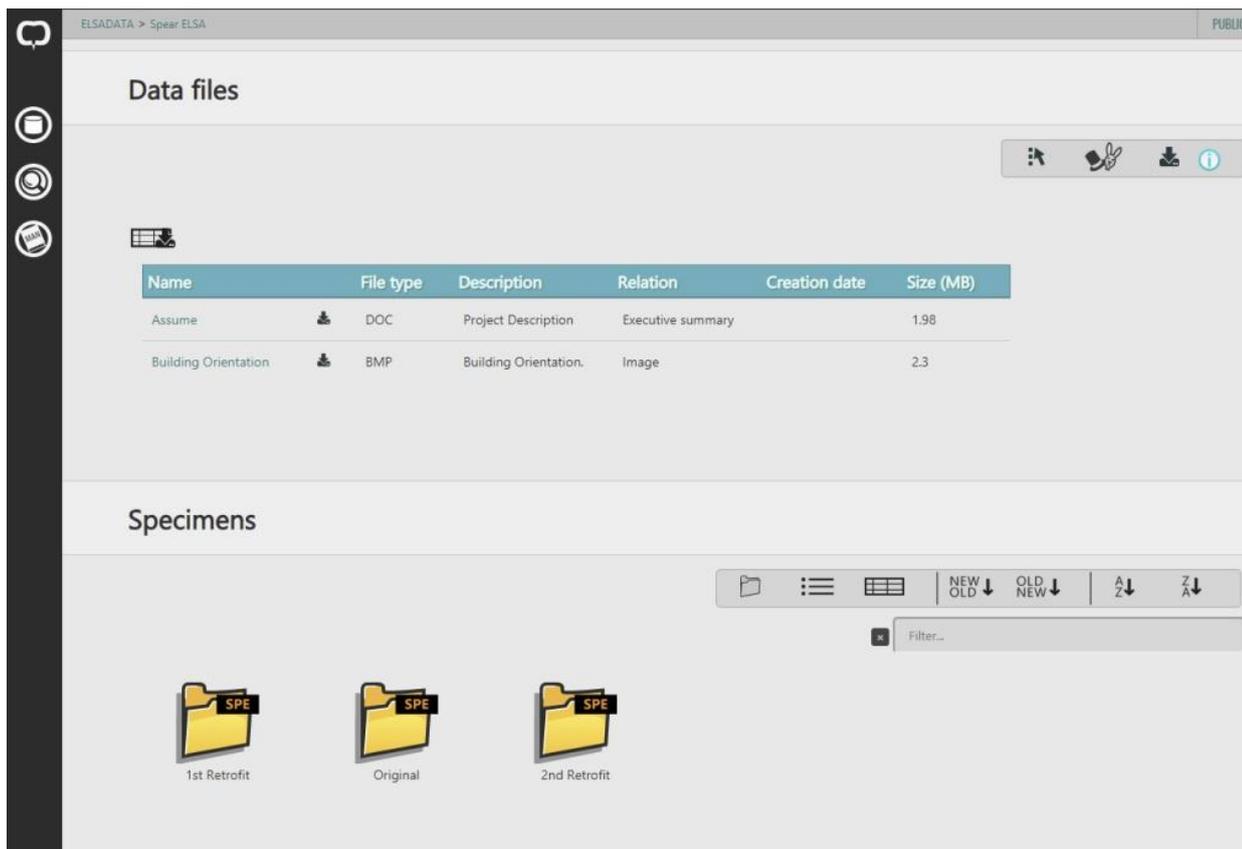


Figure 4: List of large files related to a project

ELSA DATA > Spear ELSA > 407 PUBLIC

Output

- IDENre - Source: hybrid system | Sampling: resampled | Elaboration: identified 37 signals

17 hidden
4 selected

Name	Description	Magnitude	Unit	Time	Notes(1)	Notes(2)	Size (MB)
004	Spatial-Model	Frequency	Hz		Mode 4	RefD w=1000	0.0000114
018	Spatial-Model	Frequency	Hz		Mode 9	MeaD w=1000	0.0000114
019	Spatial-Model (Damping)	Ratio	Dimensionless		Mode 1	RefD w=1000	0.0000114
020	Spatial-Model (Damping)	Ratio	Dimensionless		Mode 2	RefD w=1000	0.0000114
021	Spatial-Model (Damping)	Ratio	Dimensionless		Mode 3	RefD w=1000	0.0000114
022	Spatial-Model (Damping)	Ratio	Dimensionless		Mode 4	RefD w=1000	0.0000114
023	Spatial-Model (Damping)	Ratio	Dimensionless		Mode 5	RefD w=1000	0.0000114
024	Spatial-Model (Damping)	Ratio	Dimensionless		Mode 6	RefD w=1000	0.0000114
025	Spatial-Model (Damping)	Ratio	Dimensionless		Mode 7	RefD w=1000	0.0000114
026	Spatial-Model (Damping)	Ratio	Dimensionless		Mode 8	RefD w=1000	0.0000114
027	Spatial-Model (Damping)	Ratio	Dimensionless		Mode 9	RefD w=1000	0.0000114
028	Spatial-Model (Damping)	Ratio	Dimensionless		Mode 1	MeaD w=1000	0.0000114
029	Spatial-Model (Damping)	Ratio	Dimensionless		Mode 2	MeaD w=1000	0.0000114

Figure 5: List of signal data generated in an experiment. Some of the signals are selected for download or visualisation

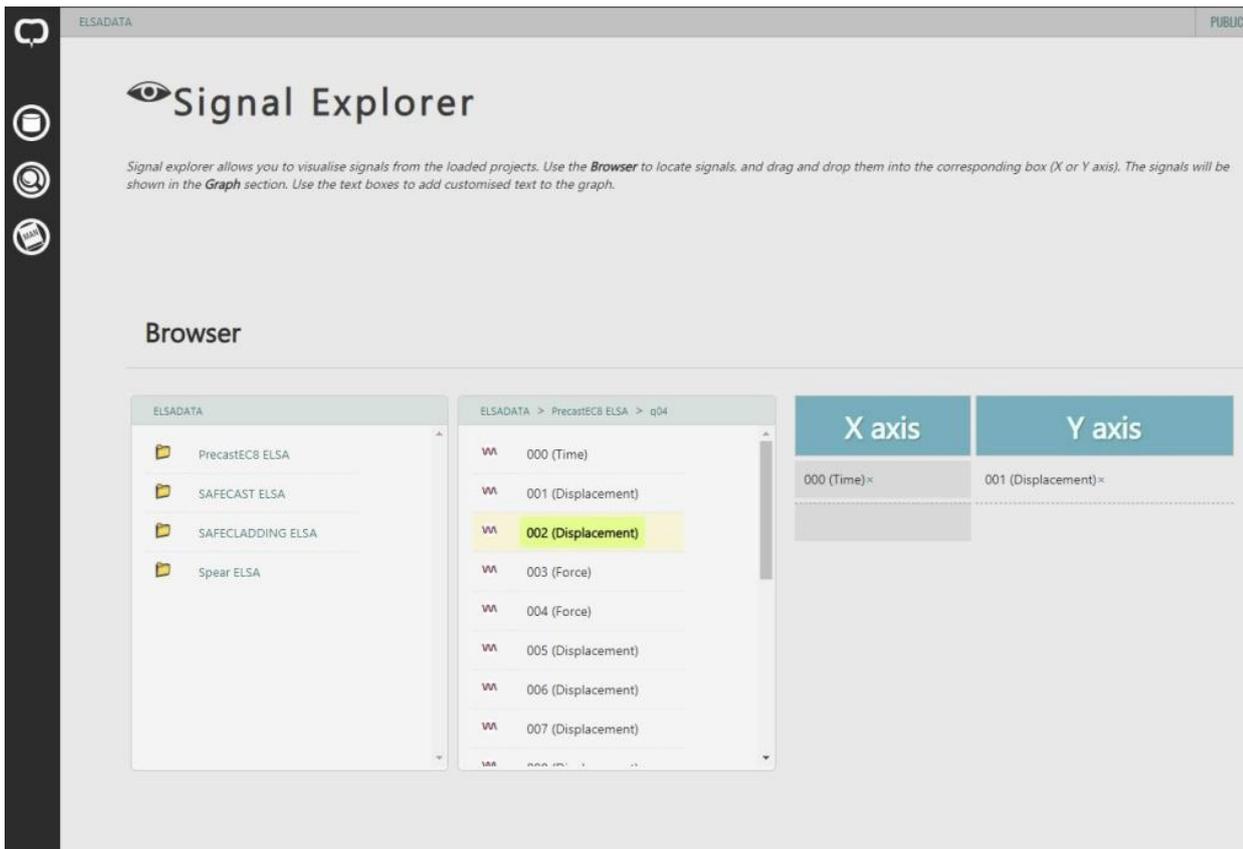


Figure 6: Signal Explorer, a tool to navigate signal data in the different projects and plot them



Figure 7: Plot of some signals data. The visualisation of the graph is interactive, allowing zooming, exportation as an image, hiding specific parts, etc.

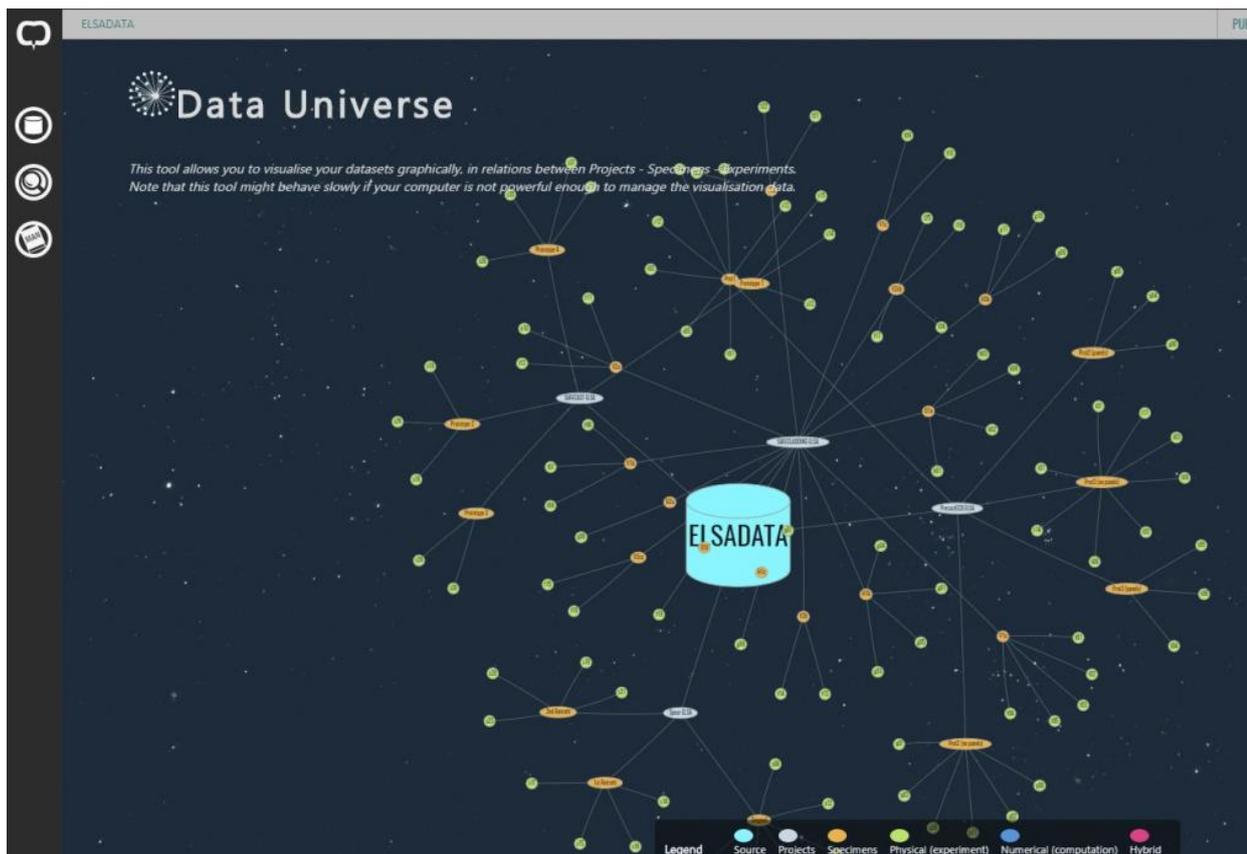


Figure 8: A full visualisation of the data of the whole repository as a network of nodes

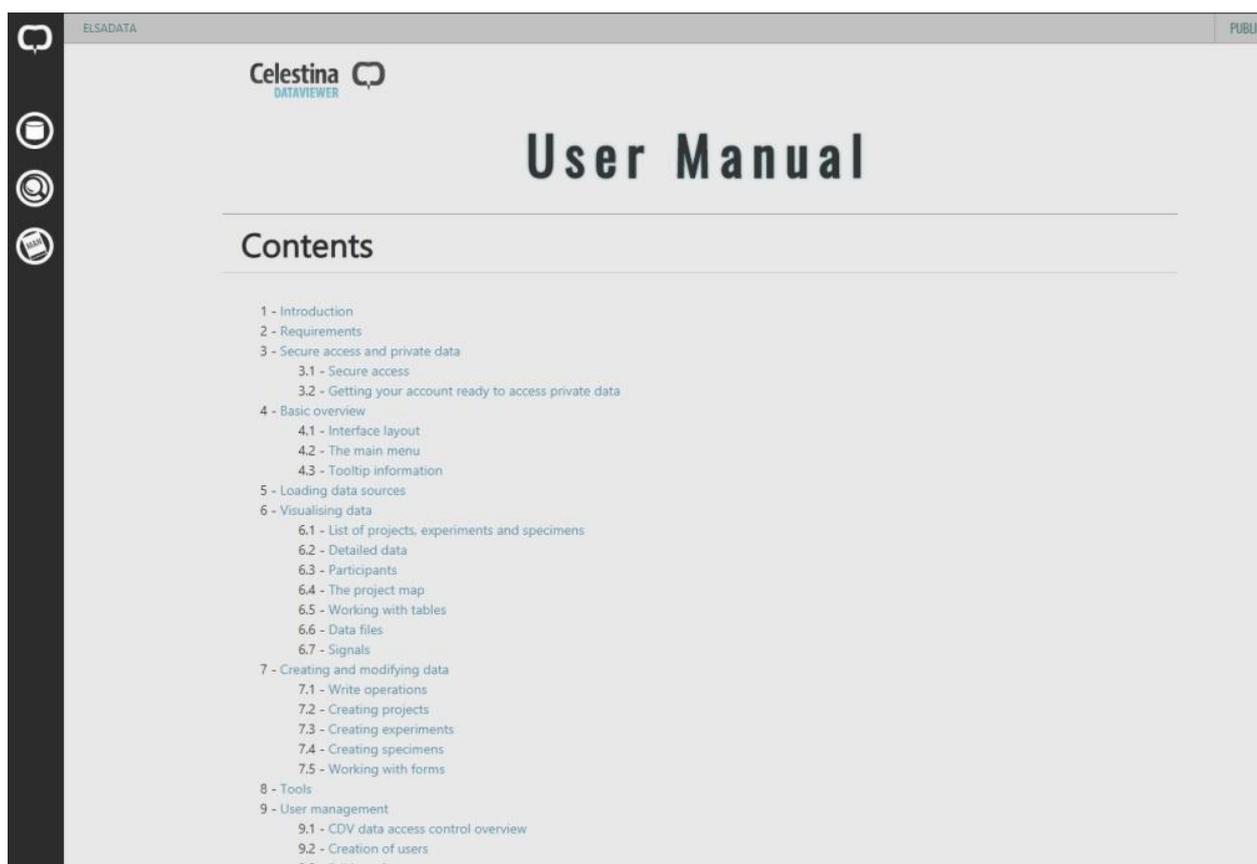


Figure 9: The user manual, which is contained within the GUI itself

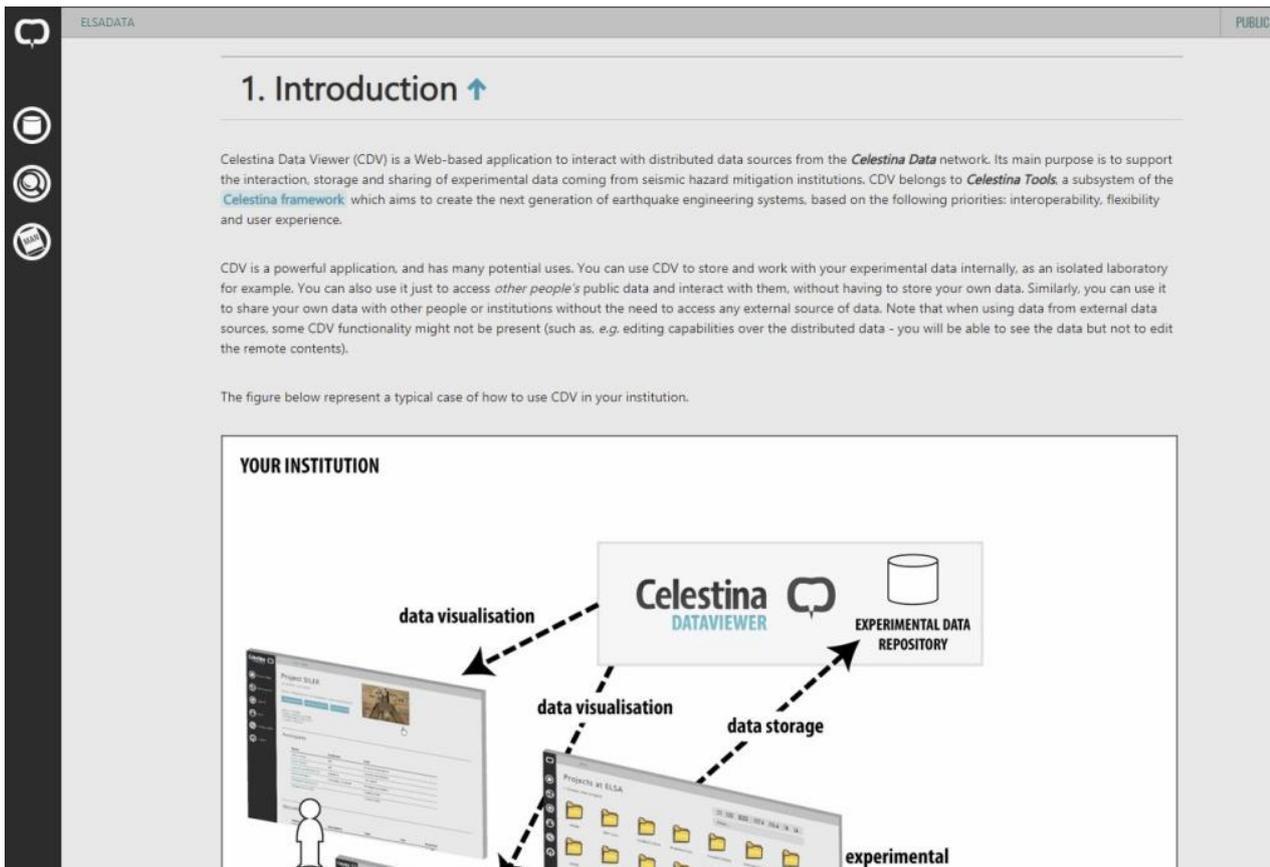


Figure 10: The user manual is quite detailed and provides information about every aspect of the GUI

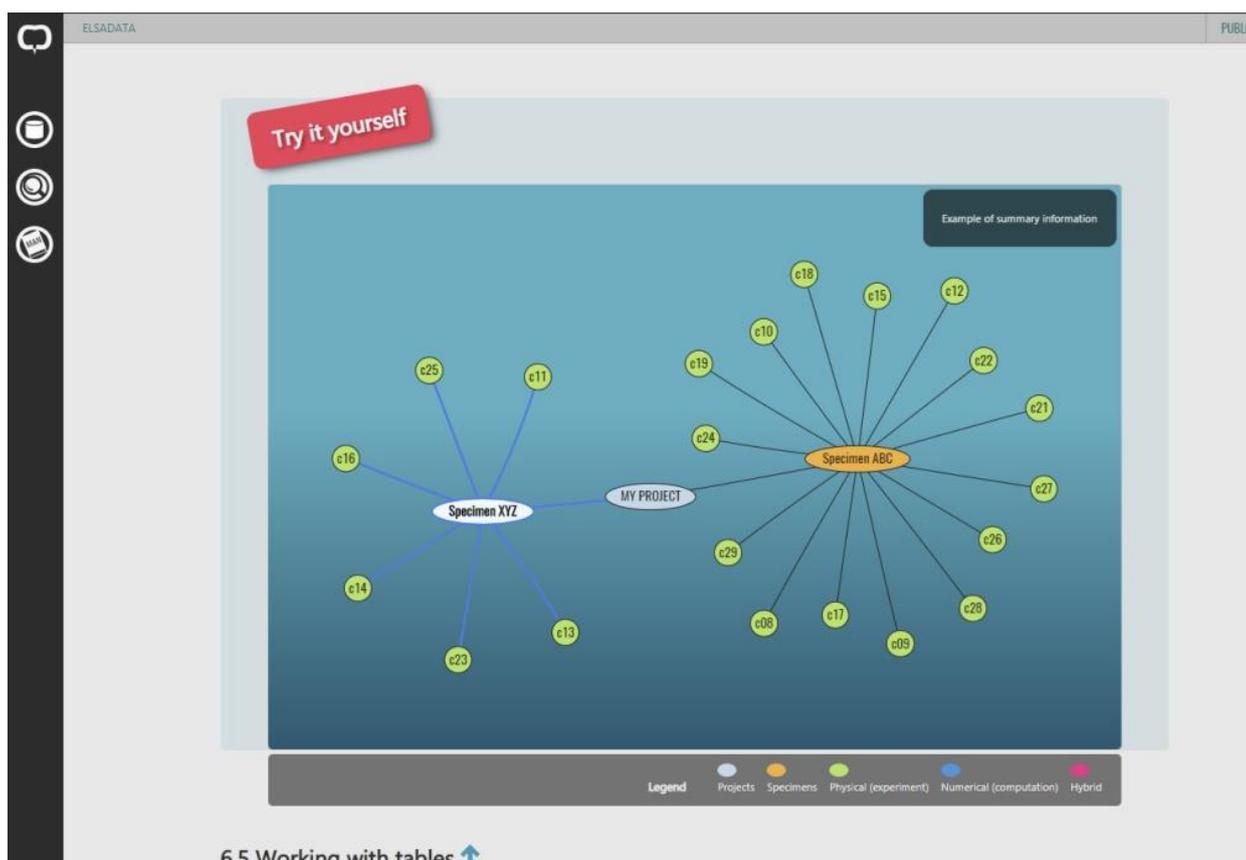


Figure 11: One of the features of the user manual is that it is interactive, so it allows users to interact with the different features of the GUI in an easy way, directly from the manual

3 Support for input of data

The tools that support the input of data come in two different components:

- **GUI based.** The graphical user interface contains different functionality that simplifies the input of data. Firstly, most of the information is collected by means of forms, as shown in Figure 12. To simplify the process of file uploading, a simple drag and drop functionality exists, as depicted in Figure 13. Users only have to click on the files that they want to upload and drag them to the box to have them uploaded. In case of single files for specific data (e.g. a logotype icon for a project), dialogs are used, as shown in Figure 14.
- **API based.** While many tasks are easily accomplished with the GUI based component, this component requires interactive human intervention. Herein, repetitive tasks are not very well suited for the GUI component, since they tend to be tedious and error prone. The API based component is a better choice for this situation. The API (Application Programming Interface) exposes a set of methods that can be called programmatically to manage the content of a data repository. Herein, programs in MATLAB, Python, C or any other language can be created to automate the process of input of data. This supports the creation of local scripts that collect experimental data coming from some laboratory activity, and automatically upload the relevant results in the local repository. Figure 15 depicts a user using MATLAB to interact with the application.

These components will enable users to upload new data into their respective repositories.

ELSADATA > Spear ELSA ADMIN INI

Update Project *Spear ELSA*

Name

Short name

Purpose

Description

Start date

End date

Keywords +

reinforced concrete structure × dynamic testing × buildings ×

Figure 12: Form to collect project information

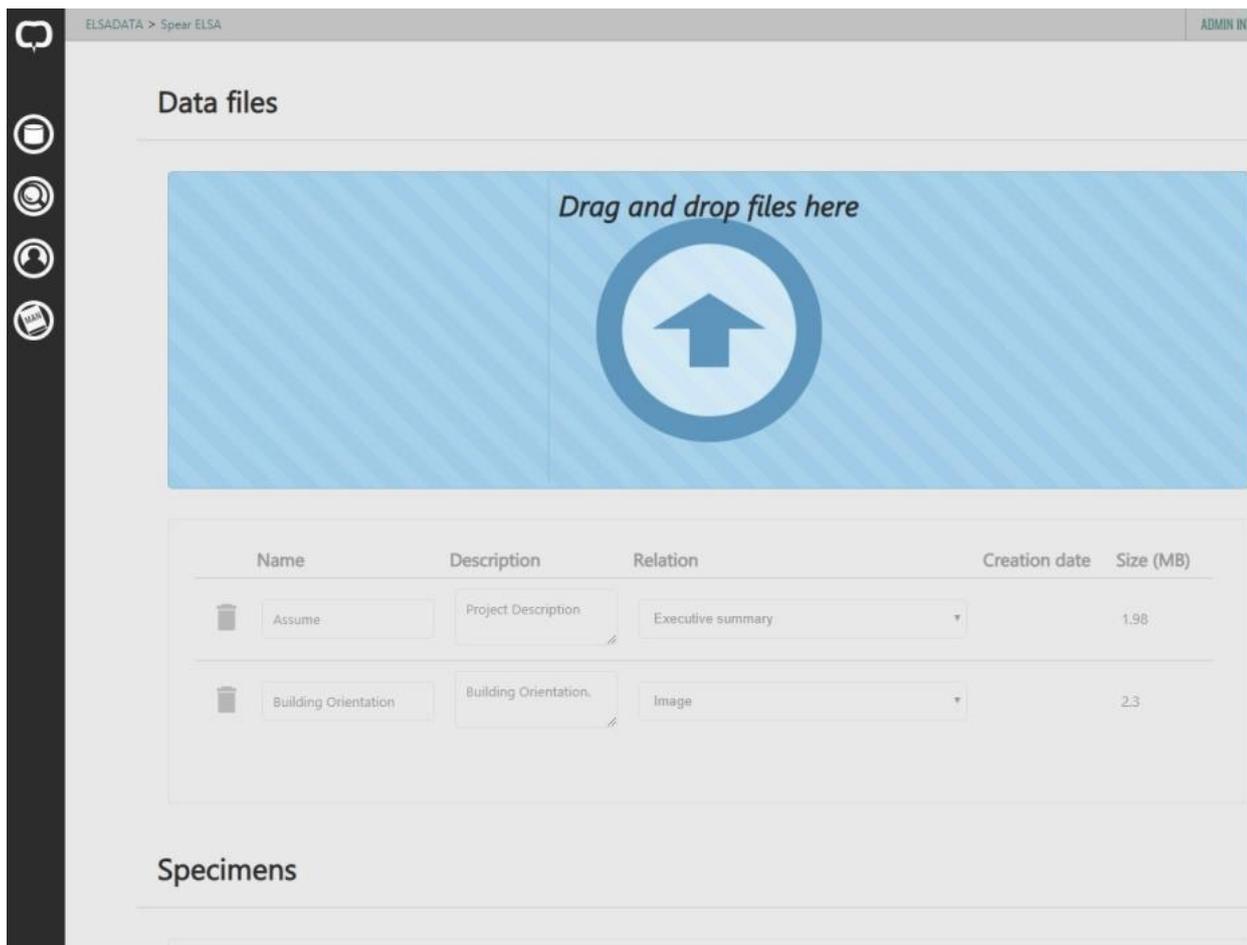


Figure 13: Drag and drop functionality to upload multiple files

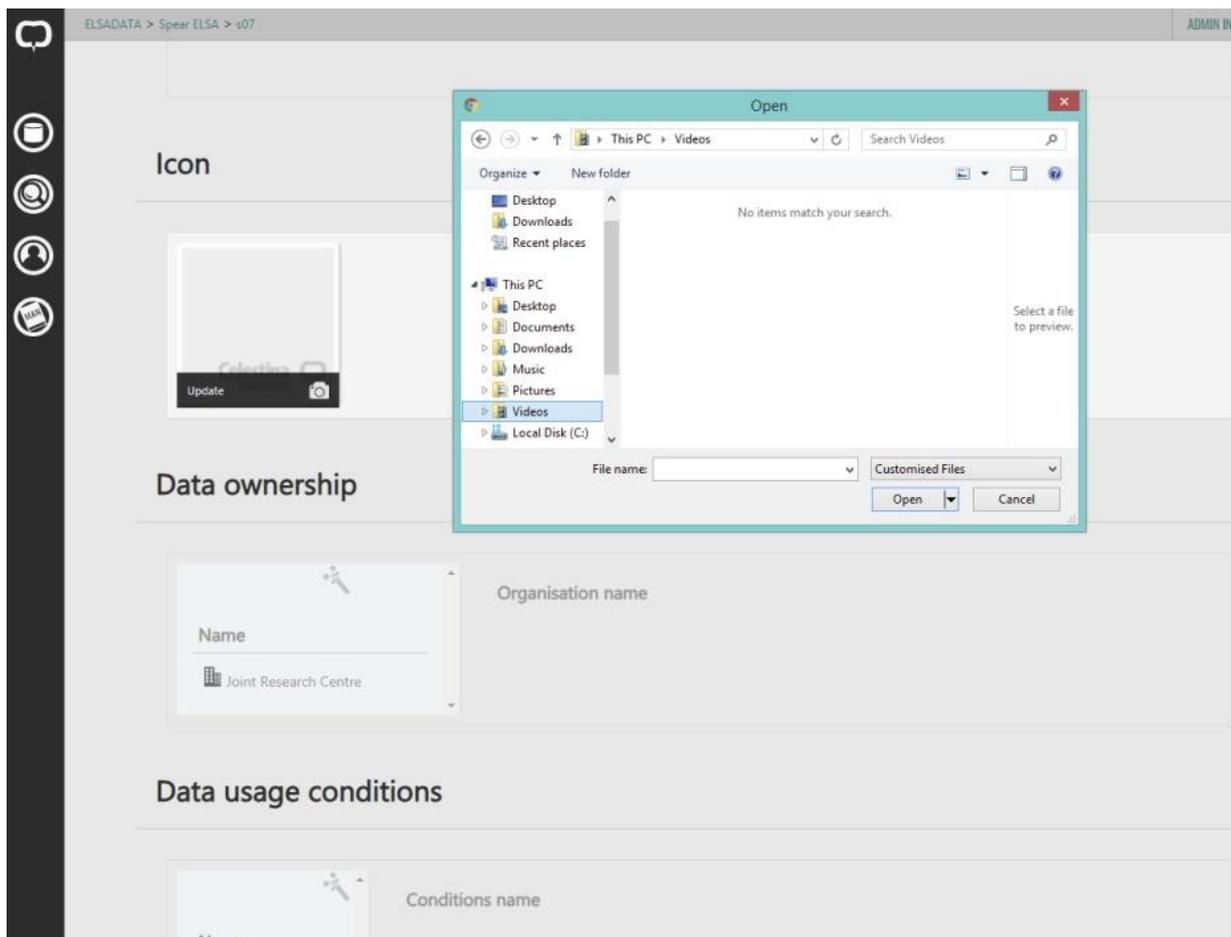


Figure 14: Single file upload, such as the icon for a project

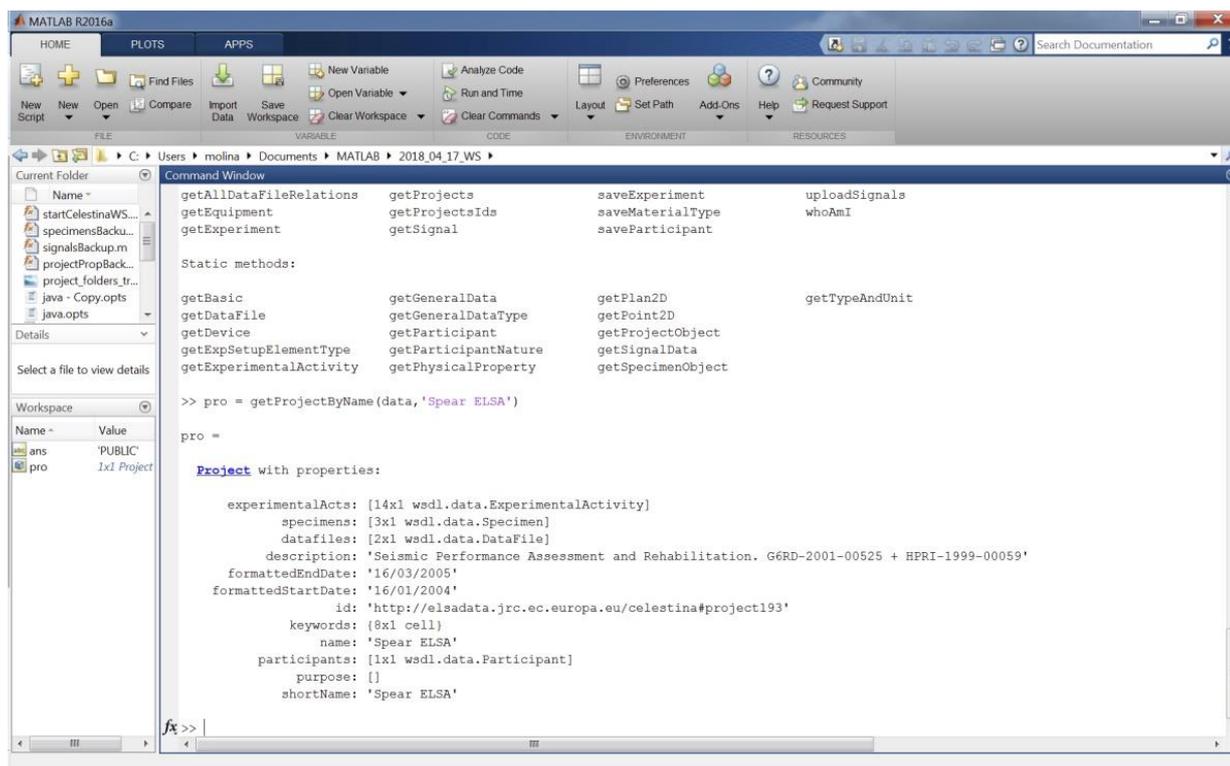


Figure 15: MATLAB being used to interact with the application

4 Distribution of the new tools

The distribution of the GUI and its associated software has been designed and planned to be as simple as possible, so the installation and configuration instructions have been reduced to the minimum for the end users. Two main actions have been taken to accomplish this:

- The system is delivered as a "black box", where most of the software has been preinstalled for the user. This box is implemented as a virtual machine, that has been created and configured for the sole purpose of running the GUI and the tools to support the input of data. This approach greatly reduces the work that end users have to do to run the new tools.
- In order to simplify the configuration process of the user interface, a piece of software called *cdvconf* has been developed. The aims of this tool are: (i) Abstract end users from technical details, (ii) Greatly simplify the tedious and error prone process of setting the right parameters in all configuration files and running the right configuration commands correctly, (iii) Provide a clean way to customise a GUI instance at each potential SERA institution, and (iv) Establish the infrastructure to support (hassle-free) future updates on the tools and other administrative tasks.

All details about this virtual machine-based system and how to use the *cdvconf* tool can be found in the Installation Manual, which is provided as an annex of this document.

5 Modifications in DAP website and back-end

A number of re-design activities have been undertaken in order to maintain and update the Data Access Portal (Series DAP). These activities relate both to the DAP website and to its corresponding database (SERA db). The purpose of these changes was to lightly update the protocol of communication between the DAP back-end and the SERIES partners, specified in Web Services Description Language (WSDL) and

to fix some errors. In order to implement the updated WSDL, the following modifications were implemented and tested in various modules, during this period:

- SERIES Database:
 - The types of all localIDs fields in all tables (project, specimen, compexp, personnel, video, materials, etc.) within the SERA database were changed from integers to strings.
 - A bug was fixed in the personnel management table, which had led to multiple errors. The bug allowed duplicate insertion of personnel data.
- Web service for retrieving lab data:
 - Modifications were implemented and tested in the source code of the web service. The purpose was to update the web service so that it communicates correctly, in accordance with the updates that were implemented in the SERIES database.
 - The new WSDL was implemented. The web service can now retrieve lab data from partners using the new WSDL.
 - Some bugs within the source code of the web service were fixed. These bugs were causing unexpected errors and wrong handling of data that were received from other partners. (e.g., keywords, signals, etc.).
- DAP website:
 - Modifications were made in the download function of the DAP website. The purpose was to modify the URL that is provided for downloading project data, so that the certificate's signature is appended in the download URL.

Liability claim

The content of this publication does not reflect the official opinion of the European Union. Responsibility for the information and views expressed in the therein lies entirely with the author(s).

6 Annex I: Celestina Data Viewer Installation Manual



Installation manual

v. Oct 2018

Record of changes

Version	Author	Description
Sep 2018	Ignacio Lamata Martinez	Initial version
Oct 2018	Ignacio Lamata Martinez	Some corrections. Additional corrections coming from Dionysis Biskinis testing

1 Introduction

This informal manual describes the steps that have to be taken in order to install the Celestina Data Viewer application (from now on, CDV). The process is fairly simple thanks to these two factors:

- CDV is delivered inside a Virtual Machine. This means that the application is distributed in a "box" containing a pre-installation of everything that is needed, which greatly simplifies the installation and configuration process for you.
- A user-friendly tool has been developed to facilitate the configuration of your specific CDV instance, and the upgrading of software in the future. This specific configuration cannot be preinstalled, since you need to customise your own instance with your own system name, your own passwords, and so on.

The ultimate goal of these is to improve your user experience by making the tedious process of installation and configuration as easy as possible. Thanks to the use of a virtual machine, the installation of software that you will have to do has been reduced to the minimum.

2 Using CDV

CDV can be used in many scenarios, and its installation will vary slightly depending on how it is used. Herein, a small compendium of "configurations" will be shown in this section, remarking the differences during the installation process.

CDV can be used to store and work with your experimental data internally, as an isolated laboratory for example, as depicted in Figure 1. You can also use it just to access *other people's* public data and interact with them, without having to store your own data. Similarly, you can use it to share your own data with other people or institutions without the need to access any external source of data. Note that when using data from external data sources, some CDV functionality might not be present (such as, *e.g.* editing capabilities over the distributed data - you will be able to see the data but not to edit the remote contents). The CDV is installed on one of your institution computers and it works as an application that your institution can use internally to store experimental results obtained from experimental testing in a laboratory. In this way, CDV supports your personnel to store results in a formalised way and to visualise results in a user-friendly manner. CDV is ready to be used with just a Web browser but it can also communicate with your applications (*e.g.* Matlab) or your own programs created in Java, C++, Python, etc. In this way, CDV provides a complete solution for your institution to manage experimental data.

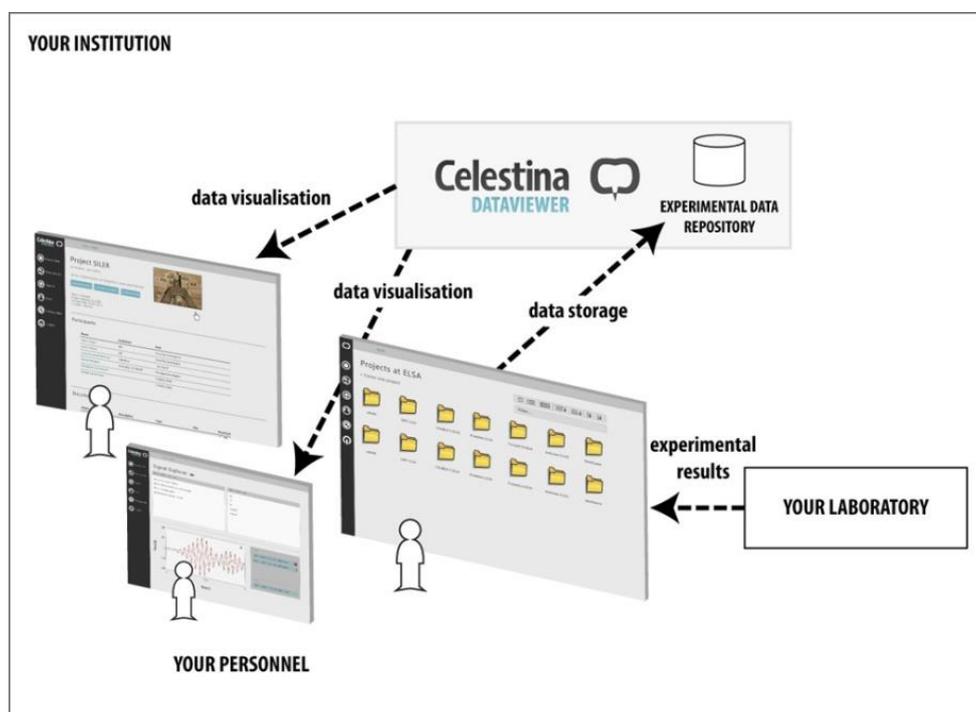


Figure 1. Typical use of CDV in a single organisation

You can go one step further and open the access to the CDV to other people around the world, so they can access your data too. This access can be limited to be read-only for external users, and to

access only a sub-set of your data set, so they can visualise and interact with some data but not write data into your repository. Of course you can open it to some external users so they can also create content in your repository. This situation is depicted in Figure 2. Note that even though actors are represented with a person icon, they can be people or systems. For CDV, there is no distinction between dealing with humans and with computers. The installation of CDV follows the exact same steps as in the previous configuration of Figure 1, with the exception that your IT department will have to prepare the network devices (such as routers or firewalls) to enable external access to your CDV instance.

As soon as you use CDV, your data will be stored in the Celestina Data format. It is easy for your institution to become a Celestina Data node if desired. This means your data will be part of the Celestina network, so they will be publicly announced for the rest of the nodes. Of course, it is encouraged, but not necessary, to share your data if you want to access data from other Celestina Data nodes. So you could obtain data from other nodes while keeping all your data private for your institution. This situation is depicted in Figure 3. The installation process for this configuration will not be described, since the Celestina network is still under development and, therefore, these features are not currently available.

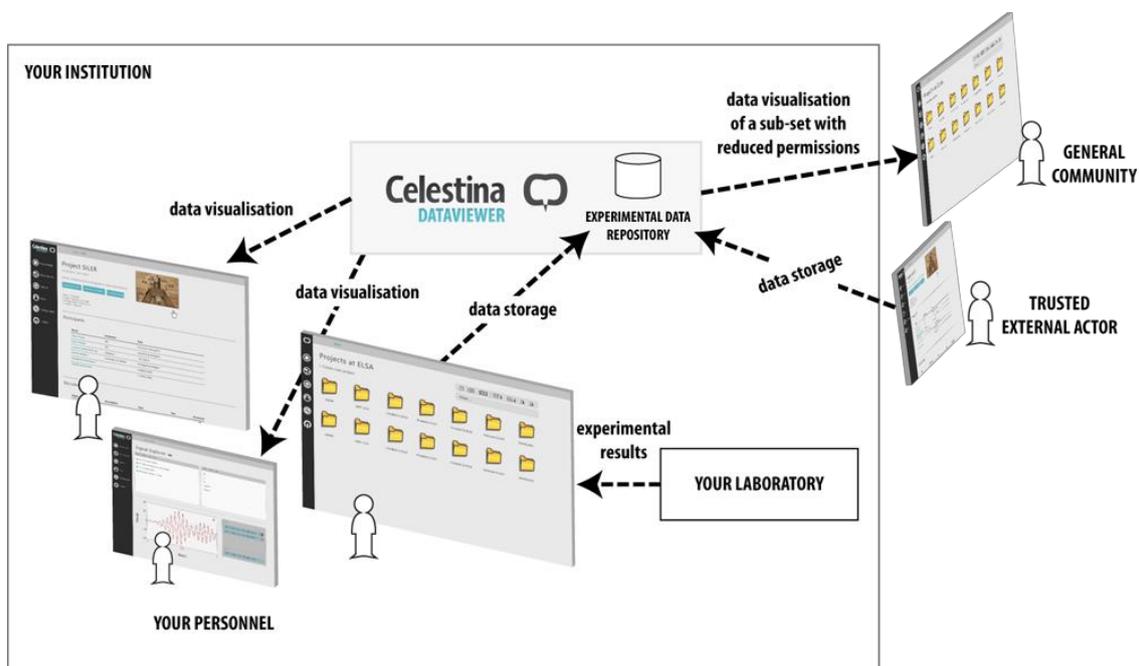


Figure 2. CDV being accessed externally

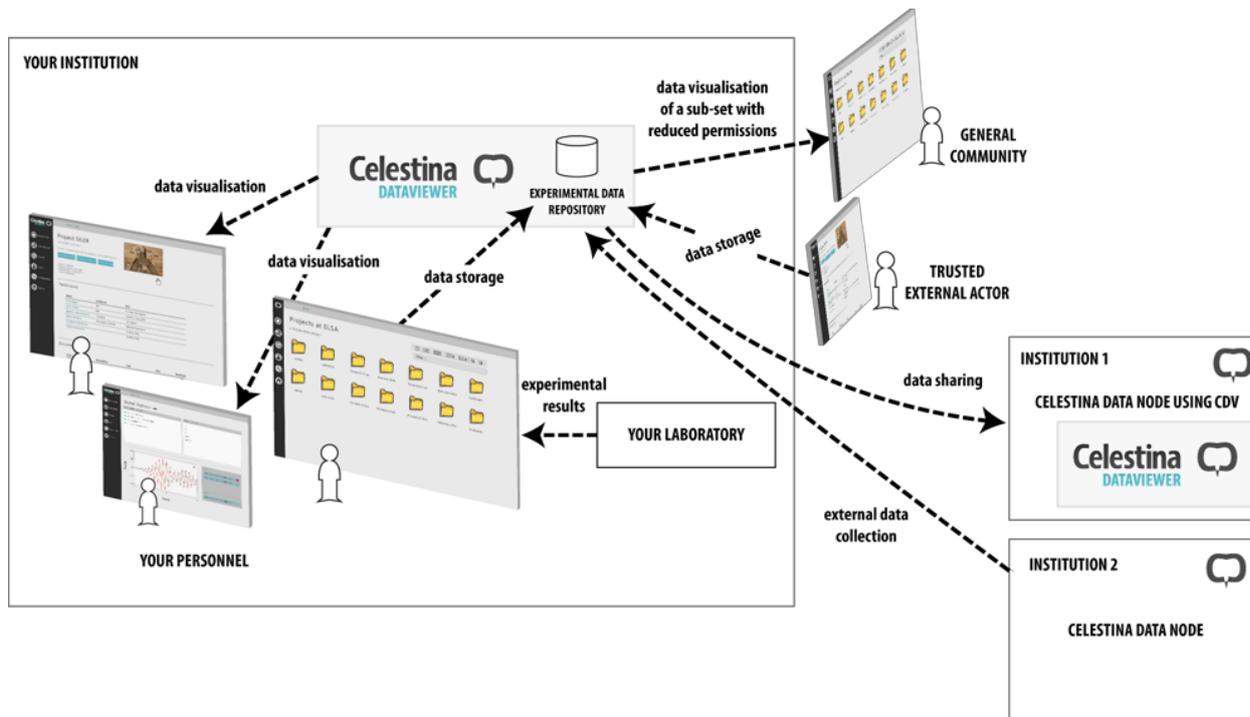


Figure 3. Operate as a Celestina node interacting with other nodes

You can use CDV even if you do not have a laboratory or do not store any experimental data at all. CDV can be used just as a viewer to visualise experimental data publicly available in the Celestina Data network. In this case, as depicted in Figure 4, a repository is not necessary (from a functional point -technically, one can be used for performance reasons) and the CDV will be in charge of loading data from external Celestina Data nodes and present them to you. The installation process of this configuration is exactly the same as the first case.

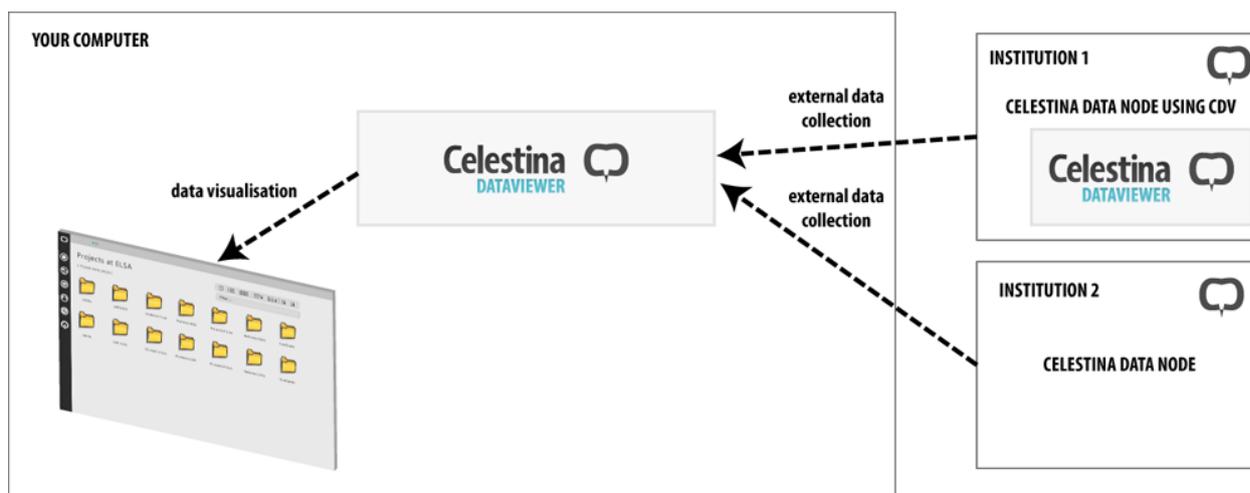


Figure 4. Using CDV as a simple viewer, without storing data

As discussed, CDV can be configured and used in a variety of ways. This overview is given to understand the potential and some of the features of CDV. The installation process for all configurations is very similar, with two exceptions: (i) If external access is needed, your organisation

network policies have to be modified accordingly. (ii) If CDV needs to communicate with other CDV systems (within the Celestina network), additional configuration steps have to be taken. However, these are not discussed in this document since the features are not available at this time.

Additionally, an institution could run two or more instance of CDV. For example, an instance of CDV can be made publicly available for the general community, containing only public data, while another instance of CDV can be used internally, containing private data for your organisation. This is not really a necessary step, since a single instance of CDV can manage both public and private data internally, without the need to use several instances. However, it might be the only solution for organisations that cannot create a common network infrastructure to serve both internal and external users.

2.1 Document organisation

The rest of the document is organised as follows:

Section 2 discusses some details you need to know before starting.

Section 3 explains the process to install the virtual machine in your computer.

Section 4 describes the configuration process of CDV. It uses a tool that has been created to simplify this process for you.

Section 5 explains how to check the application is up and running, and can be accessed.

Section 6 discusses some administration details for the CDV virtual machine.

Section 7 points to the contact information for further support.

3 Pre-requisites

3.1 Required files

Some files are required to install the application. You can download them from:

```
http://celestina-integrations.com/cdv\_releases
```

The link will get you to a list of CDV versions named by release date (e.g. CDV_18-OCT). Choose the most recent version. Normally, you will find at least three directories inside each version:

- **CDV VM:** Contains the files of the preinstalled machine (see Sections 3.3 and 4)
- **cdvconf:** Contains a tool to support an easy configuration and management of the machine (See Section 5). Normally, different versions can be found (e.g. Windows, Linux, the original Python script,...).
- **Actions:** Contains the tasks that can be conducted by the *cdvconf* tool (See Section 5.2). They allow you to do administrative tasks inside the CDV computer in a simple way. There are different actions, from the ones that configure the system for the first time to the ones that power off the machine or collect log files from the application. This directory also contains an example of the file *cdv.conf*, which you should edit and copy in the same directory as *cdvconf*.

3.2 Computer requirements

You need a computer in which to install CDV. This computer should have some minimum specifications. In general, CDV will take advantage of better processors, more cores, more and faster RAM memory, and faster hard drives. It is recommended you use a modern computer, because CDV will run inside a virtual machine, which will already have a noticeable performance penalty, and your computer should be able to cope with such load. It is recommended you use a modern i9/i7 Intel processor (or similar server-oriented options), with at least 12 cores, 16GB RAM and SSD hard drives. Of course CDV will run on more modest hardware, but the best specifications your computer have, the best it will behave. In our experience, running a virtual machine in a 4 years-old i7 processor with 4 cores and 8GB RAM was enough, but the user experience was occasionally affected negatively with some long delays. You can easily do the test yourself on your hardware and decide whether to give more power to the machine or not. If your users experience bad performance, please report it to us. There are some ways to address this issue:

- Improving the hardware specifications of your machine. Assigning more processor power, especially, will have a direct impact in the application performance. This is something you can do yourself.

- Improving overall performance of the application and the data model. This is something that we are investigating and considering for future versions.
- Installing the application directly in the machine, instead of running it from a virtual machine. This might also have a substantial impact, especially in some environments where virtual machines do not run efficiently. Contact us to obtain more information if you need it.

Also, consider that CDV runs partially in the server (your CDV computer) and partially in the client (the user's computer), meaning that some performance issues can be directly attributable to the user's computer.

3.3 CDV VM (Celestina Data Viewer Virtual Machine) files

The first thing needed are the files containing the CDV VM: the virtual machine that contains all required software preinstalled. They are a few files of a total of 1 to 30 GB of size (depending on the version that is delivered). You can think of this virtual machine as a full isolated computer, with its own operating system and software. The CDV VM is running Ubuntu Server¹. These files can be obtained with the instructions from Section 3.1 . The files alone cannot run by themselves, and additional software (called a hypervisor) is required. The next section discusses about this hypervisor.

3.4 Hypervisor

In order to run the CDV VM, you need to install a hypervisor. A hypervisor is a piece of software that runs virtual machines. There are many ways to run virtual machines, but this document will focus on a program that you install in your computer, in the usual way, and that will run the virtual machine. Two of these programs will be recommended in this document:

- VMware^{2 3}, which is a commercial product that has a free version for non-commercial use (called *Workstation Player*). It is the recommended option.
- VirtualBox⁴, which is a free hypervisor.

Both programs work in a very similar way, but VMware has been recommended based on personal experience. You can install the hypervisor in your computer (called the "host" computer) and run on it the CDV virtual machine (called the "guest" computer), so both systems will share the computer hardware. Of course, the VM will have a performance penalty since the VM will run in an emulated environment. To take full advantage of the hardware of your computer, you could install all software directly on the computer, but then you will have to do yourself all the installation and configuration manually. The advantage of using a VM is to simplify the computer management in detriment of performance.

¹ <https://www.ubuntu.com/server>

² <https://www.vmware.com/products/workstation-pro.html>

³ <https://www.vmware.com/products/workstation-player.html>

⁴ <https://www.virtualbox.org/>

Once you have decided which hypervisor you are going to use, you just need to go to the Web and download the software, and then install it in the usual way for your operating system. This process is quite standard and easy.

3.5 Network

CDV is an application that will be accessed through the network, so you need to know the network environment in which it will be installed. There are many possible configurations, but one of the standard ways is to assign a static IP to the CDV VM (because it is like another isolated computer in your network). So the computer you are using to install the virtual machine will have an IP address and your virtual machine will have a different IP address (you should think of the virtual machine as a totally independent computer, with the exception that it is sharing hardware with a host computer). You might need help from your IT department to identify which IP address you can use or how this address can be assigned to the virtual machine (DHCP, statically⁵, etc). If you want the machine to be accessible not only from your internal network but from the outside world this is something that you need to consider and explain to your IT department.

Likewise, you need to know the firewall rules that apply to your machine, since network devices might prevent external clients to connect to the CDV VM. In basic terms, you need access to the TCP ports **443** and **444** for application use, and the TCP port **22** for administration, so your department firewalls in the network path should permit the access to these ports. If only internal use is required, then these ports can be closed for external users (in any case, port 22 should be closed for external use, unless you really need to administrate the computer remotely).

Finally, it is **strongly suggested** that you assign a domain name to the CDV VM, so that you can address it as e.g. "elsadata.ec.europa.eu" instead of by using its IP address (e.g. "131.168.33.4"). This has many advantages, not only in terms of independence of IP address but also when configuring the security in the application. To assign a domain name, it is better if you ask for one to your IT department. This will be referred in this document as the computer hostname (sometimes mentioned as `YOUR_HOSTNAME`).

Additionally, you will have to choose a short name, with Latin characters, for your Celestina Data Viewer instance. For convention, it is always written in uppercase. This will be your "system name". For instance, at the ELSA lab the CDV instance is called "ELSADATA". The University of Patras could decide to call their instance "UPATDATA", "PATRASDATA", "UPATCDV", "CELESTINAUPAT", "UPATDB" or simply "UPAT". This name is decided only by you, but should be relatively short and unique. You can use this system name to create the domain name of the computer, e.g.: elsadata.ec.europa.eu, upatdata.upatras.gr, celestinaoxf.ox.ac.uk, etc.

3.6 Security and certificates

⁵ More information at <https://michael.mckinnon.id.au/2016/05/05/configuring-ubuntu-16-04-static-ip-address/> and <https://help.ubuntu.com/lts/serverguide/network-configuration.html.en> and

CDV uses a security model based on X.509 public key certificates⁶. The subject is not trivial to explain but it will be oversimplified in this document, for the sake of clarity.

In summary, the communication with CDV and the authentication of users is secured by the use of public key cryptography. To do so, CDV, which is on the server side, has to provide a certificate to its users. There are two main ways to obtain this server certificate:

1. Create a certificate that is signed by an external entity, which is "trusted" by default by most of the Web browsers. This usually means to "buy a certificate" from a provider, which will last for some months or years and has to be renewed after that period of time. The price to issue a certificate varies from provider to provider (from being free to a few hundred of euros).
2. Create and sign your own certificate (called a "self-signed certificate" or a certificate signed and trusted only by your own organisation). This option is free and is automatically managed by CDV for you.

Option 1 is the "correct" way to do it. If you acquire a certificate from a trusted entity, all your users will trust automatically your CDV application, without any other action. However, there is normally an economic cost and some management involved to renew the certificate every once and a while. If you are exposing your CDV application to the general public, it is recommended you use this option.

Option 2 is a perfectly valid alternative if you only use CDV internally in your organisation. If you control the environment of your users, you can ask them to trust your existing organisation certificate, or your CDV-created certificate (there are still some minor risks, but it is a valid option). The problem if you use your own certificate without obtaining one from a trusted organisation, is that your users will get the message depicted in Figure 5 when they access your CDV instance. It is quite tricky to guarantee the security of the communication, and it is especially tricky if you use a certificate that is not trusted by default.

Not implementing this security correctly implies the possibility for someone to stand in the middle of the communication between your CDV instance and your users. This person could then eavesdrop the data in transit, change it, impersonate your application, etc. One may argue that this is unlikely to happen to your organisation, but you should be aware of the potential risks.

There exists the possibility to disable the secure communication for your public users, so they can connect without any security but without getting security warnings coming from a certificate that is not trusted. If you wish to do so and accept the risks, please contact us to provide you with instructions.

Further information about the security of CDV can be found in the user manual of CDV, which is contained inside the application (once you install and run the application you will be able to access it).

⁶ <https://tools.ietf.org/html/rfc4158>

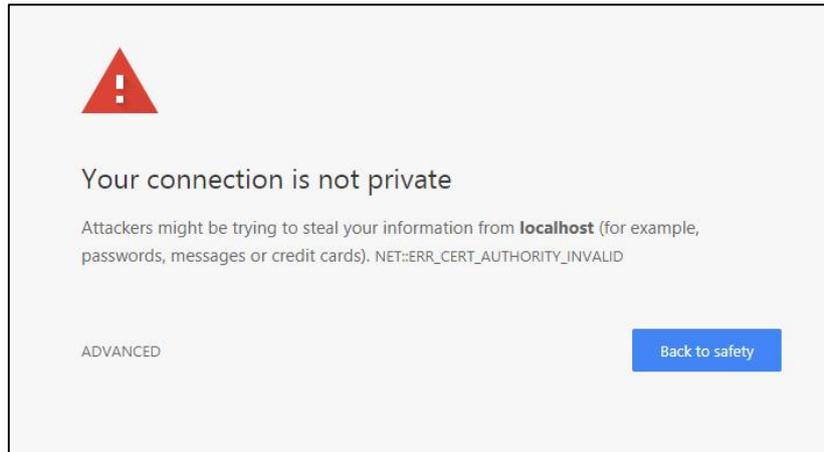


Figure 5. A message served by a browser when a server certificate is not trusted

3.7 Database licence

CDV uses a piece of software that cannot be distributed inside the virtual machine. You will have to download it yourself, although the installation will be done for you as explained later in this document.

The software can be obtained at:

<https://www.stardog.com/#download>

This is a commercial product that you are encouraged to purchase if you find it useful. There exists, however, a free version (Community) for non-commercial use that has some limitations but will be appropriate for most of the CDV users. You will use the downloaded files as explained in Section 5.4 .

4 Virtual Machine Installation and Configuration

NOTE: The virtual machine provided is a computer itself. As such, it suffers from all security problems that computers have. Although it has been configured and has been slightly hardened, note that ultimately you are responsible for ensuring the security of this machine. We do not hold any liability about the machine, and you should use it under your own risk.

4.1 Virtual Machine Importation

Once the hypervisor has been installed and you have downloaded the files for the CDV VM, it is time to import them. The process is very simple, and will be explained for VMware (VirtualBox instructions are very similar):

- Run VMware
- Click File - Open
- Select the .ovf file of the CDV VM
- Accept and wait for the importation process to complete

The machine should be ready for execution as shown in Figure 6.

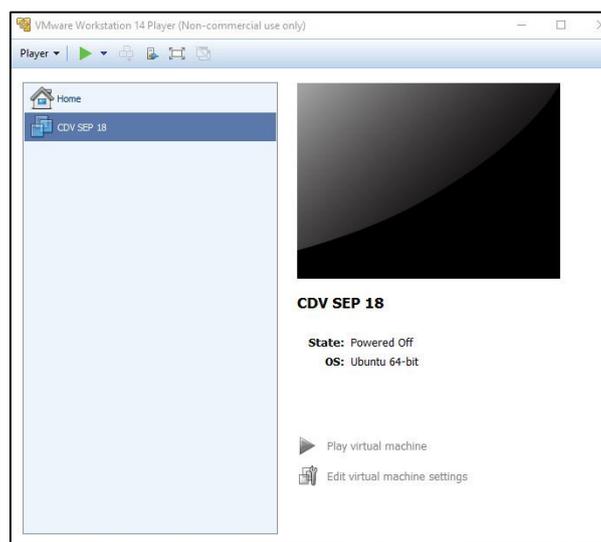


Figure 6. CDV VM imported and ready for execution

4.2 Virtual Machine Hardware Configuration

By clicking on the "Edit virtual machine settings" option shown in Figure 6, you will access the menu to configure the hardware assigned to the CDV VM (see Figure 7). It is important you provide the machine with as much resources as you can without severely affecting the performance of the host machine⁷:

- Memory. Give the machine at least 8GB, although more is desirable.
- Processors. Give the machine as many as you can without affecting host performance.
- Hard disk. The hard disk is configured to have a dynamically assigned space. This will affect performance but allows the allocation of space as it is needed. This means that only a few GB will be used initially in your hard disk, growing as needed up to what has been configured (in this case, 500GB). The amount of GB that has been initially reserved for the system can be extended if necessary.
- Network adapter. The configuration here will depend on your network configuration. If you want to assign a static IP to the machine, you can use the "Bridge" method⁸, which places the virtual machine as if it was plugged directly on the network. Please discuss this with your IT department, always considering your network environment.

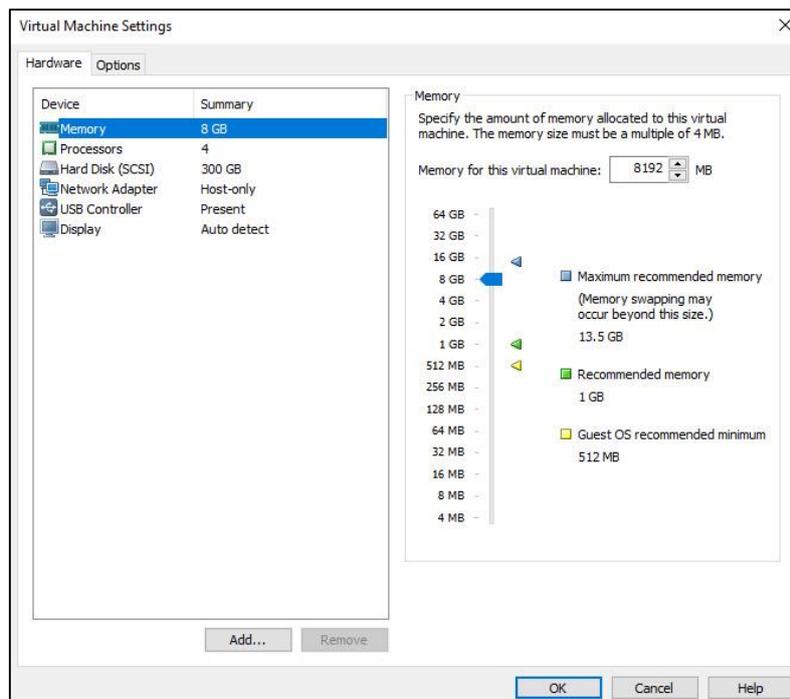


Figure 7. Hardware configuration of virtual machine

4.3 Running the Virtual Machine

Once the virtual machine has been imported and the hardware has been configured, the machine can be run by using the play button (). Note that in this emulated environment, the virtual

⁷ A benefit of virtual machines is that you can easily change its hardware at a later time, and assign more processors and RAM memory if performance is bad

⁸ https://www.vmware.com/support/ws4/doc/network_configure_ws.html

machine can be paused and resumed at any given time. If you start the virtual machine, after the booting process you should see the login screen shown in Figure 8. This is the login screen for Ubuntu Server 18.04. If you wish, you could attempt a login in the machine with the username *cdv* and the same password (you will see it is a normal computer running inside your computer), although it is not recommended to login just now.

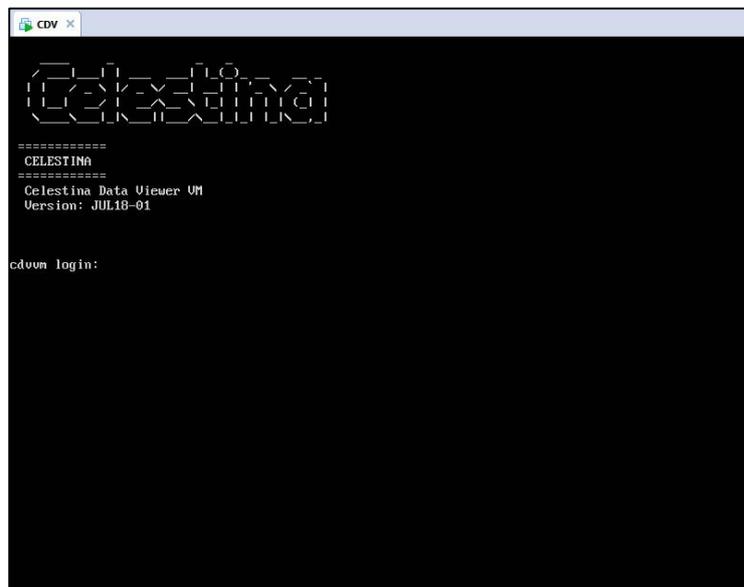


Figure 8. Login screen of CDV VM


```
CDVVM.SSH.PORT=22

CDVVM.USER.USERNAME=cdv

CDVVM.USER.PASSWORD=cdv

CDVVM.USER.NEWPASSWORD=8udo2_7d-dkdn2ydkarkshcupz

SITE.NAME=ELSADATA

SITE.DESCRPTION=Experimental database for the ELSA laboratory at the
JRC (Ispra, Italy)

BASE_DOWNLOAD.IRI=https://elsadata.ec.europa.eu:444/files

BASE_DOWNLOAD.DIR= /home/cdv/cdvapp/datafiles/files/

SITE.DATA.IRI=http://elsadata.ec.europa.eu/celestina#

SITE.DNSNAMES=elsadata.ec.europa.eu,localhost

DATABASE.NAME=elsadata

DATABASE.ADMIN.USERNAME=admin

DATABASE.ADMIN.PASSWORD=S3OrnKHdlZ4S7b42gc

DATABASE.ADMIN.DEFAULTPASSWORD=admin

DATABASE.USER.USERNAME=cdvuser

DATABASE.USER.PASSWORD=9DlXQL5DHw6b5g5w1Hu

DATABASE.HOSTNAME=localhost

DATABASE.PORT=5820

DATABASE.CONNECTION_STRING=http://localhost:5820

DATABASE.MEM=3

DATA.IMPORTFILE=ELSA_DB.ttl

KEYSTORE.FILE=/home/cdv/cdvapp/sec/local.jks

KEYSTORE.PASSWORD=3Xo5e6PbLvV1NeXz6091t

KEYSTORE.TYPE=JKS

TRUSTSTORE.FILE=/home/cdv/cdvapp/sec/local.jks

TRUSTSTORE.PASSWORD=3Xo5e6PbLvV1NeXz6091t

TRUSTSTORE.TYPE=JKS

TRUSTSTORE.LOCALCAALIAS=local

WAS.XMS=2048m

WAS.XMX=2048m
```

```
WAS . PERMSIZE=1024m
WAS . MAXPERMSIZE=2048m
```

The description of these variables is given below (please read them carefully, since a mistake configuring this might mean you will have to repeat the installation process):

- #: This represents a comment. Any line starting with this character will be ignored.
- CDVVM.ADDR: This is the hostname or IP address of the virtual machine (CDV VM). You will need to modify this value to set your own hostname or IP address (e.g. elsadata.ec.europa.eu).
- CDVVM.SSH.PORT: This is the port for SSH access. You should not modify this unless you have changed manually the SSH server port.
- CDVVM.USER.USERNAME: This is the username of the account in the Linux system of the virtual machine. You should not modify this.
- CDVVM.USER.PASSWORD: This is the password of the account in the Linux system of the virtual machine. You should use the default value the first time, but then you should change it to a new value, as specified in the variable CDVVM.USER.NEWPASSWORD.
- CDVVM.USER.NEWPASSWORD: This variable indicates that you want to change the current password (CDVVM.USER.PASSWORD) in the virtual machine for the new password specified here. After execution, remember to comment this variable with a "#" and to set the new password in the variable CDVVM.USER.PASSWORD for future use of the tool.
- SITE.NAME: This is your "system name" discussed in Section 2.4. It should be short and unique, and you should use standard Latin characters for it. For example, at the ELSA lab the SITE.NAME is "ELSADATA". The University of Patras could decide to use "UPATDATA", "PATRASDATA", "UPATCDV", "CELESTINAUPAT", "UPATDB" or simply "UPAT".
- SITE.DESCRPTION: A brief description about your system. For example, you can mention where it is geographically located, the source of the data and which institution is in charge of it.
- BASE_DOWNLOAD.IRI: This is the URL that will be used for download large files. By default, you should normally use **https://YOUR_HOSTNAME:444/files/**
- BASE_DOWNLOAD.DIR: This is the directory where large files are stored. You should not modify this value.
- SITE.DATA.IRI: This is your datasource ID. You should use **http://YOUR_HOSTNAME/celestina#**
- SITE.DNSNAMES: This is a comma-separated list of the domain names you need to include in the creation of your certificates. Normally, you will just need to include the computer hostname (YOUR_HOSTNAME) but you can add any other domain name you think you will need to address the computer. **NOTE:** Some Web browsers will require you to access your CDV application via one of these hostnames in order to use your certificates. For example, if you configure here the host "elsadata.jrc.it" and then you access CDV by using e.g. "https://elsadata.jrc.eu.europa.eu" or "https://192.168.1.100", then your Web browser might not try to authenticate your specific user, ignoring your credentials, and you will always access the application as the *PUBLIC* user.

- DATABASE.NAME: This is a formal name of the database. You should use your system name in lowercase (providing it has been selected correctly, with no spaces and other characters).
- DATABASE.ADMIN.USERNAME: The username of the database administrator. You should not modify this value.
- DATABASE.ADMIN.PASSWORD: The password for the database administrator. You HAVE to change this value and choose a "good" password. The password should be between 4 and 20 characters and can contain alphanumeric characters and the symbols @\$%!&. Other symbols such as []()<>= are not allowed, so make sure you do not choose them or the configuration process will fail.
- DATABASE.ADMIN.DEFAULTPASSWORD: The default password for the administrator. You should not modify this value.
- DATABASE.USER.USERNAME: A username for the database. You do not need to modify this.
- DATABASE.USER.PASSWORD: The password for the database administrator. You HAVE to change this value and choose a "good" password, DIFFERENT from any other password. The password should be between 4 and 20 characters and can contain alphanumeric characters and the symbols @\$%!&. Other symbols such as []()<>= are not allowed, so make sure you do not choose them or the configuration process will fail.
- DATABASE.HOSTNAME: The hostname of the database. You do not need to modify this value.
- DATABASE.PORT: The access port of the database. You do not need to modify this value.
- DATABASE.CONNECTION_STRING: The string to connect to the database. You do not need to modify this value. Normally it will be made from `http://DATABASE.HOSTNAME:DATABASE.PORT`
- DATABASE.MEM: Number of GB of RAM you want to allocate for the database. This value will be used for both the heap and off-heap memory assignment of the Java Virtual Machine (so using 3GB means using 3GB for heap memory and 3GB for off-heap memory).
- DATA.IMPORTFILE: This is only necessary if you have data from a previous database and need to import the data. Leave blank or comment the line out with "#" otherwise.
- KEYSTORE.FILE: File of the keystore. You normally will not need to modify this value.
- KEYSTORE.PASSWORD: Password for the keystore. You HAVE to change this value and choose a "good" password, DIFFERENT from any other password.
- KEYSTORE.TYPE: Type of the keystore. You normally will not need to modify this value.
- TRUSTSTORE.FILE: File of the truststore. You should not modify this value.
- TRUSTSTORE.PASSWORD: Password for the truststore. You HAVE to change this value and choose a "good" password, DIFFERENT from any other password (**note that** if KEYSTORE.FILE and TRUSTSTORE.FILE are equal, then KEYSTORE.PASSWORD and TRUSTSTORE.PASSWORD **must be equal too**, because they are pointing to the same file. Because by default you will not modify KEYSTORE.FILE and TRUSTSTORE.FILE then **make sure** KEYSTORE.PASSWORD and TRUSTSTORE.PASSWORD are the same).
- TRUSTSTORE.TYPE: Type of the truststore. You should not modify this value.
- TRUSTSTORE.LOCALCAALIAS: Alias for the local Certificate Authority. You should not modify this value.

- `WAS.XMS`, `WAS.XMX`, `WAS.PERMSIZE`, `WAS.MAXPERMSIZE`: Amount of RAM you want to allocate for the Web Application Server. Change the values 2048m (2GB) if you need to allocate more memory.

The `cdv.conf` file is not a "closed" file. New variables might be added in future, as they are needed.

5.2 Actions

An action is simply a ZIP file that contains scripts to automate some task(s). Normally, you will be provided with an Action to conduct a specific task, so you do not have to create any Actions yourself. Cdvconf has been conscientiously made to facilitate tasks for users and eliminate any potential burden, and Actions play a key role on this.

This does not mean, of course, that you cannot create your own Actions to automate some tasks. Advanced users are suggested to open an existing Action ZIP file and take a look at its format. You can easily open any Action ZIP file to see what the Action will do when executed.

NOTE: *The rest of the section will give specific details about Action files. You can move to the next section if you do not plan to develop any Action and just want to use Actions as a normal user.*

As stated before, Actions are just self-contained files, compressed as a ZIP, containing scripts and other files to conduct some administrative tasks over a remote computer. The scripts allowed are either Bash Shell scripts or Python scripts. The `cdvconf` tool is just the tool that will interpret and run the scripts, in a predefined order based on convention over configuration (so the expected names of the scripts are predefined and cannot be changed). The execution order is given by the structure of the ZIP file, as follows:

- `pre.sh` -> If exists, this is the first script that is run
- `pre.py` -> If exists, this is the second script that is run
- `X-NNNN` -> A directory, where X is a number and NNNN is any arbitrary text. Example: 01-database, 3-Data loading, etc. The tool will go through all the directories, in the order set by X (starting from 0), and will attempt to execute the scripts `install.sh` (if exists) and then `install.py` (if exists). If a file `output.txt` exists, the tool will attempt to download the files specified (more about this below).
- `pos.sh` -> If exists, this will be executed after all directories have been treated
- `pos.py` -> If exists, this is the latest script that will be executed
- `output.txt` -> If this file exists, the tool will attempt to download the files specified (more about this below)

Scripts and files might need to use the values declared by users in the `cdv.conf` file. To do so, `cdvconf` will preprocess the files, substituting any string defined in a special format, with the value found in

the *cdv.conf* file. Please note that only non-binary files will be preprocessed. The special format used in the files to declare a variable from *cdv.conf* is:

```
[!!NAME_OF_VARIABLE!!]
```

This means that if a script (e.g. *pre.sh*) uses the following sentence:

```
cp [!!DATA.IMPORTFILE!!] /home/cdv/tmp/
```

Being *DATA.IMPORTFILE* defined in *cdv.conf* as:

```
DATA.IMPORTFILE=ELSA_DB.ttl
```

The preprocessing will substitute the value, leaving the sentence as:

```
cp ELSA_DB.ttl /home/cdv/tmp/
```

This preprocessing is the main way to bind values in the *cdv.conf* file with scripts and files inside an Action. This preprocessing occurs in the main executable, *cdvconf*.

The format of the *output.txt* files is just a list of paths, with files that have to be retrieved from the server (the virtual machine). The following *output.txt* example will instruct *cdvconf* to collect two files:

```
/home/cdv/cdvapp/myfile.txt  
/home/cdv/results.ttl
```

5.3 Using *cdvconf*

Using *cdvconf* is very simple:

- Make sure you have edited first the *cdv.conf* file setting the right values. Copy this file in the same directory as the *cdvconf* executable.
- Make sure you have an action ZIP file you want to apply.
- Run *cdvconf* (the tool downloaded as explained in Section 3.1) by using one of the executables provided (the one with ".exe" extension for Windows, the one without extension for Linux or the one with ".py" extension for Python). You will be asked to select your *cdv.conf* file if it is not found in the same directory as the tool, and a ZIP file as depicted in Figure 9. Choose the Action you wish to use.

- Wait for execution to complete. In some cases, you might be asked to provide the password of the user of the virtual machine (*cdv*). Some Actions need to wait for some seconds to finish, so it is normal if occasionally you do not get feedback in a couple of minutes.
- That is all. You might get some files as result of the execution of the action. This can be found in the *cdvconf* directory under a new directory called "*output-yyyyMMdd*".

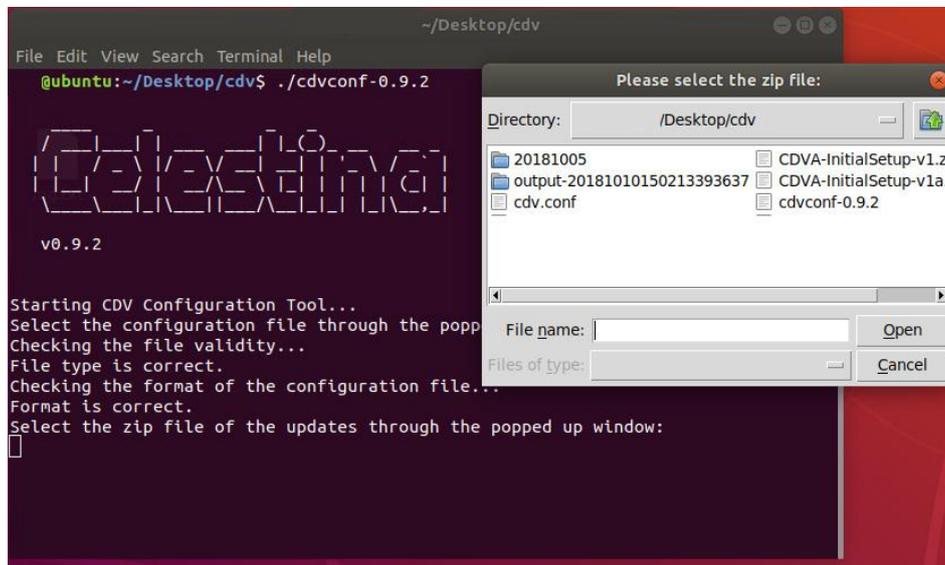


Figure 9. cdvconf tool screen

NOTE: The rest of the sub-section will give specific details about how *cdvconf* operates. You can freely skip the rest of this section.

In general terms, the sequence of operation of the *cdvconf* tool is:

- Establish a connection via SSH to the virtual machine. To do so, it uses the values *CDVVM.ADDR* and *CDVVM.SSH.PORT* to connect, and then the values *CDVVM.USER.USERNAME* and *CDVVM.USER.PASSWORD* for authentication. If the latter values are not found in *cdv.conf*, the tool will ask for them interactively.
- Ask the user for an Action file to be run.
- Preprocess values from *cdv.conf* in any non-binary file inside the Action (this includes scripts but also any other type of files, such as XML files).
- Copy the Action files inside the server (the virtual machine).
- Execute the files in the order established. If necessary, the tool will download the files specified in any *output.txt* file to the user's computer.

5.4 First-time configuration of CDV

To conduct the first-time configuration of CDV, you will need the Action called:

CDVA-InitialSetup-v1.ZIP

That you can download from the link of Section 3.1 . You should have two files: a copy of Stardog and a licence. Copy both files inside the ZIP file directory:

```
03-stardog/
```

So the directory will contain the files *install.sh*, *stardog-license-key.bin* and *stardog-X.X.X.zip* (X.X.X is the version). **NOTE:** The files have to be copied **INSIDE** the ZIP file.

Make sure you have edited the *cdv.conf* file and run *cdvconf*. Select the Action and let it work (some of the tasks will take for a bit longer than a minute without giving you any feedback). Once finished, you will have a new directory called "output-" followed by the time you executed the tool. Inside you should see a file "iniadmin.p12" that you will need later on to configure users (as explained below in Section 7.3). After this first-time configuration, the next steps are:

- Verify that CDV is running correctly. Section 6 explains how to do it.
- Once the application is running, read the user manual (how to access this manual is also explained in Section 6).
- Create your CDV users, as explained in Section 7.3 and the user manual.

5.5 About security

As you can see, *cdvconf* is a powerful tool to automate administrative tasks for you. However, this convenience comes also at a price. You must be aware that the tool can be used to break into your CDV system easily, not because of the tool itself, but because the *cdv.conf* file contains most (if not all) the passwords of the software used in the virtual machine. Therefore, you might take special care in protecting this file from unauthorised access. Also, passwords and other credentials will be moved into the computer, so if something fails in the middle they might remain there unencrypted.

Likewise, Action files can contain potentially undesired actions to be run in your system. Make sure you only run Actions from people you trust. Alternatively, you can open the ZIP file yourself to check what the Action is really doing.

assigned so you do not have to use an IP address. When trying to access the application, different results can occur, as explained in the following sections.

6.1 Network problems

Figure 11 depicts the usual error when the computer cannot be reached (the connection was reset). This is normally a connectivity problem, and for some reason your computer cannot communicate with the virtual machine. It is possible that the hostname you are using is not correct (e.g. you used "elsadta" instead of "elsadata"), that the hostname is not pointing to the right IP address of the machine, that there is a firewall preventing the communication, etc. You might require the help of your IT department.

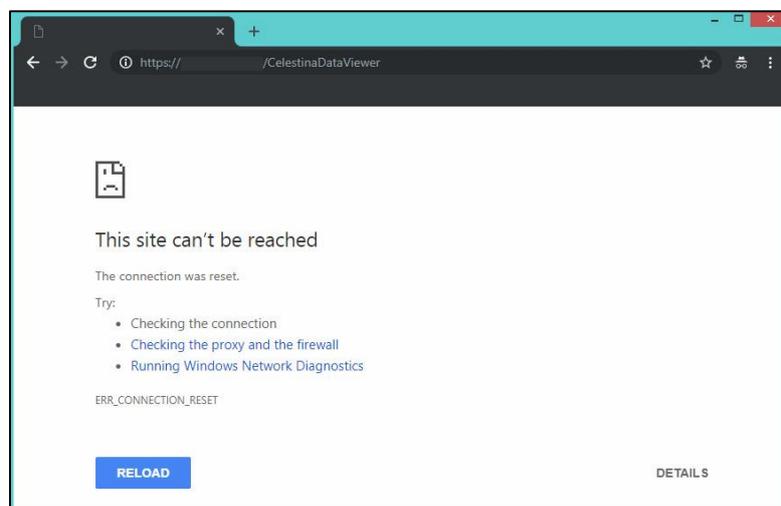


Figure 11. Network error - the site cannot be reached

If you feel like doing some tests yourself, you can attempt to use the *ping* command to the hostname and see if you can reach it from the host machine (the computer where you installed VMware):

```
ping virtual_machine_hostname
```

Getting a positive answer might mean that you have access from the host machine, but perhaps not from another machine (is there a firewall in the middle? is there not network connectivity?). If you get no positive, you can try the same command with the IP instead:

```
ping virtual_machine_IP_address
```

If this time you get a positive answer, it is likely the hostname is not pointing to the right IP address or has not been configured or populated yet. If no positive answer is received either it probably

means the IP is not correct, or the virtual machine has no IP address configured (since, by default, ICMP echo packets are not filtered in CDV VM). A static IP address can be configured easily by logging into the virtual machine and set it yourself manually¹⁰.

6.2 Application not running

If you receive the answer of Figure 12, the chances are that the application is not running. The key message is that a hostname "refused to connect", so it seems there is network connectivity but nothing was serving the request you have made.

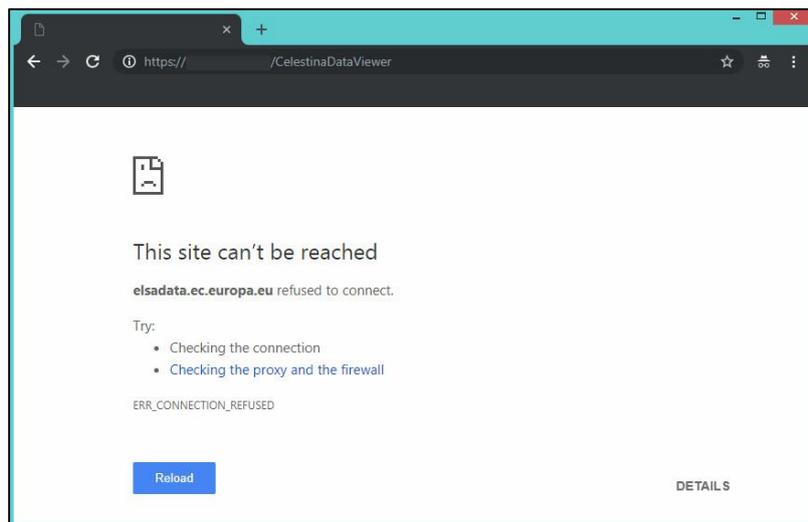


Figure 12. Error - Application not running

This could also mean that you use the wrong URL to access the application. So, for example, if you use "http://..." instead of "https://..." you will likely get this error too.

To run the application, you can follow the process explained above: log in into the CDV VM. This will trigger the command "cdv-start-all.sh" which will ask you one more time for your password. Afterwards, the application will start to load, wait until the process finish (see Figure 13).

¹⁰ https://www.swiftstack.com/docs/install/configure_networking.html

```
=====  
CELESTINA  
=====  
Celestina Data Viewer UM  
Version: JUL18-01  
  
cdvum login: cdv  
Password:  
Last login: Tue Sep 25 09:32:03 PDT 2018 on tty1  
Welcome to Ubuntu 16.04.5 LTS (GNU/Linux 4.4.0-127-generic x86_64)  
  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:       https://ubuntu.com/advantage  
=====  
Starting CELESTINA DATA VIEWER  
=====  
* * * Exporting version information...  
* * * Starting Nginx...  
[sudo] password for cdv:  
* * * Starting Tomee...  
Using CATALINA_BASE:   /home/cdv/cdvapp/tomee/apache-tomee-7.0.0-plume  
Using CATALINA_HOME:   /home/cdv/cdvapp/tomee/apache-tomee-7.0.0-plume  
Using CATALINA_TMPDIR: /home/cdv/cdvapp/tomee/apache-tomee-7.0.0-plume/temp  
Using JRE_HOME:        /usr  
Using CLASSPATH:       /home/cdv/cdvapp/tomee/apache-tomee-7.0.0-plume/bin/bootstrap.jar:/home/cdv/cdvapp/tomee/apache-tomee-7.0.0-plume/bin/tomcat-juli.jar  
Tomcat started.
```

Figure 13. Application starting after login process

6.3 Application is running OK

When the application has started from the first time and your certificate configuration is correct, you should see something similar to Figure 14. Note that the figure is showing some data, and your repository will likely be empty.

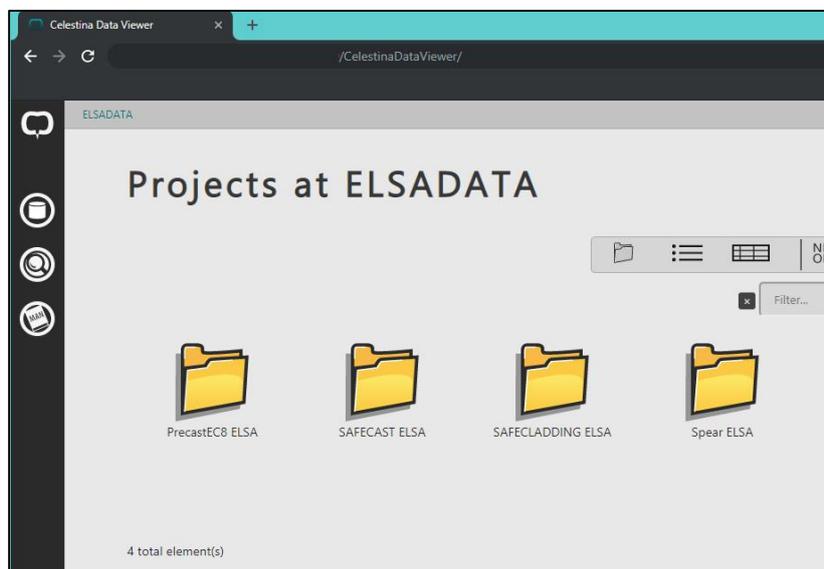


Figure 14. Application started OK

If your certificate configuration is not quite right, before you see the image of Figure 14 you might get a page like the one depicted in Figure 15. Do not worry at this time about it, just click in "Advance" and "Proceed" (different Web browsers might show different options to do so). You should then see something similar to Figure 14.

In order to solve the issue with the certificate, please refer to a previous section of this manual and also check the user manual that can be found inside the application at:

<https://HOSTNAME/CelestinaDataViewer/manual.xhtml>

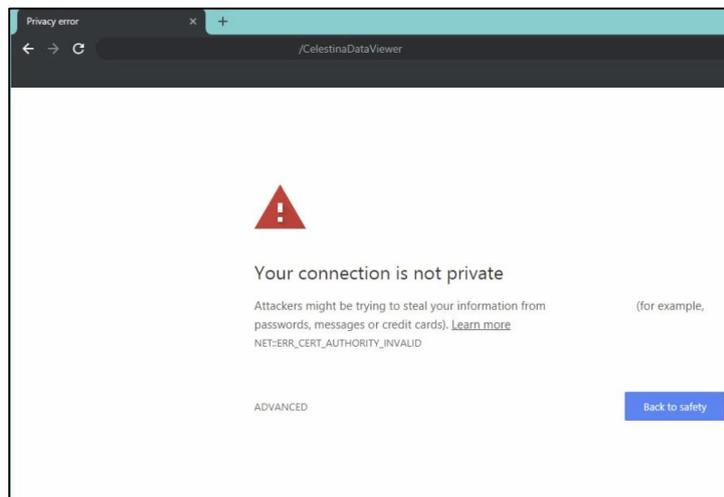


Figure 15. Certificate error

6.4 Application running, with some errors

If you see something similar to Figure 16, the application is running or partially running, but there were some errors at some point. There are many possible reasons for this, and sometimes it will be necessary to check the logs that the application is writing.

For example, if the database has not been started for any reason, this error will appear. It is possible that the database is "starting" but the process depicted in Figure 13 has not finished yet and the database has not completely started.

In general terms, this error page will happen during the life of the application, and usually will imply the retrieval of log files to investigate what the source of the problem was.

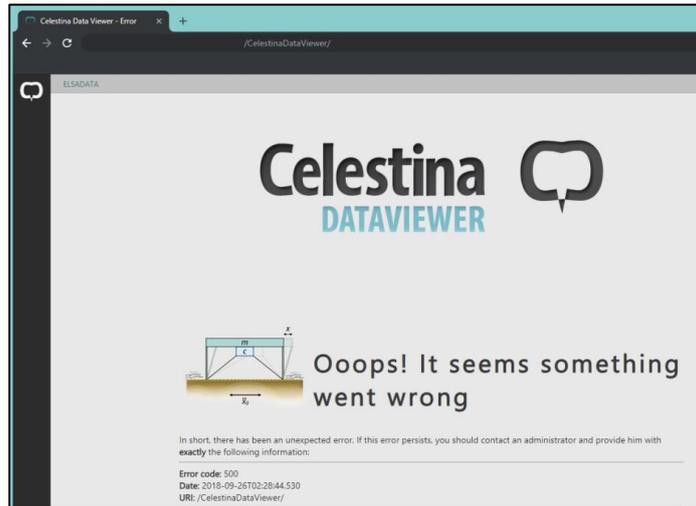


Figure 16. Application error

7 Administration

NOTE: Some administration tasks are discussed in this section. However, please consider you are the only responsible for the administration and maintenance of both the application and the virtual machine provided. You should use them under your own risk. Please note that NO SUPPORT is officially provided.

7.1 Administration methods

The CDV VM can be administrated in these three ways:

- By using the **cdvconf** tool. This is the preferred way, since the tool has been created to simplify all the process for you. The basis of the tool is that you only provide some variable values in a text file, and by using a ZIP file that is provided to you, the system is managed for you.
- By using a **SSH client** on TCP port 22 of the virtual machine. This is the next best way, but it requires you to have some knowledge of Linux. You can use SSH from your computer to the CDV virtual machine, logging in remotely into the system where you can run commands and upload and download files. SSH clients are available for Linux (normally preinstalled) and Windows (Bitvise¹¹ is recommended). To use *ssh* you normally need only the name or address of the virtual machine, the port (22 by default) and the username and password of a user of the system (*cdv* by default).
- By having **physical access** to the virtual machine. You can use the virtual machine directly from the host computer, as is shown in Figure 8 and Figure 13. Sometimes this option is not possible, since the machine might be stored in a place where you have no easy access.

7.2 Directory structure

*NOTE: This information is given as a reference if you use SSH to connect to the virtual machine. The **cdvconf** method does not require knowledge of this.*

Inside the virtual machine, you will normally access as the *cdv* user, which has *sudo* permissions. This is a summary of some directories that can be found in the home directory:

```
/cdv
  /scripts -> Contains useful scripts for management
```

¹¹ <https://www.bitvise.com/>

```
/cdvapp -> Contains most of the data needed for CDV

/applications -> Contains WAR files with the apps

/data -> Contains the database data

/datafiles -> Contains the large files

/logs -> Points to the logs for the database, WAS and WS

/sec -> Contains certificates and keystores for security
```

7.3 Creating a CDV administrator

Once CDV has been configured and we have checked it is running and can be accessed, the first thing to do is to obtain administration privileges for a first CDV user configuration. This refers to the users inside CDV, as an application, not the users to manage the Linux system of the virtual machine (e.g. cdv).

If you have run *cdvconf* for a first configuration, you should have obtained a file named "iniadmin.p12" as result. If you cannot find the file in your computer, you could find it inside the virtual machine in the directory:

```
/home/cdv/cdvapp/sec/iniadmin.p12
```

You can retrieve this file with SSH if *cdvconf* did not provide it for you. If the file does not exist in the virtual machine, then maybe the configuration process was not successful, but you should not have been able to verify the application is running correctly.

Once the file is in your computer, the first thing to know is that this file contains the credentials for an administrator of CDV, but they are **TEMPORARY** credentials (for an "initial administrator"). They should be used to create a real administrator and then removed from all locations. The reason for this is that even though these credentials are different for each CDV instance, its password is fixed and known.

The next steps are:

- Install the iniadmin.p12 file in your computer (the password of the file is "iniadmin"). The user manual, accessible from the application itself, gives details of how to do it (see below).
- Access your CDV via your Web browser, as explained in Section 6 . The browser should ask you to pick some credentials, use the INIADMIN credentials you have just installed. If you are not asked for the credentials, **restart your Web browser** (or open a fresh "incognito" mode to make the request) and try again. It is also possible that you are using a hostname or IP that was not configured in the variable *SITE.DNSNAMES* (read comments about *SITE.DNSNAMES* in Section 5.1). Once you are authenticated as a user (either PUBLIC, INIADMIN or other) the browser will assume you are this user for the rest of the time it is running, so if you are logged in as PUBLIC in a tab, the browser will not ask you for any

certificate in a new tab. That is why the browser has to be restarted (or incognito mode started).

- Go to the User Management, and configure a new administrator and all the users you wish.
- Remove INIADMIN credentials, and all existing iniadmin.p12 files. You can keep this administrator if you wish, since the credentials are unique for your site, but be aware that the key that protects the credentials is publicly known. To remove the credentials from your computer, just proceed in a similar way as when you imported them, and delete the remaining files with .p12 extension.

The process to install the iniadmin.p12 and create the users is explained in the user manual at your CDV instance:

```
https://HOSTNAME/CelestinaDataViewer/manual.xhtml
```

7.4 Starting and stopping CDV

Starting CDV involves the execution of several software packages. The standard way is to use *cdvconf* with an Action, as has been explained in Section 6 .

Alternatively, you can start all the software manually via this script:

```
/home/cdv/scripts/cdv-start-all.sh
```

Equally, to stop all necessary software, another script can be used:

```
/home/cdv/scripts/cdv-stop-all.sh
```

Both are included in the PATH variable. The script will require to use *sudo*, so the password of the *cdv* user will have to be provided when requested. Note as well that whenever the *cdv* user logs in the system, the script *cdv-start-all.sh* is executed automatically as depicted in Figure 10 (this behaviour can be easily changed if desired).

7.5 Logs

The log files of the applications can be found under:

```
/home/cdv/cdvapp/logs/
```

When errors occur, log files can be very helpful to determine what the problem was. An Action has been created to obtain logs via *cdvconf*.

7.6 Firewall rules

By default, the firewall is configured with the following rules:

To	Action	From
--	-----	----
22/tcp	ALLOW	Anywhere
443/tcp	ALLOW	Anywhere
80/tcp	ALLOW	127.0.0.1
444/tcp	ALLOW	Anywhere

These rules can be changed at any time for you if you consider it necessary. The TCP port 22 is open for remote administration purposes. The TCP port 443 is open to access CDV. The TCP port 444 is open to serve large files of CDV. The TCP port 80 is open for local access because it might be needed for people that need to serve requests coming from SERIES. If this is not desired, the rule can be deleted.

7.7 Rebooting and shutting down the machine

The computer can be rebooted via:

```
sudo reboot
```

And shut down via:

```
sudo poweroff
```

Some *cdvconf* Actions have been created for this purpose too.

8 Further support

If you have any problems and need further support, please contact:

- Pierre Pegon at
pierre.pegon@ec.europa.eu

- Ignacio Lamata Martinez at
ignacio.lamata@oxfordalumni.org

- Patrick Petit at
patrick.petit@ec.europa.eu

7 Annex II: Celestina Data Viewer User Manual

User Manual

Contents

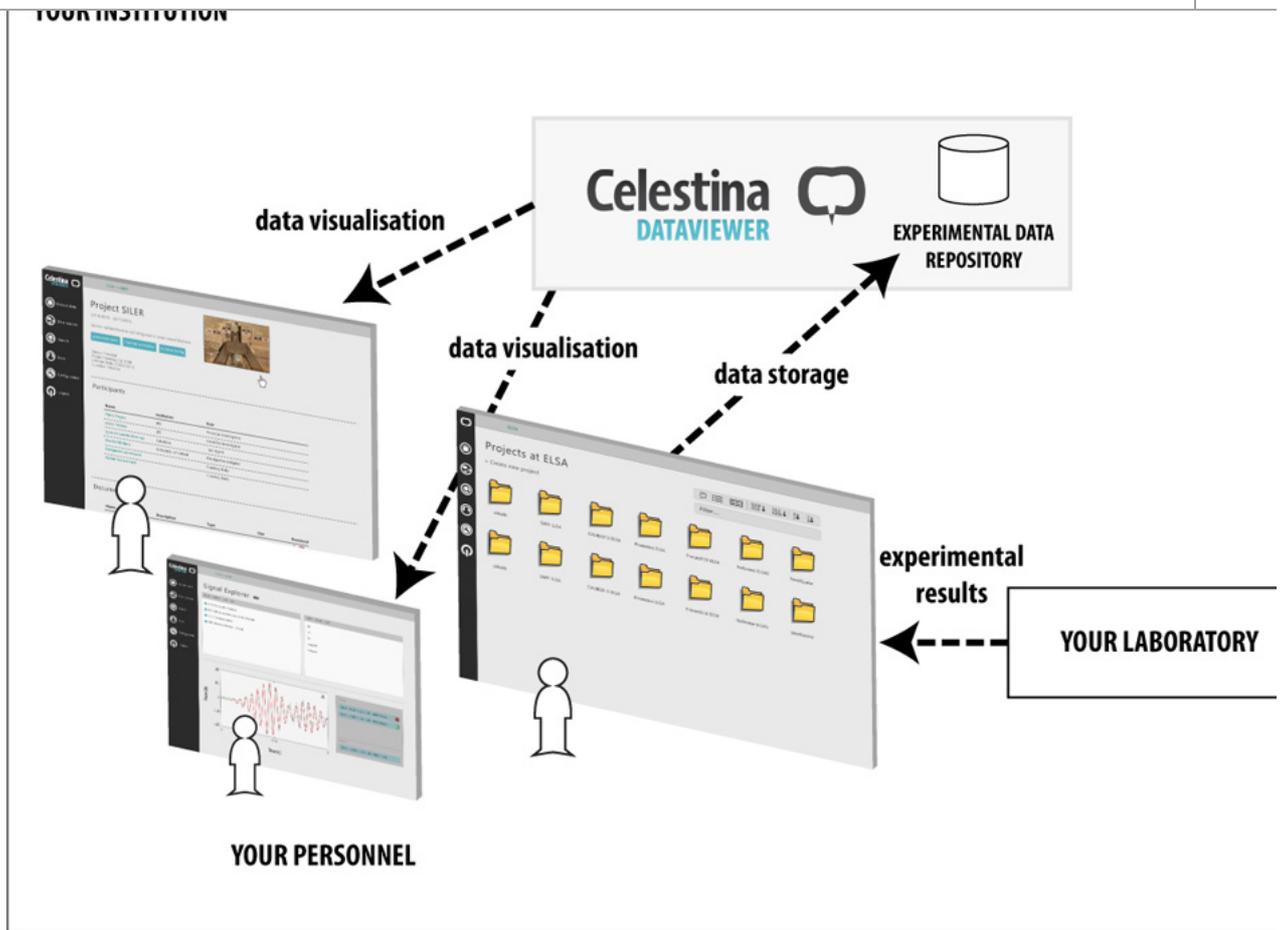
- 1 - Introduction
- 2 - Requirements
- 3 - Secure access and private data
 - 3.1 - Secure access
 - 3.2 - Getting your account ready to access private data
- 4 - Basic overview
 - 4.1 - Interface layout
 - 4.2 - The main menu
 - 4.3 - Tooltip information
- 5 - Loading data sources
- 6 - Visualising data
 - 6.1 - List of projects, experiments and specimens
 - 6.2 - Detailed data
 - 6.3 - Participants
 - 6.4 - The project map
 - 6.5 - Working with tables
 - 6.6 - Data files
 - 6.7 - Signals
- 7 - Creating and modifying data
 - 7.1 - Write operations
 - 7.2 - Creating projects
 - 7.3 - Creating experiments
 - 7.4 - Creating specimens
 - 7.5 - Working with forms
- 8 - Tools
- 9 - User management
 - 9.1 - CDV data access control overview
 - 9.2 - Creation of users
 - 9.3 - Edition of users
 - 9.4 - Deletion of users
 - 9.5 - Creation of roles
 - 9.6 - Edition of roles
 - 9.7 - Deletion of roles
 - 9.8 - User effective permissions
 - 9.9 - Project access by role
- 10 - The API
 - 10.1 - Services description
 - 10.2 - Using the API from Matlab

1. Introduction

Celestina Data Viewer (CDV) is a Web-based application to interact with distributed data sources from the **Celestina Data** network. Its main purpose is to support the interaction, storage and sharing of experimental data coming from seismic hazard mitigation institutions. CDV belongs to **Celestina Tools**, a subsystem of the **Celestina framework** which aims to create the next generation of earthquake engineering systems, based on the following priorities: interoperability, flexibility and user experience.

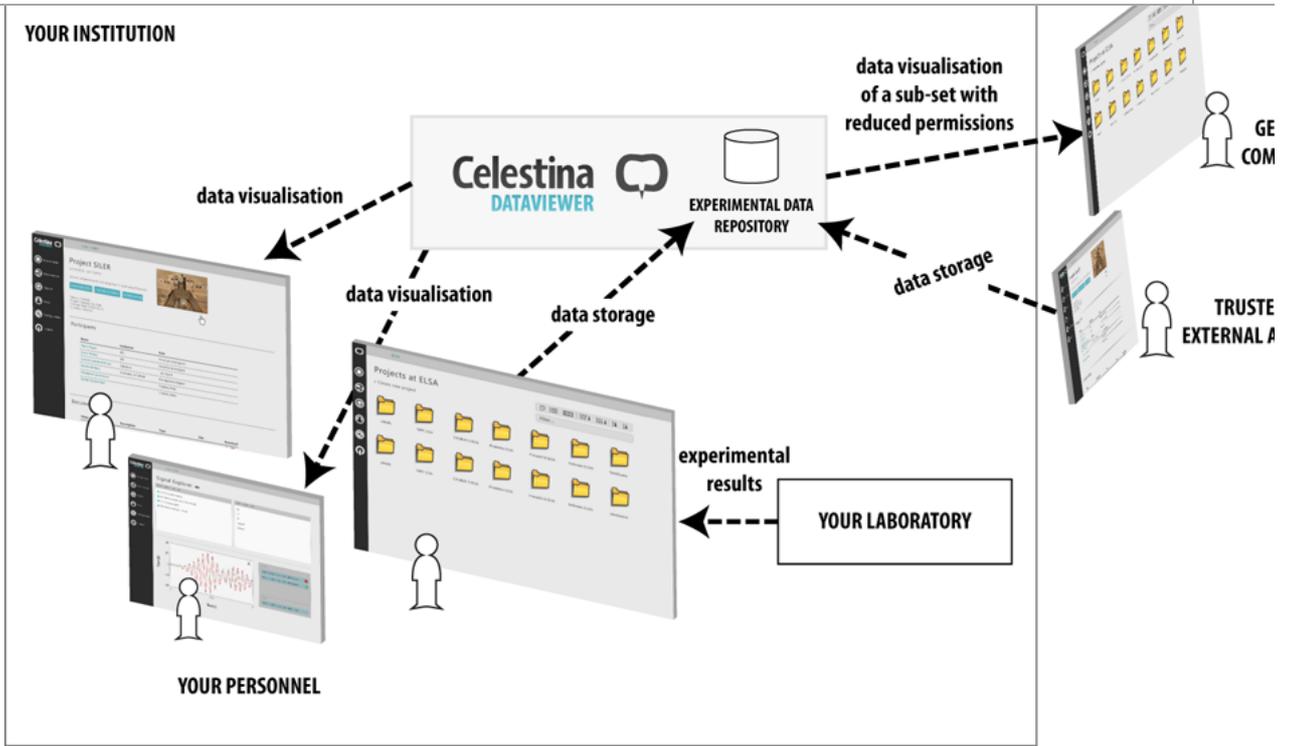
CDV is a powerful application, and has many potential uses. You can use CDV to store and work with your experimental data internally, as an isolated laboratory for example. You can also use it just to access *other people's* public data and interact with them, without having to store your own data. Similarly, you can use it to share your own data with other people or institutions without the need to access any external source of data. Note that when using data from external data sources, some CDV functionality might not be present (such as, *e.g.* editing capabilities over the distributed data - you will be able to see the data but not to edit the remote contents).

The figure below represent a typical case of how to use CDV in your institution.

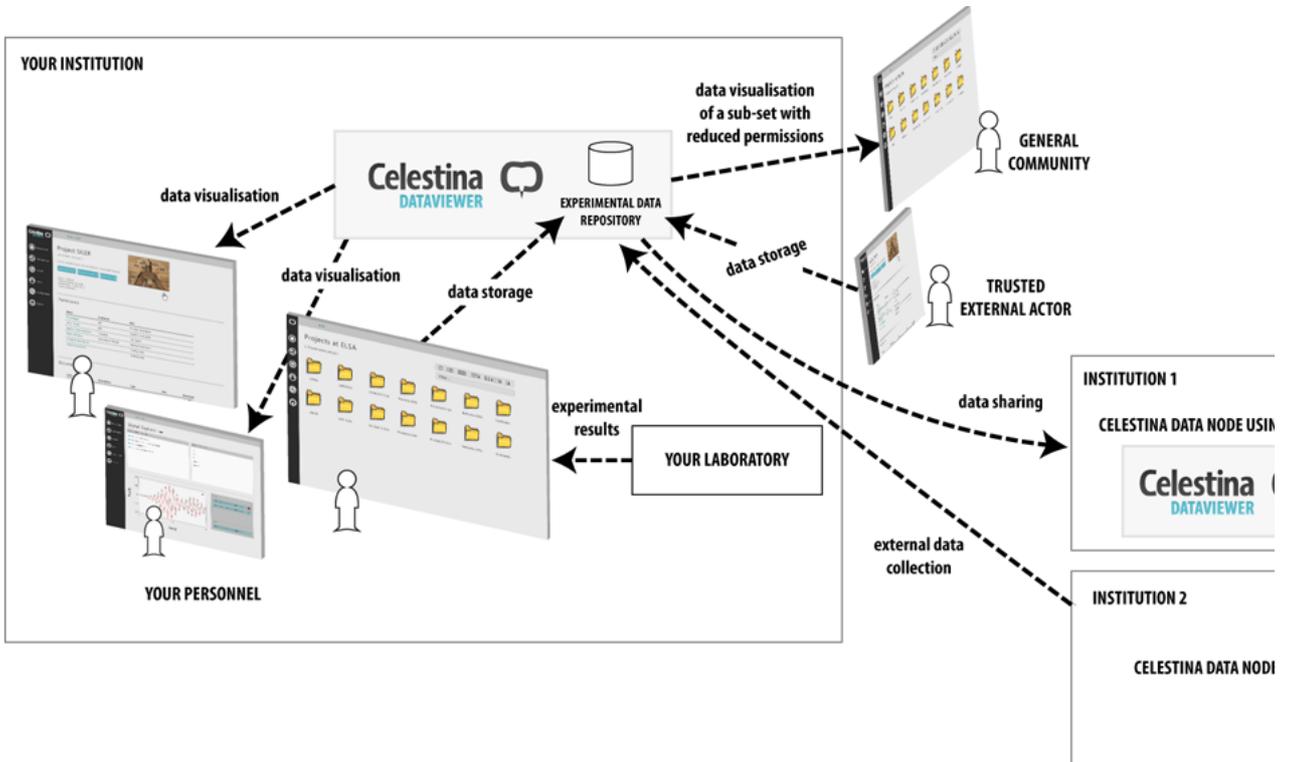


The CDV is installed on one of your institution computers and it works as an application that your institution can use internally to store experimental results obtained from experimental testing in a laboratory. In this way, CDV supports your personnel to store results in a formalised way and to visualise results in a user-friendly manner. CDV is ready to be used with just a Web browser but it can also communicate with your applications (e.g. Matlab) or your own programs created in Java, C++, Python, etc. In this way, CDV provides a complete solution for your institution to manage experimental data.

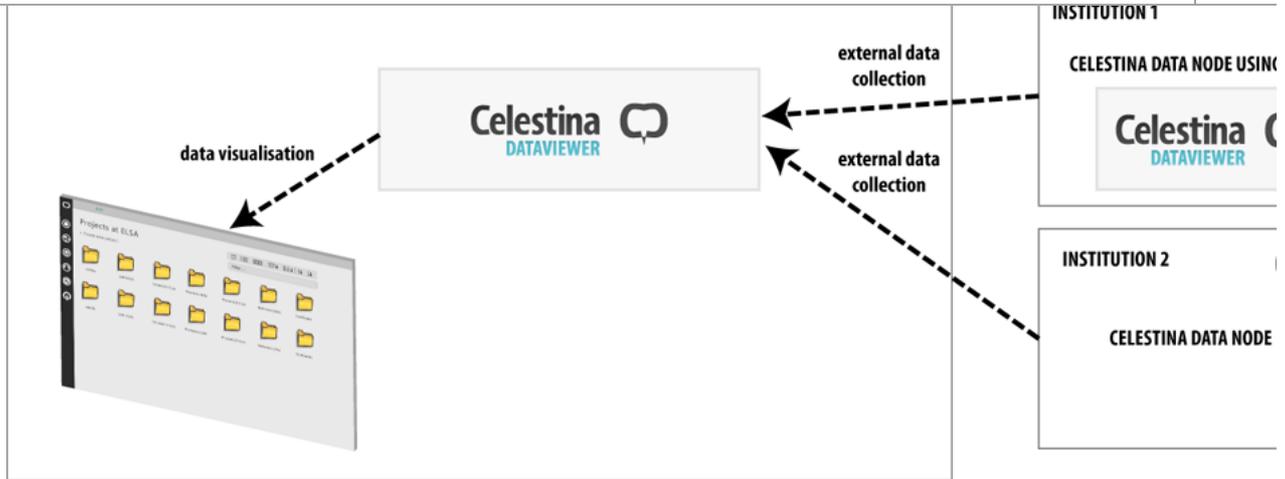
You can go one step further and open the access to the CDV to other people around the world, so they can access your data too. This access can be limited to be read-only for external users, and to access only a sub-set of your data set, so they can visualise and interact with some data but not write data into your repository. Of course you can open it to some external users so they can also create content in your repository. This situation is depicted in the following figure. Note that even though actors are represented with a person icon, they can be people or systems. For CDV, there is no distinction between dealing with humans and with computers.



As soon as you use CDV, your data will be stored in the Celestina Data format. It is easy for your institution to become a Celestina Data node if desired. This means your data will be part of the Celestina network, so they will be publicly announced for the rest of the nodes. Of course, it is encouraged, but not necessary, to share your data if you want to access data from other Celestina Data nodes. So you could obtain data from other nodes while keeping all your data private for your institution. This situation is depicted in the following image.



You can use CDV even if you do not have a laboratory or do not store any experimental data at all. CDV can be used just as a viewer to visualise experimental data publicly available in the Celestina Data network. In this case, as depicted in the image below, a repository is not necessary (from a functional point -technically, one can be used for performance reasons) and the CDV will be in charge of loading data from external Celestina Data nodes and present them to you.



As discussed, CDV can be configured and used in a variety of ways. This overview is given to understand the potential and some of the features of CDV, but this user manual mainly focuses on how to use CDV from a user point of view (from a client perspective), without explicitly describing the configuration of the aforementioned options (which imply a discussion about the installation and configuration of the application from a server perspective that is out of the aim of this manual). However, some of the features described in the options are a direct consequence of the use of CDV as a user, which will be explained in this manual.

2. Requirements

Because CDV is Web-based, there is no need for users to install additional software and a **Web browser** can be used. However, some Web browsers (especially old ones) do not manage very well some aspects of the application. Herein, it is recommended that **Google Chrome** is used, since the application has been tested on it, and other browsers (including text-based) might present some issues rendering aspects of the application. If you are using a different browser, you might experience some incorrect rendering, so make sure that your browser supports latest versions of **HTML5** and **CSS3**. Also, the browser must have **Javascript** enabled, or otherwise much of the functionality will be broken.

CDV runs partly in a server and partly in a client (*i.e.* in **your computer**) so it is important that your computer is powerful enough. In general terms, CDV is not very demanding and will run well in any modern computer. However, some features such as the graphical visualisation of project maps, the Data Universe tool or the Signal Explorer, run almost entirely in your computer, and if it cannot manage the load you might experience a bad performance. Make sure that your computer has a minimum of 4GB of RAM and a 4th-generation Intel Core i5 processor or equivalent. The better they perform, the better your experience will be. These are just rough estimations, and of course performance will also depend on how demanding the other processes you are running in your system are.

Most of the application features will work on small devices, yet mobile or tablet support is still limited. For a better experience, your screen should have at least a 1280x800 resolution.

3. Secure access and private data

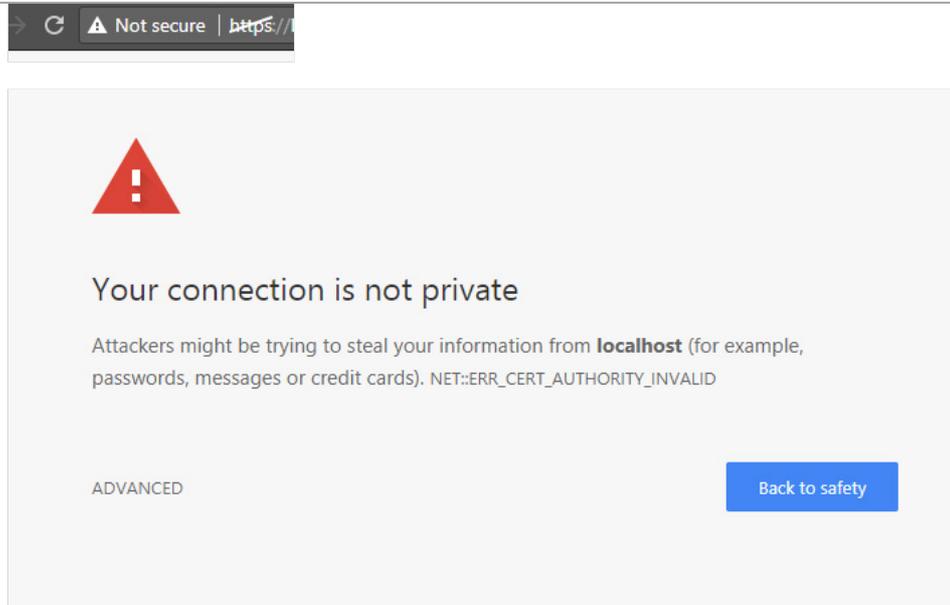
3.1 Secure access

The access to the Celestina Data Viewer (CDV) will be marked as secure or insecure by your Web browser. A secure access will look similar to the image below.



This basically means that your communication with the CDV is encrypted and that you (often, implicitly) *trust* the organisations that are certifying the identity of the site you are visiting. If this is your case, you can skip this section and go to [the next one](#), since we will assume that your connection is "secure".

The previous situation is what *should* always happen. However, it is possible that this is not the case, and the very first time you access Celestina Data Viewer you might get a security error message from your browser, similar to the errors shown in the images below.



This means that your browser is marking the site as insecure, which can happen for a variety of reasons, some more malicious than others: your browser does not trust the system you are trying to access (like specifically depicted in the image above), there is a problem with the encryption, there is a configuration error in the security established for the communication, there is some parameter in the communication that your browser considers potentially unsafe, there is something/someone trying to eavesdrop or alter your communication with CDV, etc.

In many cases, one of the reasons for this to happen is when the CDV installation you are accessing has not been trusted by any of the vendors that your browser trusts by default. In other words, the people at ELSADATA did not acquire a certificate from an external party that you already trust. If that is the case, **it is OK**. If you trust ELSADATA (which should be the case) then you should tell your browser/system to trust this CDV. The way to do this is explained below in this section.

However, a note of caution before. You might also receive this security warning because someone is genuinely eavesdropping the communication. If this is the first time you access ELSADATA, it is difficult to tell the difference. And there are quite a few tricks to lead you into believing their certificate is genuine. So if you wish to be extra cautious, before downloading your credentials, you can take extra measures: (i) At least when you export ELSADATA certificate, try to connect from a wired network you really trust (like your home rather than your workplace or university - this is only possible if there is external access to CDV). (ii) Verify the CDV address you are accessing from your browser looks OK, contains "https://", the hostname is correct and is the same as the one in the ELSADATA certificate, etc. (iii) Ask an administrator at ELSADATA either to send you ELSADATA's CDV certificate via a *secure* channel or to confirm that the certificate you are exporting from the steps below is the correct one. The weakest point of the system is to accept an insecure connection when you are going to download your credentials, since they could be copied at that time by a malicious user if you have either an insecure connection or a secure connection with the wrong destination. If you are very concerned about security, you can create your own credentials and agree with ELSADATA a way to register with them (which is the way it should be done, but this creates a bit of burden for everyone involved). Most of the people will not need to take so many security measures, but it is our responsibility to inform you of some of the risks.

So this is how to install ELSADATA CDV certificate in your computer, to explicitly trust it:

Chrome for Windows

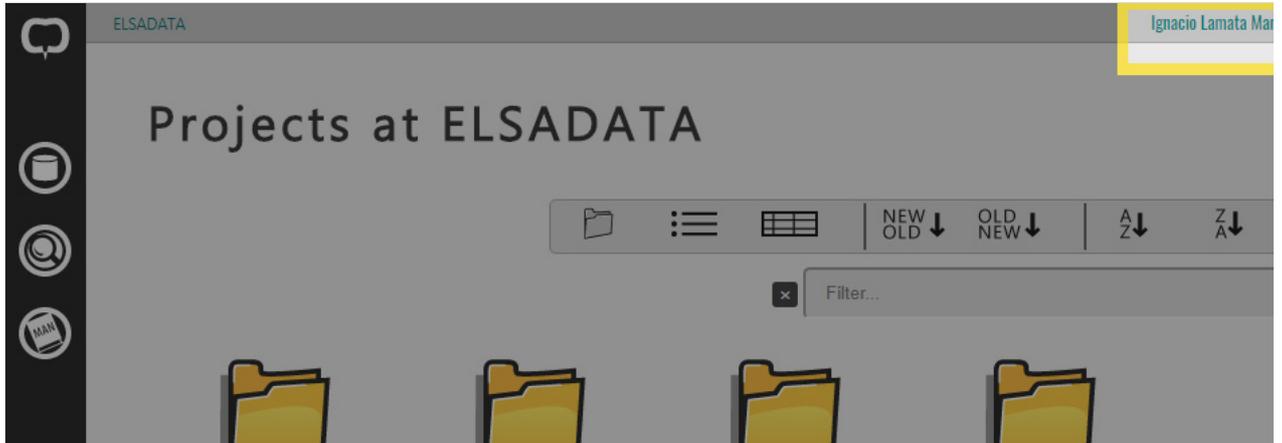
Installing a certificate as a trusted Certification Authority **involves some risks** and you should verify the following. ****TODO****

3.2 Getting your account ready to access private data

NOTE You can skip to [the next section](#) if you just want (or only have permission) to access the public data.

CDV is designed to work with both public and private data. To access the private data, as a user you need to be registered in the system, and to do so you need to contact the person in charge from the institution where CDV is installed. This person should send you **a link** (not a file), so that you are able to create your credentials without anybody else intervention (note that it is still possible to send you a link that automatically downloads a file or that fakes the creation of your credentials, so you need to trust them. In the end, it is their system which you are using). Once you access the link, you will see full instructions of the process to create your credentials. It should involve the selection of a personal password and the download of a file with *.p12* extension containing your credentials. The password you choose is just to protect this file. This file has to be installed in your computer (or any computer you use to access CDV). The way to do it is the following:

Once your credentials have been installed in your computer, you can store the .p12 file in an external *secure* place as a backup (or to be installed in other computers), and delete it from your computer. Please **restart** your browser before accessing CDV, just in case it has not detected you have imported a new certificate. Once you do so, you should be asked by your browser to *choose a certificate* and you should see the one you just imported listed. Choose it and CDV should load correctly, showing your user name at the far right of the navigation bar as depicted in this image:

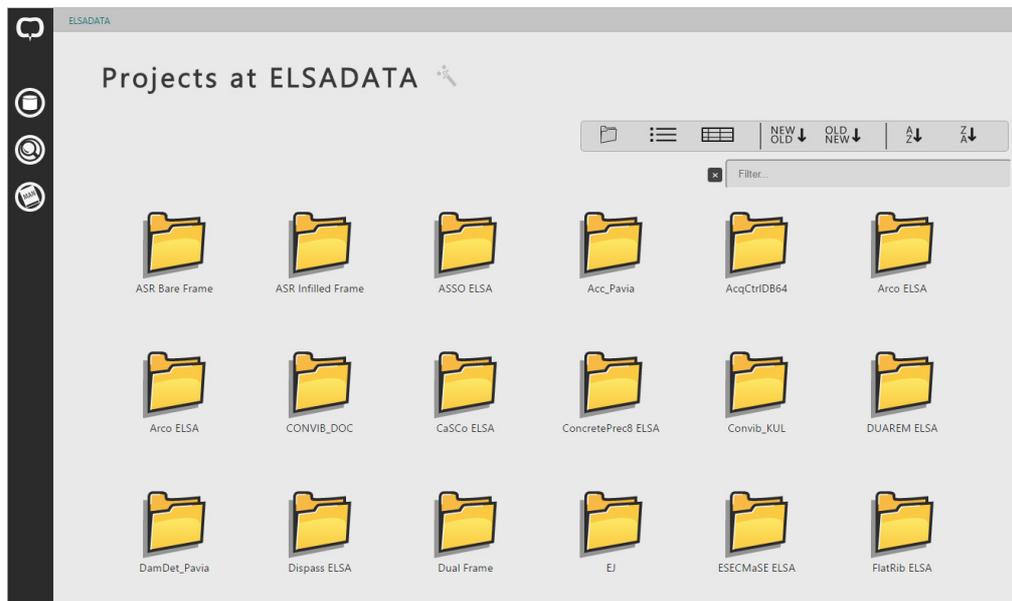


A note for users of the system - Security has been considered in CDV from its design stage. CDV uses strong cryptography for the authentication of users, and provides mechanisms to guarantee data privacy. Users prove who they are with a relatively transparent process, free of burden, in which they provide some information that is only known and stored by them, and not stored in CDV in any way (if no malicious modification has been done in CDV, of course). Data privacy in CDV follows a users-roles model that balances functionality and easiness of usage and maintenance. However, note that security is a perception, it is **a process, not a product**, and no system can be considered fully secure. There are many ways to escape or break these security measures with enough time, knowledge, persuasion skills or money.

4. Basic overview ↑

4.1 Interface layout ↑

The basic layout of the user interface has 3 main elements: (i) a main menu, which is folded on the left side of the screen, (ii) a navigation bar, on top, and (iii) the main content, covering the rest of the screen. This can be seen in the following image (hold over the image to see the 3 elements).

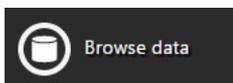


The **main menu** shows all general operations that can be done on the application and it is described in the next section. The **navigation bar** shows the data sources that have been loaded (in the image, a single datasource called *ELSADATA* is loaded) and the location inside the datasource where you

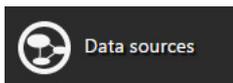
ELSADATA > Example Project > EOL. Also, at the far right of the navigation bar, the logged user name is shown. This is only relevant if you need to access private data. Finally, the **main content** represents the main area of content interaction.

4.2 The main menu

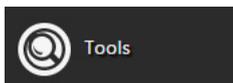
All operations in the user interface can be accessed via the main menu, under each specific option. By default, the main menu only shows icons on a vertical bar. By placing the mouse over the vertical bar, the main menu is fully displayed to show a textual reference for each different menu option. The Celestina Data Viewer logo lies on top of all options, which are described below:



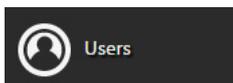
Browse data: This option provides the main functionality of the CDV. It enables access to all data that has been loaded into CDV from the data sources. After using this option, a list with all the projects available will be shown (see Section [Visualising data](#)), and from this point the data can be navigated or edited.



Data sources: This option enables users to load new data into the CDV. Each Celestina Data repository is called a *data source*. Data sources can be found wherever a public Celestina Data node exists. So, for example, the University of Oxford can make a data source publicly available that external users in Japan can load into their CDV application to visualise it. The ELSA laboratory at the JRC can make 2 different repositories publicly available too. By using CDV, a user could load the data source from Oxford and the 2 data sources from ELSA and visualise all data within the same CDV transparently, as if the 3 repositories were coming from a single local database. Normally, an institution will always have by default their own local repository loaded, which can be accessed with the *Browse data* option. However, CDV is a tool that can be used with no local repository at all, by loading data from distributed repositories as explained before. It can also be used with a local repository that is enriched with data from other distributed repositories, visualising all loaded data sources as if they were the same source. The [Loading data sources Section](#) provides more information.



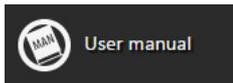
Tools: This option gives access to the tools available in CDV. Tools are gradually increasing and provide new features to CDV. Examples of tools are *Signal Explorer*, which enables users to plot signal data from the experiment results, *Data Stats*, which presents statistical data about the loaded data sources such as number of projects finished per year or the distribution of signals per experiment, or *Data Universe*, which displays an undirected graph of all loaded data, representing projects, specimens and experiments as nodes. Tools are further discussed in [the Tools Section](#)



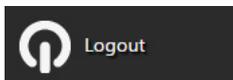
Users: This option enables the management of access permissions in CDV. From this option, different users can be created, who can be granted access to the application. In order to distinguish which data or features each user can access, roles are used. By default, access to any part of CDV is restricted, and different roles should be created to allow users to access the information in the repositories or the allowed tools. This option is further discussed in [the User Management Section](#)



Configuration: This option permits the configuration of some internal aspects of CDV. Some aspects of CDV should be



User manual: This option opens the user manual that you are currently reading! Contained inside CDV, in here you will find help about how to use CDV.



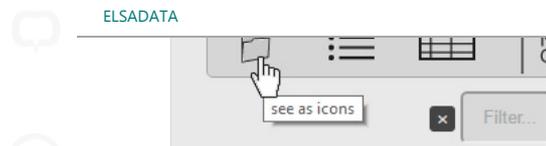
Logout: This option logs the user out of the system.

NOTE Some of these options might not be available in development versions of the CDV

4.3 Tooltip information

Most of the features of CDV are quite intuitive and can be naturally found/used without reading this user manual. However, some specific operations (e.g. table multisorting, multiselection, etc) can be a bit more tricky to find, and in these cases this manual can become handy.

When you do not know what a link of an icon does, hold the mouse pointer over the link and wait for the help tooltip to appear. This will give you a hint on what to expect when you click the link.



Try it yourself



5. Loading data sources [↑](#)

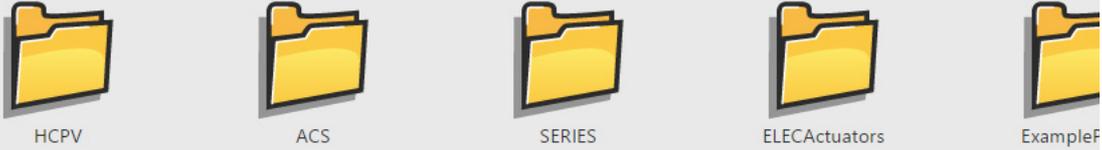
This feature is not currently available.

6. Visualising data [↑](#)

6.1 List of projects, experiments and specimens [↑](#)

The visualisation of projects, specimens and experiments starts with a list of elements (for clarity of reading, this section will refer to the visualisation of projects, but, unless stated otherwise, what is discussed for projects applies to specimens and experimental activities in a similar way). There are 3 options to visualise this list of projects, as depicted in the image below: (i) as icons (the default one), (ii) as a list, and (iii) as a table. Each option has its own advantages when showing data, depending on how the user wishes to find or sort the data.

Icon visualisation



List visualisation

- HCPV
- ACS
- SERIES
- ELECActuators
- ExamplePROJ

Table visualisation

Name	Start date	End date	Status	Description
HCPV			Foreseen	
ACS			Foreseen	
SERIES	1/2/2009	1/7/2013	Finished	The SERIES European project
ELECActuators	5/2/2017		In Progress	
ExamplePROJ			Foreseen	

Icon visualisation is the default visualisation, and it is ideal when only names matter. When holding the mouse over the folder icon, a box will appear with additional information about the element, such as a description, status, dates, etc. When a project has a short name, this is the preferred name that is displayed below the folder icon (in this situation the project full name, if exists, can be seen in the appearing box). If it has no short name, its full name is displayed. If no short name nor full name exists, the identifier of the project is displayed. Since identifiers are not easy to read, it is strongly recommended that all projects have a short name and a full name. Also, very long names below the icon will be truncated for presentation reasons. Different icons are used for different elements: projects, specimens, physical experiments, numerical computations and hybrid tests/simulations, which makes it easy to make distinctions in a first glance (especially in a list of experimental activities with both physical and numerical experiments). When placing the mouse over a project icon, a box appears to show further details. To read this details, place the mouse over the box.

List visualisation has a similar utility to icon visualisation, with the difference that it might be more suitable for a large number of projects, since only the name is displayed, with no icon. Placing the mouse over a list element will show the same box with more details as in the icon visualisation.

Table visualisation includes more features to visualise data, since it displays available information in a tabular manner, which allows easier comparison. Tables enable sorting by any of their columns (see Section [Working with tables](#) for more detail). Visualisation tables slightly differ between Projects, Specimens and Experiments, depending on the requirements of the element. For example, the visualisation table for experiments shows the specimens that are used by that experiment.

In any of the 3 visualisation types, a project is opened in a new screen for deeper visualisation by clicking on the name of the project (see Section [Detailed data](#)). Below any visualisation, the total number of elements is displayed (note that filtering the display of results does not alter this number).

On the top right side of any visualisation of projects, specimens and experiments, there is a tool bar that allows users to swap the visualisation option as well as to find and sort the different elements of interest. The options , ,  will change the visualisation to *icon visualisation*, *list visualisation* and *table visualisation* respectively. Then, some sorting options are available: by date  (newer to older),  (older to newer), or alphabetically by name  (ascending),  (descending). By default, elements will be sorted alphabetically, but the projects with no short name (or acronym) defined will be displayed first; herein, projects with no short name will be displayed first, alphabetically sorted, and then projects with a short name will be displayed after, alphabetically sorted. If an absolute sorting is desired -including projects with and without short names in the same bag, the A-Z sorting option should be used. As mentioned before, the table visualisation includes more sorting options than the other 2 visualisations, since it allows explicit sorting by any of its columns. When a sorting option is not possible to do (e.g. sorting by date over a list of projects with no

ELSADATA

relegated to the last positions when sorting both ascending and descending. Sorting is quite uniform between all visualisations, but slight mismatches might occur on very specific cases. For example, table sorting considers "SAFE" to be alphabetically prior "SAFE ELSA", whereas the icon and list visualisation will consider the opposite.

Below this tool bar there is a filter. The filter enables limiting the number of elements displayed, according to their names. By using a filter of "S", any element that contains an "S" in its name will be displayed. Note that this does not necessary mean that the "S" is at the beginning of the name, the filter will show names like "SERIES", "TAS", "NEES" or "ANSE", as long as they contain an "S". If "SE" is used, the filter will be more restrictive and only elements that contain "SE" will be displayed, such as "SERIES" or "ANSE". To clear the filter, press on the cross on the left or delete manually the characters introduced. Visualisation and sorting options can be used and changed while a filter is applied.



SERIES

MY PROJECT

uWalls

HCPV

ELECActuators

ACS

ExamplePROJ

6.2 Detailed data ↑

When accessing an element (Projects, Specimens and physical or numerical Experimental Activities) in a visualisation list, a new screen is opened with detailed data about the element. Detailed data screens show information uniformly to all of them, but they might present specific information only relevant for a type of element, such as signals for an experiment or specimen components for a specimen. Some of these features are discussed in following sections.

When some information is missing, by default CDV will normally omit it by not showing anything about it. So, for example, a project without a start date will show nothing instead of "unknown start date". This improves the user experience in most of the cases. Detailed data will normally start with a header as depicted in the following image (hold over the image to see the different data names).

Project SEISMIC2017 ✎

Seismic Project for testing the assessment and rehabilitation of bridges

22/06/2017

InProgress

Purpose: Testing on bridges

Beam bridges are horizontal beams supported at each end by substructure units and can be either simply supported when the beams only connect across a single span, or continuous when the beams are connected across two or more spans. When there are multiple spans, the intermediate supports are known as piers. The earliest beam bridges were simple logs that sat across streams and similar simple structures. In modern times, beam bridges can range from small, wooden beams to large, steel boxes. The vertical force on the bridge becomes a shear and flexural load on the beam which is transferred down its length to the substructures on either side[12] They are typically made of steel, concrete or wood. Beam bridge spans rarely exceed 250 feet (76 m) long, as the flexural stresses increase proportional to the square of the length (and deflection increases proportional to the 4th power of the length).[13] However, the main span of the Rio-Niteroi Bridge, a box girder bridge, is 300 metres (980 ft)

reinforced concrete structure bridge cyclic testing pseudodynamic

The big title of the project will show the short name, but in the case of a missing short name the name would be used instead. If no names can be found, the identifier of the project will be used, which, as said before, it is not easy to read so the use of names and short names are strongly encouraged.

In the example image, the project does not have an end date, so no end date is shown. If it has both dates, they both will be displayed, but if only an end date exists, then a reference to a missing start date will be displayed (N/A - not available), so users can realise that the shown date is an end date. The status is inferred from the dates of an element. If an element has an end date, it is considered **Finished**. If it has no end date, but a start date, it is considered **In Progress**. If it has no dates, it is considered **Foreseen**. This rules might differ slightly on elements with only one date (e.g. elements without end dates). In such cases, the element is considered foreseen if it has no date and finished if it has a date. This rules improve the user experience by eliminating the need to explicitly state an element status.

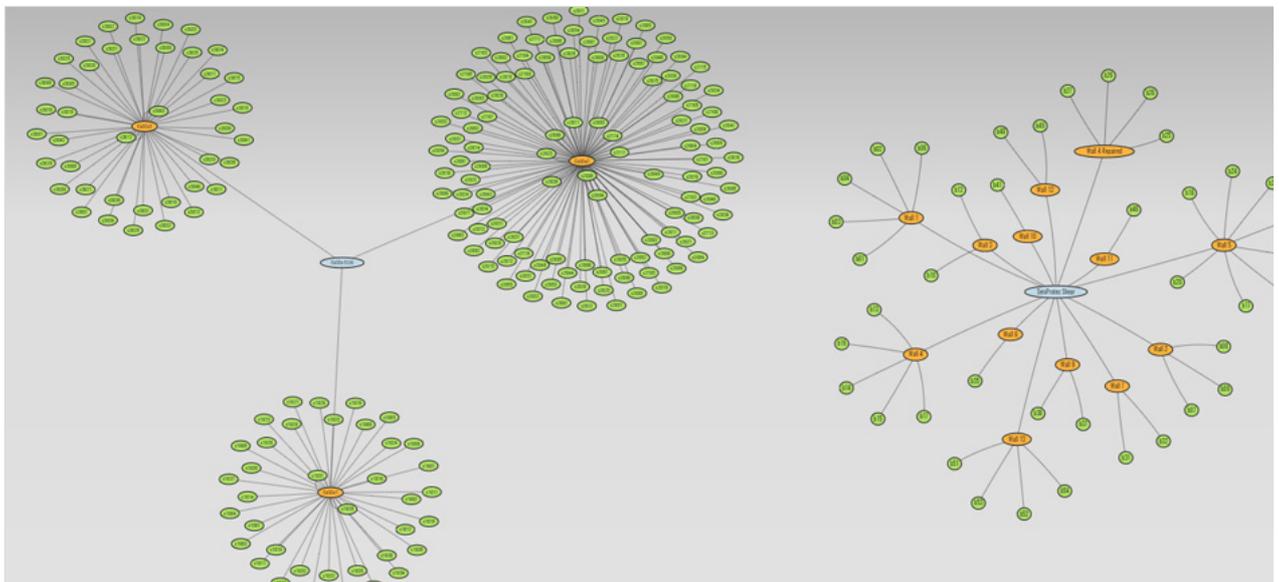
6.3 Participants ↑

A list of participants is shown for Projects, Specimens and Experimental activities (P/S/E). Each participant has a role, according to how they contributed in the specific element. The *General Participant* role (or, simply, *Participant*) is given when someone participates in the P/S/E in some way, but it is not specified. Note that participants can be persons and organisations.

In Projects, the list of participants displayed also contains the participants of its Specimens and Experimental activities. So when someone participates in an experiment, it is inferred that he/she also participated in the project that contains such experiment.

6.4 The project map ↑

The project map is shown inside the detailed data for the project and it is a great way to have a high level view of the project in a single shot. The project map shows the relations between a Project and its Specimens and Experimental Activities, which are all represented as nodes in an undirected graph.



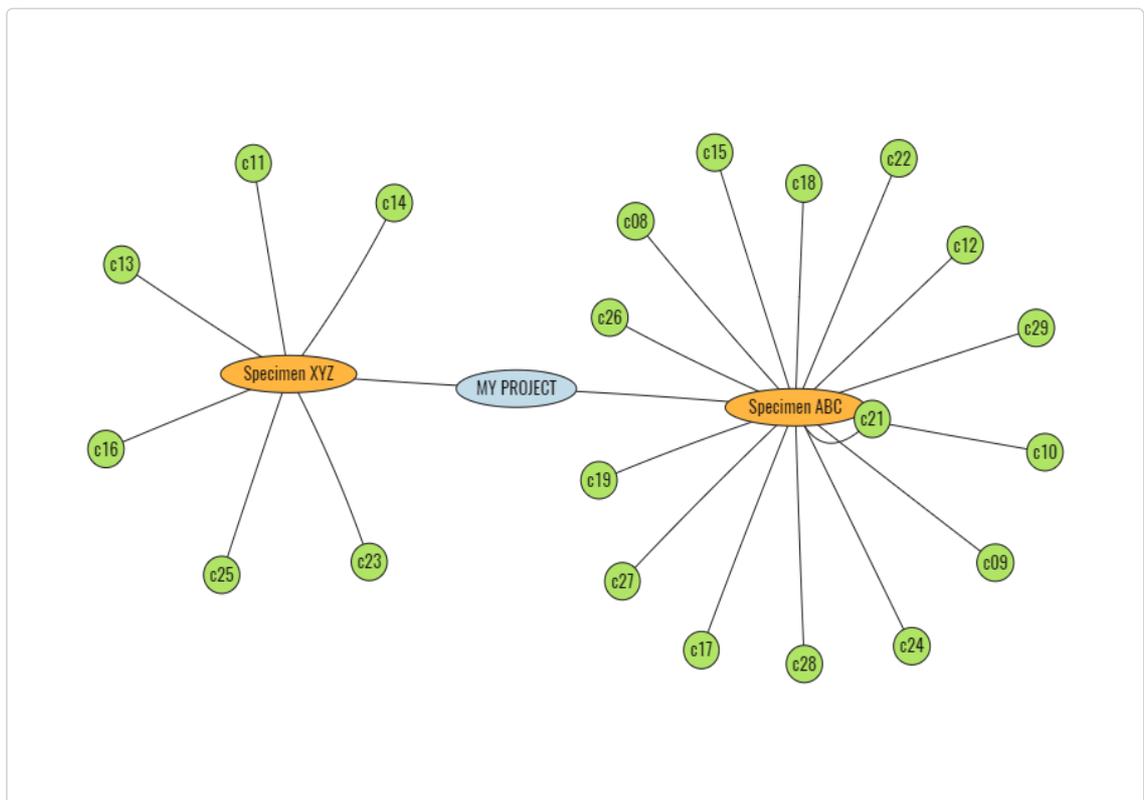
project will be then represented as a relation in the graph. However, when an experiment is also related to a specimen, the experiment will lose the explicit link to the project for clarity and will keep only its link with the specimen. Therefore, the graph will make connections according to a **P-S-E** pattern (Project -> Specimens -> Experiments). This fits nicely both situations: labs that conduct many experiments over a few specimens and labs that use only one specimen per experiment. The pattern **P-E-S** makes sometimes less clear graphs. Normally, numerical experimental activities (e.g. a computation) will be directly linked to the project.

Initially, the project map is static and does not allow any interaction. It can be made interactive by clicking on any place of the project map, which will change the colour of the background. To get back to the static state, a click outside the project map has to be made. When the project map is interactive, some functionality is enabled. For example, while the project map is interactive the view can be moved (by clicking on an empty space and dragging) and zoomed (by using the mouse wheel). Also, nodes can be moved and rearranged up to a certain point (the graph will continuously attempt to get in a stable state, by reorganising its nodes automatically so they do not cover each other). A **single click** on a node will highlight the node and all its relations, and some information about the node will be shown in the top right corner of the project map (note that this information is obtained on demand and might take some time to appear). A **double click** on a node will open a screen with detailed information about the element (e.g. it will open the detailed data view of an experiment) on the current screen. A **double click** while pressing the **SHIFT** key will open the element in a new screen, so the current screen will not be replaced.

Project map's graphs can be saved as transparent PNG images by making the graph interactive, clicking with the second button of the mouse over it and choosing "Save image as". The exported image will have a transparent background and will be the actual image displayed in the project map, including position and zoom.

The project map runs partially on the user computer, so if there is much data to load and the computer is not powerful enough, it can take some time to load. For this, a loading progress indicator is shown during the loading process.

Try it yourself



Legend Projects Specimens Physical (experiment) Numerical (computation) Hybrid

6.5 Working with tables

Much of the information accessed by CDV is displayed in tables, which are obviously excellent to display tabular data. The first operation that can be done with tables is **sorting**.

To sort any table within CDV, click on a header column, and the contents of that column will be sorted in a descendent way. You will see a small arrow on the right of the header column indicating the direction of the sorting. If you wish to order in ascendent way, just click again over the same header

ELSADATA

the **SHIFT** key. Note that if you sort by a column that has no similar values, this will effectively *lock* the order of the rows, and subsequent multisorting attempts with other columns will have no effect because the rows are locked by the first sorting.

University	Department	Contact person
Oxford	Engineering	Williams
Oxford	Mathematics	Smith
Trento	Engineering	Mancini
Bristol	Physics	Smith
Salamanca	Art	Alonso
Oxford	Engineering	Bridge

In the animation above you can see how the first sorting locked the row with *Art* in the first position, so the following multisortings did not change its position. Only the rows that share values are affected by the following multisorting (like the *Engineering* rows for the second sorting and the *Engineering - Oxford* rows for the third sorting).



University	Department	Contact person
Oxford	Engineering	Williams
Oxford	Mathematics	Smith
Trento	Engineering	Mancini
Bristol	Physics	Smith
Salamanca	Art	Alonso
Oxford	Engineering	Bridge

Next operation with tables is **selection**. You can select any row by clicking on it, and you will see that the row is highlighted and an indicator on the top right will inform you how many rows are selected. To deselect the row, just click again on it. To select all rows at once, you can use the *Select all* feature. If you click on the Select all icon, all rows will be selected at once. This is a useful way to count how many rows are present in a table. To deselect all at once, just click again on the *Select all* icon. When a few rows are selected, a quick way to deselect all rows is by clicking twice over the icon - the first time it will select all rows and the second will deselect them all.

Multiple selection is also possible on table rows. Just click a row to select it and then click another row somewhere below while pressing the **SHIFT** key. This will select all rows that are between them. This is useful to select rows with a similar characteristic. For example, sort the table below by Department, select the first row that says Engineering, hold the **SHIFT** key and finally select the last row that says Engineering. All the Engineering rows will be selected in block.

Although most tables in CDV can, note that a few of them cannot be selected.



University	Department	Contact person
Oxford	Engineering	Williams
Oxford	Mathematics	Smith
Trento	Engineering	Mancini
Bristol	Physics	Smith
Salamanca	Art	Alonso
Purdue	Engineering	Roberts
Purdue	Engineering	Sanchez

Sometimes, it is useful to **hide** some rows. This allows you to focus only on specific rows for comparison or to prevent some undesired rows from being exported or copied. To hide rows, simply select them and then click on the *Hide selected rows*  feature. An indicator of how many rows are hidden will appear on the top right. To reveal all hidden rows, click on the icon when no rows are selected. This feature is only available on tables that can be selected.

Try it yourself



University	Department	Contact person
Oxford	Engineering	Williams
Oxford	Mathematics	Smith
Trento	Engineering	Mancini
Bristol	Physics	Smith
Salamanca	Art	Alonso
Purdue	Engineering	Roberts
Purdue	Engineering	Sanchez

Every table in CDV can be **exported**. This is an extremely useful feature, since you can export any table in the application and use it in reports or work with it in Excel, Matlab or any other. To export a table, click on the *Export table*  feature, and it will be automatically downloaded in CSV (Comma Separated Values) format. The exportation process will generate the exact table data that you are seeing, meaning that it will respect the order of the rows as they have been arranged and will not export rows that are hidden or filtered.

NOTE Very old versions of MS Excel (e.g. MS Excel 2007) will have difficulties interpreting UTF-8 characters shown in the CSV, such as s^2 . This can also happen in some modern versions of MS Excel. If that is the case, import the CSV file manually in Excel specifying the encoding UTF-8.

Try it yourself



University	Department	Contact person
Oxford	Engineering	Williams
Oxford	Mathematics	Smith
Trento	Engineering	Mancini
Bristol	Physics	Smith
Salamanca	Art	Alonso
Purdue	Engineering	Roberts
Purdue	Engineering	Sanchez

6.6 Data files

Data files are the large files associated with an element, such as documents, executable files, configuration files, images and videos. They are visualised in a single table to enable sorting and comparison, but other visualisations might be implemented in future.

Data file tables have all the features of regular CDV tables, and as such they can be sorted by any column, exported, selected and hidden (see Section [Working with tables](#)). The actual files can be downloaded in three ways:

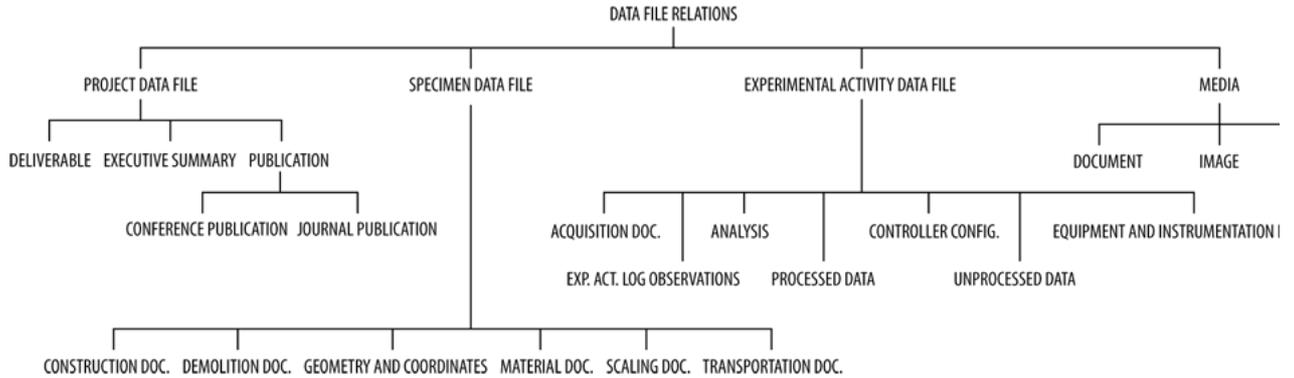
1. By clicking of the **data file name**. This will trigger the default option configured by the user in the browser of his local computer. For example, when using Chrome, by default any image that can be rendered by Chrome will be displayed in the browser itself, TXT and PDF files will be opened in the own browser, but other file formats that are not understood or cannot be rendered by Chrome will be downloaded.
2. By clicking on the **download icon next to the file name** (). This will force the download of the file to the user computer, regardless of what the default option of the browser is. This is very useful when the user wants to download a file, such as a JPG image, instead of having it displayed inside the browser.
3. By clicking on the **download icon on the top right bar** (). This option enables multiple file download. It will download all files in the table that are selected, inside a compressed ZIP file. Note that all files selected have to be retrieved and compressed in a single file, so if the files are very large (*e.g.* a large video) then this download process might consume substantial time and resources. In such cases, it is suggested that users use the single download icon instead, and download very large files one by one or in small groups.

Data files						
Name	File type	Description	Relation	Creation date	Size (MB)	
Project description	 PDF	Description of the project, objectives, scope, participants and budget	Executive summary	21/07/2017	0.01	
Group image	 JPG	Photo of the participants	Photo	10/07/2017	0.0095	
Structures	 XLSX	Specimens tested in the project	Document		0.0097	

The File type column will show the type of file based solely on the file extension.

or the project, or might contain some images or the construction of the specimens in a project. This information is contained in the **relation** property. Relations do not really say anything about the actual file type (although sometimes relations might coincide with file types) but about what the meaning of the file is for a specific element. Herein, there could be a PDF file with a relation of specimen *Image*, if it contains an image of the specimen. Or a PDF, DOC or XLS file with a relation of *Executive summary* for a project if they have contents that represent a summary of the project, even if it is an excel file with some project figures. Another example is a scanned image of a document, whose file type can be JPG but their relation is actually a project *Document*. Relations can have sub-relations. So for example, a deliverable and an executive summary is a sub-relation of a project document. There are specific relations that only apply to Projects, Specimens or Experimental Activities (e.g. specimen construction documentation only applies to specimens), whereas some others are general (e.g. Media Relations, consisting of general documents, images and videos apply to Projects, Specimens and Experimental Activities equally). Relations can be imagined as "folders" where the data files are stored in a structured hierarchical way.

Relations are not fixed and can be expanded by users, according to their needs. This provides a great flexibility to the CDV. The initial hierarchy of relations can be seen in the following figure.



Having a table to represent data file information is useful to extract data in an organised manner. So, for example, if a user wishes to download all the configuration files of a controller in an specific experiment, all the user has to do is to sort the table by the column "Relation", find the "Controller configuration" relation, click on the first row of such relation, click on the last row while holding the **SHIFT** key, and finally click the download icon () on the top right. This will download all controller configuration files at once.

Try it yourself



Name	File type	Description	Relation	Creation date	Size (MB)
Tragicomedia de Calisto and Melibea (ES)	PDF	A Spanish 15th Century novel	Document	04/07/2017	0.57
Tragicomedy of Calisto and Melibea (EN)	TXT	A Spanish 15th Century novel	Document		0.93
Cover	JPG	A cover of the book	Image	05/01/2015	0.11

6.7 Signals ↑

7. Creating and modifying data ↑

7.1 Write operations

These are the basic icons that represent means to alter data in the repository:



New: Create a new element. This icon will normally appear at the beginning of a section, to create elements that are displayed in that section.



Edit: Modify the information of an element. This icon will usually appear near the name of the element.



Delete: Delete an element permanently. This icon will usually appear near the name of the element.



Clone: Make a copy of an element, including its references to other objects. From example, if *experiment A* is cloned, a new *experiment A'* is created, which contains the same information as the original one, and includes its relations such as the specimens that the experiment uses. Note that this specimen is not duplicated, it is the same for both copies, *experiment A* and *experiment A'*. This feature is useful to quickly create new elements that share most of a common structure with other elements. Then the cloned copy can be expanded or modified as necessary.



Link: Create a reference to an existing element. The element is not copied, it is just referenced.

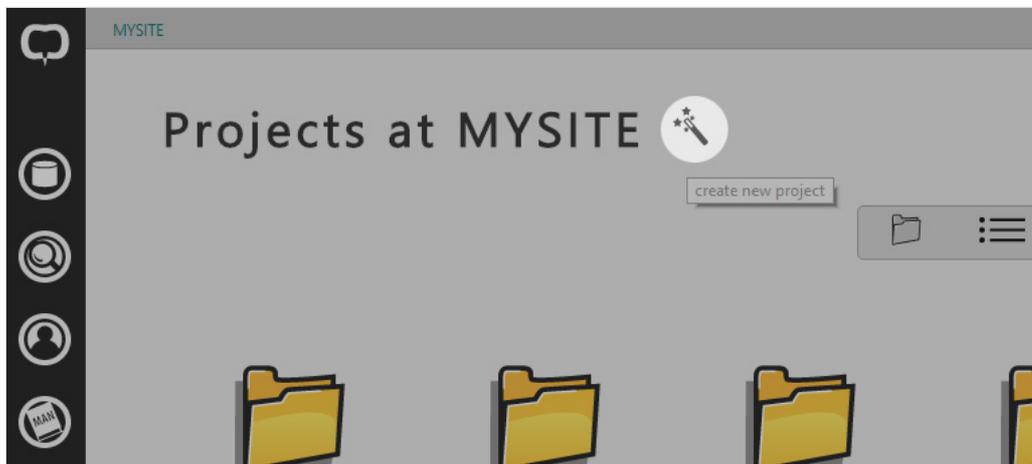


Add and remove: Adds something to a list or removes something from a list. It is common when an element contains lists of other elements (e.g. list of keywords, materials, participants, etc).

NOTE To use these features, you will need special write permissions.

7.2 Creating projects

Projects can be created by using the  icon from the project list, as depicted in the image below. Note that administration privileges are necessary to create projects.

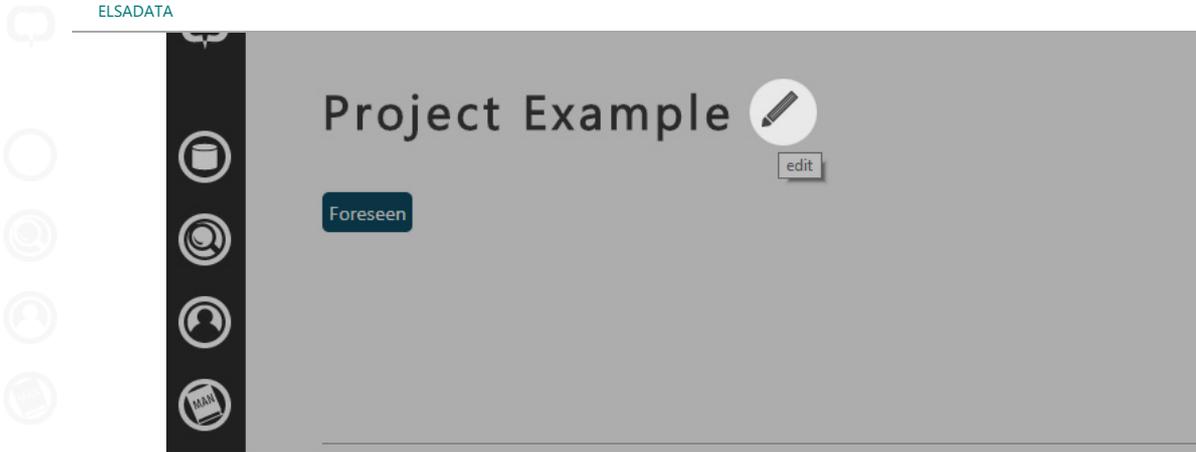


This will open the edit-project page, where all the information about the project can be filled in by using [forms](#).

7.3 Creating experiments

To create experiments, it is necessary to have write permissions on the project where the experiment is going to be created. Experiments are always created from an existing projects, so no experiment can be created in the wild without being linked to a project.

Experiments are created from the edit screen of the project, so the first step is to edit the project where the experiment belongs, by using the *Edit* icon near the project name, as depicted below.

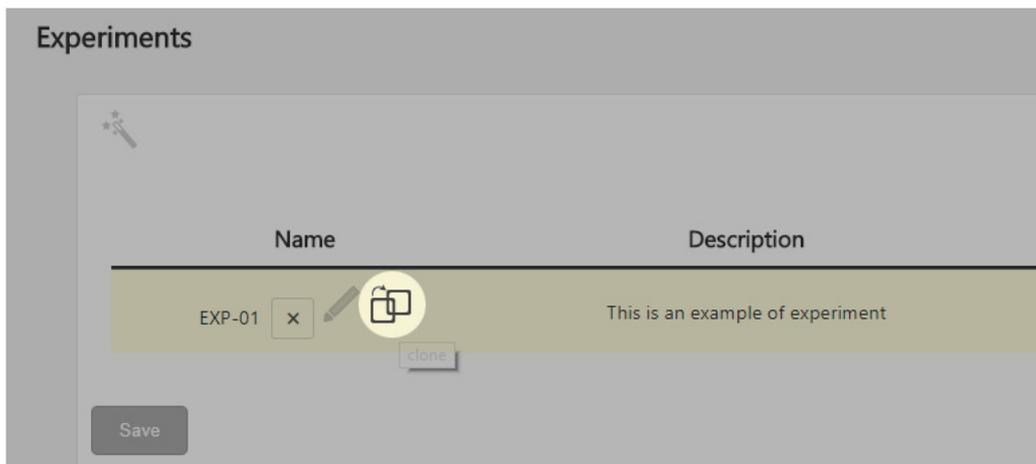


This will open the edit-project page, which contains a section to create related experiments, as shown in the image below.

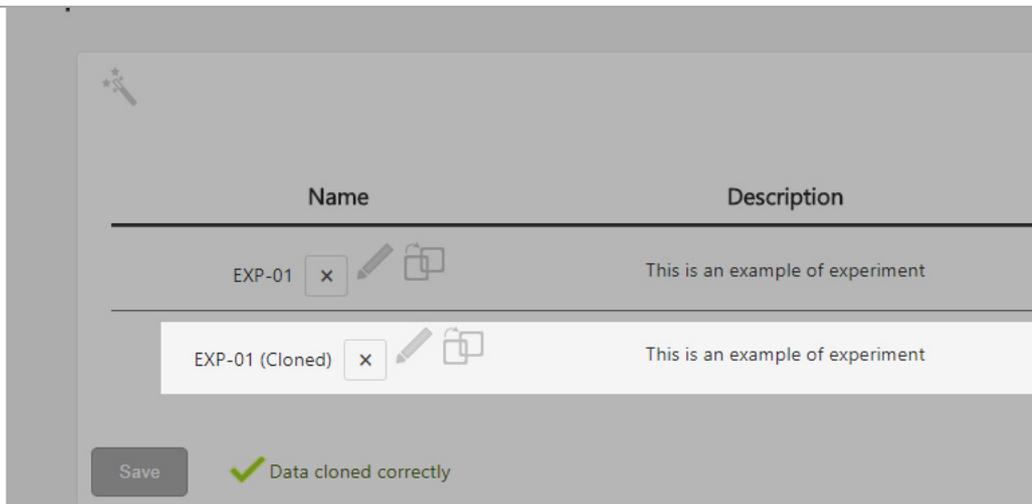


When the create experiment icon is clicked, the edit-experiment page is opened, where all the information about the experiment can be filled in by using [forms](#).

Experiments can also be created by cloning an existing experiment. This is useful when a series of experiments used the same specimen or share a similar experiment set-up. Cloning will duplicate all the experiment data and its relations, but will not duplicate the related elements, so the related elements are only "linked". If a modification is done on a related element (e.g. the name of the specimen is changed), then the modification will be displayed in all experiments using the specimen. Cloning can be done by using the clone icon as shown in the image below.



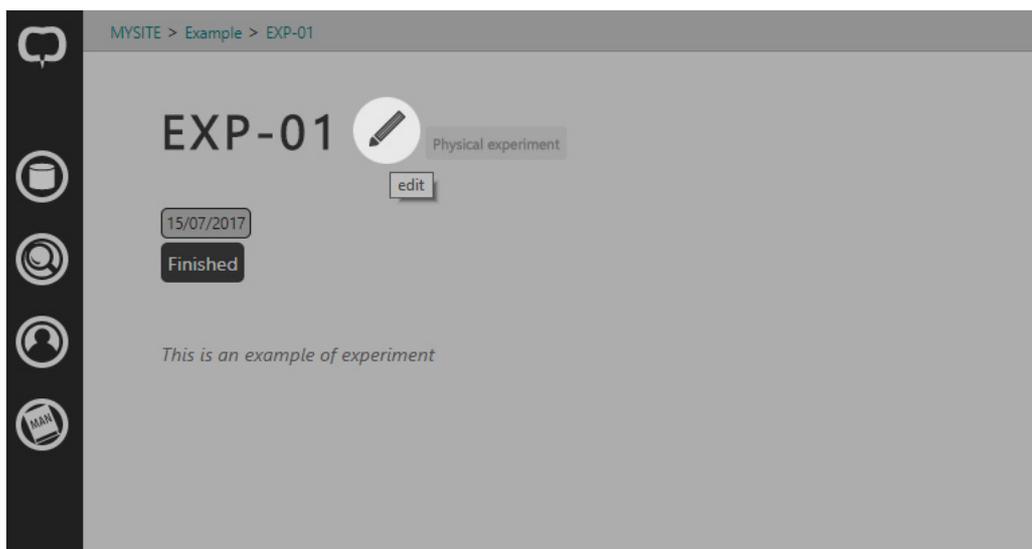
This will create a new experiment as depicted below.



7.4 Creating specimens ↑

Equally to experiments, to create specimens it is necessary to have write permissions on the project where the specimen is going to be created. Specimens are created from an existing experiment, so no specimen can be created without being linked to an experiment.

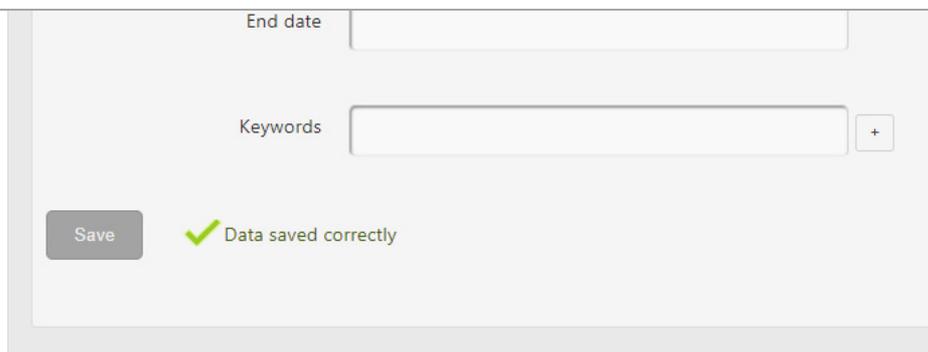
Specimens are created from the edit screen of the experiment, so the first step is to edit the experiment that will use the specimen, by using the *Edit* icon near the experiment name, as depicted below.



7.5 Working with forms ↑

Data are created and modified in CDV by using forms. Normally, a single element such as a project or a specimen will have multiple forms, which work on different blocks of information. For example, a project that is in edition mode (accessible via the ✎ icon) will show a form to save basic information (such as name, description, project dates, project keywords, *etc.*), another form to save participants in the project, another form to save project data files and so on. Each form save data independently from the other forms in the same page, and you should preferably save a form when you have made any changes before moving to the next form. Each form can be identified because forms become highlighted when the mouse is on them, and they will show a Save button.

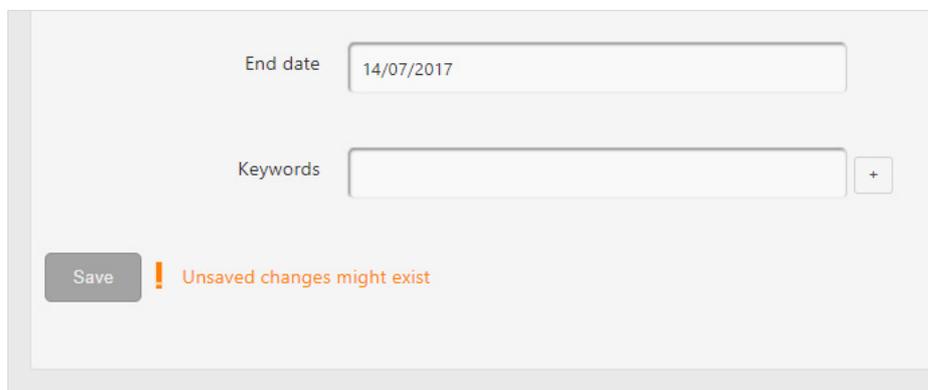
Normally, data will not be saved before the *Save* button is clicked. When this happens, you will receive confirmation of the saving process near the *Save* button. If the changes were saved correctly you should see a feedback message similar to the one in the image below.



The screenshot shows a form with two input fields: "End date" and "Keywords". Below the "Keywords" field is a small "+" button. A "Save" button is located to the left of a green checkmark and the text "Data saved correctly".

In some occasions, data will be saved automatically, normally when data were created in a emerging dialog or when files are dragged and dropped to be uploaded. Again, everytime data are saved you will receive a saving confirmation message near the *Save* button. If you need to clear the saving confirmation message, just click on it to make it disappear.

Another feedback message will appear when data have been modified but not saved yet, as shown in the image below. This is an indication that some modifications have been made but the *Save* button has not been pressed yet.



The screenshot shows the same form as above, but with the "End date" field containing the value "14/07/2017". Below the "Keywords" field, there is an orange exclamation mark icon followed by the text "Unsaved changes might exist".

Try it yourself

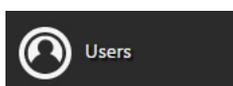
8. Tools

Tools are accesible from the main menu.

8.1 Signal Explorer

double click on a plot will hide the rest. Doble click again will take them back

9. User Management



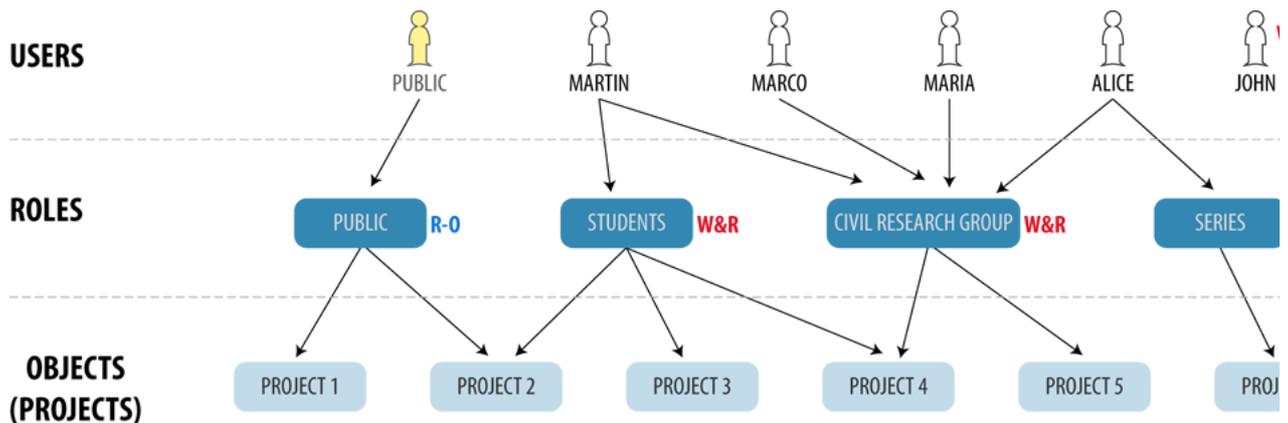
The user management option is accesible from the main menu. The main aim of this functionality is to enable data privacy by means of data access control. If all the data in a repository were public and every actor accessing the data was trusted, there would be no need for data access control. Unfortunately, this is not the case in the real world, and institutions need to control who access which data subset, and under which conditions. User management is a complex feature that involves security and data privacy. To accomplish them, CDV provides the following mechanisms: (i) A mechanism to specify which users can access which data and how. (ii) A mechanism to authenticate users, and make sure that users are who they claim to be. (iii) A mechanism to control the data access according to the authenticated user and the data access rules defined for them.

9.1 CDV data access control overview

CDV uses a flexible and scalable system to control user access to the data, and it has been designed to have a simple management while being very powerful. It is based on three elements:

- Users.** They represent the users of CDV and can be either humans or systems. Users have credentials that they provide to be authenticated by CDV to access the data, so user accounts should be used by a single actor (a single human or a single system). By default, a user has no access to any data, which has to be explicitly granted by the assignment of roles. There is a special flag *Administrator*, that grants users full access to the data and the possibility to access the User Management functionality. Since they have full access, administrators ignore any roles assigned to their users, so unless the administrator flag is assigned for intermittent periods of times, it is suggested that administrators are not explicitly associated to any role. There should be at least one administrator configured in CDV, to deal with the user management. Finally, there is a special user called **PUBLIC**, which is the user that will be assigned when no credentials are provided.
- Roles:** Roles specify what a user can do and over which data. There are two types of roles: (i) Read-Only (**RO**), which only allow to visualise data. (ii) Write & Read (**WR**), which allow read and write data. A role can be related to zero or more objects that it can access. If a role is of type RO, all objects that the role can access will be for read-only access. The special user PUBLIC is related to the role **PUBLIC** by default, which is a RO role that can be associated to all data (objects) that are publicly accessible. By default, no data is accessible by the PUBLIC role, and therefore the PUBLIC user cannot access any data by default. However, although the PUBLIC user is a special user in CDV, the PUBLIC role is a role like any other else, and therefore it can be modified, deleted, assigned to other users, etc.
- Objects (Projects),** which represent subsets of the data. In CDV, the objects that are related to the roles are the Projects, so roles grant users the access to full projects, including all their data files, experimental activities and specimens information. Since control access is granted by project data subsets instead of by smaller data subsets, control access flexibility is reduced while control access management is greatly simplified.

The following figure depicts an example of how data control access works. In the figure, the PUBLIC user has access to the PUBLIC role, which grants the user read-only access to Project1 and Project2. Note that nothing prevents the PUBLIC user from having other roles assigned, such as Students, since the management between users is uniform. The user Martin has write and read access to Project2, Project3, Project4 and Project5. He has not explicit access to Project1, although since read access to Project1 is public (inherited by the PUBLIC user) he could access it by not presenting his credentials to CDV. Normally, every non-administrator user in the system should have the PUBLIC role assigned, but it is not required (users can have different data subsets access assigned without forcing them to see the public data too).



As it can be seen, CDV data control access is powerful, simple, predictable and uniform. The only exceptional rules are the PUBLIC user (used by default when users are not authenticated) and the administrators (who can access any data and manage user accounts), but all the other rules apply consistently. Public data are managed and treated with no special rules, in the same way as any other data subset. Therefore, public users can be granted access to any object with any sort of access permission. Also, humans and systems are treated equally, so an administrator can create a user for a specific application to access a restricted subset of data, and another user for a different application to access a different subset of data (e.g. a data backup application). This flexibility enables CDV to be used in many different contexts. As an example use case, let us consider the SERIES consortium that has access to a [virtual database](#), consisting of multiple institutions sharing their own local data amongst themselves and with the general public. The different repositories that made the virtual database consist of private data (which are private for the author institution and not shared with anyone else), partner data (shared only with other SERIES institutions) and public (shared with the general public, the whole earthquake engineering community). In CDV, such data control access can be implemented by means of: (i) The creation of multiple users and roles for private internal use, as desired, some with RW access for the input of new data. (ii) The creation of a SERIES user and a SERIES role, granting it access to projects that belong to partner SERIES data. (iii) The default PUBLIC RO role, and granting it access to projects that belong to public SERIES data.

9.2 Creation of users

User creation is accessed via the main menu. In the *Users* section, click on the new element icon (). This will open the following window:

Name

Institution

Contact Email

Comments

Administrator

Roles

Lab group (READ_ONLY) PUBLIC (READ_ONLY)

Save

Input the data for the new user. Users must have at least a name. Be aware that some of these details might be exported for the user credentials and be visible by the users themselves (e.g. the user name and perhaps other details). The comments are intended to be private but if you need to put a note to remain yourself who the user was it is better not to input anything offensive or secret, because other administrators will have access to your notes.

The administrator checkbox allows you to grant administration privileges. Remember that an administrator has free access to the whole system, which includes visualising and editing any object, accessing any tool, creating projects and full user management.

The *Roles* subsection allows you to grant roles to the user, and get effectively access to the objects that the role can access. Click in the role to grant access to it. Usually, you should at least grant the user access to the PUBLIC role, if it exists, but this is not mandatory and sometimes you might want to create users that do not have access to the public subset of data. Note that administrators should not normally have any role associated, since they have already been granted access to all data. If there are no roles available, you can create the user first with no roles, [create roles](#) as necessary, and then [edit the user](#) to assign the roles.

When all details are completed, click on the *Save* button to save the user. The user will be created in the system but will not be active yet. You can see that in the user table, under the *Active* column, as depicted in the following image.

Name	Institution	Email	Comments	Administrator	Active	Roles
Ignacio Lamata Martinez	Celestina				NO	PUBLIC (READ_ONLY)

The activation of the user account should be done by the user themselves. You need to click on the *NO* word, which will open a window with the user's activation link. This link will allow the user to create their own credentials, in a way that only CDV and they are involved. You have to manually send it to the user through a *secure* channel. This process can be automated, but then some issues raise, like how to establish a secure channel (especially without the user intervention) plus the necessity of configuring new elements (like an email account). If you cannot send it through a secure channel (the email for example is not a secure channel), do not worry too much because the intrusion could be detected - the activation process can be done only once, it will not work twice for the same activation link. So if the user cannot activate their account by using the link you provided, and the user is shown as 'active' in CDV (the word *NO* is absent under the *Active* column), then someone else stole the activation link and you should delete the account immediately. A prevention system for this situation is to create the user with minimum privileges (e.g. only the PUBLIC role) and once you can confirm that the user has been activated by the right person, you can proceed to enable the rest of permissions. Note that you should encourage users to [secure their communication with ELSADATA](#) before using the activation link. It is unlikely you have to go through all this verification process, but this manual should inform you of potential risks. In many situations, administrators will just send the activation link via email to the user and only act if they receive a complain of something not working. However, note that if you store sensitive data, you should be aware of the aforementioned risks and take proper security measures.

9.3 Edition of users

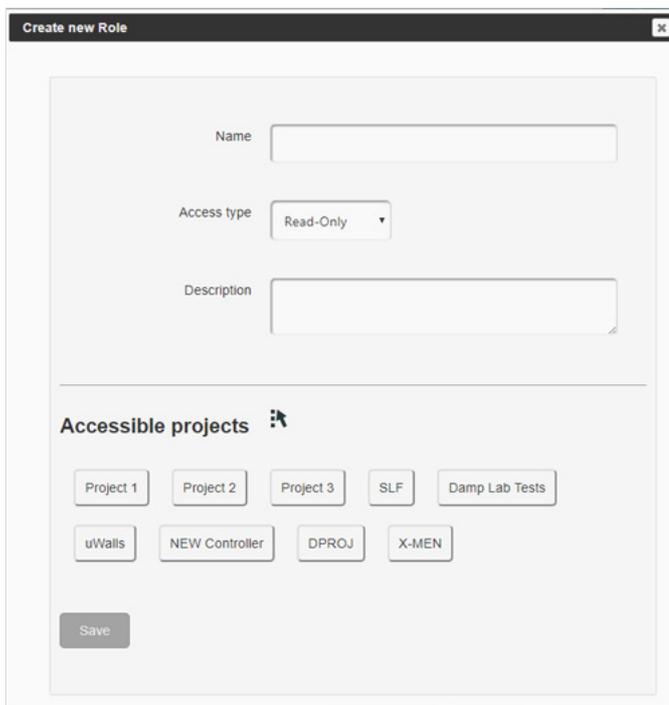
Edition of users can be done by clicking on the edit element icon () located near the user name. This will open the same window seen in the user creation. Edit user data and click on the *Save* button to make changes available.

9.4 Deletion of users

Deletion of users can be done by clicking on the delete element icon () located near the user name.

9.5 Creation of roles

Role creation is accessed via the main menu. In the *Roles* section, click on the new element icon (). This will open the following window:



Input the data for the new role. Roles must have at least a name. The default Access type is *Read-Only*, which means any project associated to this role could be only be read and not modified. The *Write & Read* Access type will allow full access, including reading, modification and deletion of data.

The *Accessible projects* subsection allows you to link the role to the different projects available. Click on the project name to link them. To select all projects at once, you can use the *Select all*  feature, which can be also used to deselect all by clicking on the icon twice.

When all details are completed, click on the *Save* button to save the role.

9.6 Edition of roles

Edition of roles can be done by clicking on the edit element icon () located near the role name. This will open the same window seen in the role creation. Edit role data and click on the *Save* button to make changes available.

9.7 Deletion of roles

Deletion of roles can be done by clicking on the delete element icon () located near the role name.

9.8 User effective permissions ↑

Verifying which permissions a user has is a matter of checking which projects are available for any of the roles that has been assigned to the user. However, this can be tedious, so the list of all permissions available for a user can be obtained by clicking on the user name, which will open a new window as shown in the following image.



In the image it can be seen that the PUBLIC user has read-only access to the projects *Example* and *uWalls*, and no write and read access to any project.

Note that every `role` and `project` element in the User Management will show a description (if it exists) when holding the mouse pointer over the element.

9.9 Project access by role ↑

There are two types of permission information that can be useful for you as an administrator. The first one is related to users: which projects a user can access, as has been covered in the previous section. The second one is related to projects: which roles have access to a specific project. In order to do this, the *Roles per project* section in the User Management can be used. This is useful to have a permission perspective from the project point of view, and to verify which roles have been granted access to them, as well as which type of access.

10. The API ↑

CDV exposes an API (Application Programming Interface) via [SOAP Web Services](#). This means that if you use a client for the Web Services, you can execute programmatically the same operations as from the Web user interface. For example, you can create projects, edit experiments or obtain specimen data via your own client program, without having to use the normal Web access. One of the advantages of this is that the data can be manipulated by a computer to computer interaction and therefore data operations can be automated and easily replicated.

The communication between a SOAP Web Service server (the Celestina Data Viewer) and its client (your own program) is governed by a software component (a document) written in a language called [WSDL](#) (Web Service Description Language). This document defines the operations exposed by the server and how to call them as if it was a software "contract". Clients can be developed in any programming language (*e.g.* Java, C, Python, Matlab scripting, *etc*) as long as they comply with this contract.

In the following Section 10.1, the operations (also called services) that can be executed on CDV are described. Section 10.2 shows a specific use of the API from a client program: how to use the API from [MathWorks MATLAB](#).

10.1 Services description ↑

The services have been grouped into colours according to the following classification:

Testing services: Useful to test different aspects of the services

Creation/modification services: Operations that create or modify data

Deletion services: Operations that remove data from the data source

For all services, their parameters, response and exceptions are listed. Parameters are the expected data you have to provide to the service as input. Response are the data that will be returned by the service as output. Exceptions are the specific errors that will be returned by the service in case something anomalous happens. Note that not all errors are listed in this section, but only the errors specific to a service. For example, any service might raise a `StoreException` if there has been a (normally, internal) problem with the data source. This is not specific of any service and affect all of them, so it will not be listed unless it is raised under a explicit circumstance for a specific service. Likewise, the general `Exception` can be raised at any service when some nonspecific error happens, and this will not be listed either.

testWS

This service is useful to test that the network connection is working between both computers client and server, that the Web Services are up and running and that a request can be sent and an answer can be received.

Parameters

None

Response

testString `String` If the service call is successful, then the string "*If you can see this, the Web Service has been called successfully*" should be received.

Exceptions

None

testWSParamException

This service is useful to test that data can be sent and receive from the client side. It is also useful to check how exceptions (errors) will be informed to the client. To test the latter, the service has to be called with an empty input parameter. In such case, the server will raise an `InvalidInputException` error, so the client knows how errors will be delivered and can prepare its code accordingly.

Parameters

○

string `String` An arbitrary string

Response

testString `String` A string that includes the input parameter

Exceptions

Invalid input `InvalidInputException` When the service is called with no parameters. It helps to check how most of the errors will be transmitted to the client

whoAml

been properly authenticated. The response of this service is PUBLIC if the client (i) is accessing the public data without any user, or (ii) the client has not been successfully authenticated and their user has been default to the PUBLIC user.

Parameters

None

Response

userName The name of the user authenticated when calling the service

Exceptions

None

getProjectsIds

This service provides all projects IDs from the data source. The project ID is a unique identifier that is useful to query specific project data with other methods, such as [getProject](#). Note that the projects IDs provided are only the ones whose projects you have read access to.

Parameters

None

Response

projectsIds A list of all projects IDs the user has access to

Exceptions

No Data When there are no available projects, either because the data source is empty or because the user does not have any read permission over any existing project

getProject

This service provides all information available for a specific project, including participants, keywords, data files, and so on. It also includes its relations with experimental activities and specimens.

Parameters

● **id** The ID of the project. The ID of a project can be obtained with methods such as [getProjectsIds](#).

Response

project The project object with all the information known about it

Exceptions

No Data When the project does not exist or the user has no permission to read it

Invalid ID When the project ID has not been passed as parameter

General When the project ID passed as parameter is erroneous (malformed)

This service provides all information available for a specific project, including participants, keywords, data files, and so on, exactly in the same way as [getProject](#). The existence of this service is justified because [getProject](#) requires an ID as input parameter. Such ID has to be previously obtained by some mean, e.g. by using [getProjectsIds](#), but this ID does not provide any hint about which project it belongs to. Normally, users know about project names but should remain unaware of project IDs, so this service provides a more human-friendly manner to query about a specific project. The service tries to find a partial match of the string provided (the project name) in the available data. The parameter is case insensitive and will return the first project that matches. It attempts to find the provided parameter within the project short names, and if it fails attempts to find it within the project names e.g. Looking for "spear" will return the first project whose shortname contains the substring "spear" case-insensitive (like "Spear ELSA"). If not found, it will look for the substring in project names.

Parameters

● **name** `String` The approximate name of the project. It is case insensitive

Response

project `Project` The project object with all the information known about it

Exceptions

No Data `NoExistingDataException` When the project cannot be located by the provided name, either because it does not exist or because the user has no permission to read it

Invalid name `InvalidInputException` When the project name has not been passed as parameter

getProjects

This service provides basic or reduced information about all accessible projects, mainly consisting of IDs, names, descriptions, dates and some other additional data. This service provides more information per project than [getProjectsIds](#) (which only provides the projects IDs), but will provide less information per project than querying a specific project with [getProject](#).

Parameters

None

Response

projects `List(Project)` A list of projects with basic information about each one

Exceptions

No Data `NoExistingDataException` When there are no existing projects or the user has no permission to read any project

getSpecimensIds

This service provides all specimens IDs from the data source. Similar to the project ID, the specimen ID is a unique identifier that is useful to query specific specimen data with other methods, such as [getSpecimen](#). Note that the specimens IDs provided are only the ones whose specimens you have read access to.

Parameters

None

Response

specimensIds `List(Strings)` A list of all specimens IDs the user has access to

Exceptions

not have any read permission over any existing specimen. Specimens are available when they belong to a project over which the user has read permissions.

getSpecimen

This service provides all information available for a specific specimen, including participants, keywords, data files, and so on. It also includes the parent project and its relations with experimental activities that use the specimen.

Parameters

- **id** The ID of the specimen. The ID of a specimen can be obtained with methods such as [getSpecimensIds](#) or from the specimen list associated to a project or an experimental activity provided by [getProject](#) or [getExperiment](#).

Response

- specimen** The specimen object with all the information known about it

Exceptions

- No Data** When the specimen does not exist or the user has no permission to read it
- Invalid ID** When the specimen ID has not been passed as parameter
- General** When the specimen ID passed as parameter is erroneous (malformed)

getExperimentsIds

This service provides all experimental activities IDs from the data source. Similar to the project ID, the experiment ID is a unique identifier that is useful to query specific specimen data with other methods, such as [getExperiment](#). Note that the experiments IDs provided are only the ones whose experiments you have read access to. Often, in this manual the term *Experiment* will be used as a generalisation for all types of experimental activities, either physical (sometimes referred as simply "experiments", "tests", etc), numerical (sometimes referred as "computations" or "simulations") or hybrid (a mixture of both physical and numerical experimental activities).

Parameters

None

Response

- experimentsIds** A list of all experiments IDs the user has access to

Exceptions

- No Data** When there are no available experiments, either because the data source is empty or because the user does not have any read permission over any existing experiment. Experiments are available when they belong to a project over which the user has read permissions.

getExperiment

This service provides all information available for a specific experiment, including participants, keywords, data files, and so on. It also includes the parent project, its relations with the specimens it uses and the signal data of the experiment.

Parameters

- **id** The ID of the experiment. The ID of a experiment can be obtained with methods such as [getExperimentsIds](#) or from the experiment list associated to a project or a specimen provided by [getProject](#) or [getSpecimen](#).

experiment `ExperimentalActivity` The experiment object with all the information known about it

Exceptions

No Data `NoExistingDataException` When the experiment does not exist or the user has no permission to read it

Invalid ID `InvalidInputException` When the experiment ID has not been passed as parameter

General `Exception` When the experiment ID passed as parameter is erroneous (malformed)

getSignalsOfExperiment

This service obtains all signals of a specific experiment. Note that the signals associated to an experiment can be also obtained via [getExperiment](#). However, [getExperiment](#) will return signals classified into *input* and *output*, whereas this service will provide all associated signals with no classification.

Parameters

● **id** `String` The ID of the experiment. The ID of a experiment can be obtained with methods such as [getExperimentsIds](#) or from the experiment list associated to a project or a specimen provided by [getProject](#) or [getSpecimen](#).

Response

signals `List(SignalData)` The list of signals associated to an experiment

Exceptions

No Data `NoExistingDataException` When the experiment does not exist, the user has no permission to read it or the experiment has no signals associated

Invalid ID `InvalidInputException` When the experiment ID has not been passed as parameter

General `Exception` When the experiment ID passed as parameter is erroneous (malformed)

getSignal

This service obtains all the information available for a specific signal. This service will be rarely used since normally signals are obtained based on a specific experiment via [getExperiment](#) or [getSignalsOfExperiment](#).

Parameters

● **id** `String` The ID of the signal. The ID of a signal can be obtained with methods such as [getExperiment](#) or [getSignalsOfExperiment](#), but they will also provide other signal data that will make the call to this service unnecessary. However, this service can prove useful in some unusual scenarios were the signal ID is known beforehand and it is not desired to query the experiment.

Response

signal `SignalData` The signal object with all the information known about it

Exceptions

No Data `NoExistingDataException` When the signal does not exist or the user has no permission to read it. Such permission is granted if the user can read the project that conducts the experiment associated with the signal

Invalid ID `InvalidInputException` When the signal ID has not been passed as parameter

General `Exception` When the signal ID passed as parameter is erroneous (malformed)

This service obtains all person information stored. It can be used to retrieve persons IDs in order to associate participants to projects, specimens, experiments and others. This service is public and every user is able to see this information.

Parameters

None

Response

persons `List(Person)` The list of persons stored in the datasource, containing all information available for each of them. Note that all the data about persons are returned, but not which elements are associated to them. To know which persons are associated to *e.g.* a project, other methods that query the specific element, such as [getProject](#), must be used

Exceptions

No Data `NoExistingDataException` When no persons exist

getAllOrganisations

This service obtains all organisation information stored. It can be used to retrieve organisations IDs in order to associate participants to projects, specimens, experiments and others. This service is public and every user is able to see this information.

Parameters

None

Response

organisations `List(Organisation)` The list of organisations stored in the datasource, containing all information available for each of them. Note that all the data about organisations are returned, but not which elements are associated to them. To know which organisations are associated to *e.g.* a project, other methods that query the specific element, such as [getProject](#), must be used

Exceptions

No Data `NoExistingDataException` When no organisations exist

getAllDataRights

This service obtains all data rights stored. It can be used to retrieve data rights IDs in order to associate data rights to projects, specimens, experiments and others. This service is public and every user is able to see this information.

Parameters

None

Response

datarights `List(DataRight)` The list of data rights stored in the datasource, containing all information available for each of them. Note that all the data about data rights are returned, but not which elements are associated to them. To know which data rights are associated to *e.g.* a project, other methods that query the specific element, such as [getProject](#), must be used

Exceptions

No Data `NoExistingDataException` When no data rights exist

This service obtains all kinds for physical properties. The list of default physical property types are: Mass, Height, Width, Length, Thickness, Radius, Ex Eccentricity, Ey Eccentricity and Ez Eccentricity, and new types can be created by using [createPhysicalPropertyType](#). Physical properties are used to describe both specimens and specimen components (which are sub-parts of specimens). A physical property type has two main data: (i) A **type** that contains the actual ID of the property type and a name (e.g. Mass), and (ii) A **unit** of the type (e.g. Kilogram). This service is public and every user is able to see this information.

Parameters

None

Response

physicalPropertyTypes The list of physical property types

Exceptions

No Data When no physical property types exist

getAllUnits

This service obtains all units stored. Units are used in different parts of the data, such as the units in which signal magnitudes are stored or the units in which specimen or specimen components physical properties are defined. This service is public and every user is able to see this information.

Parameters

None

Response

units The list of units stored in the datasource, normally containing an ID, a name e.g. kg) and a description (e.g. kilogram)

Exceptions

No Data When no units exist

getAllMaterialTypes

This service obtains all material types stored. Material types are classified into **STRUCTURAL** and **GROUND**. Material types (e.g. Glass, Steel, Sand, etc) are defined under one of these two categories. Materials are used to describe specimens and specimen components, and every material must be associated to a material type. This service is public and every user is able to see this information.

Parameters

None

Response

materialTypes The list of material types stored in the datasource. Each specific material should be defined in terms of one of these types

Exceptions

No Data When no material types exist

This service obtains all specimen component types stored. The specimen component type (e.g. 2D frame, column, beam, soil, etc) is used to define specimen components.

Parameters

None

Response

specimenComponentTypes List(Basic) The list of specimen component types

Exceptions

No Data NoExistingDataException When no specimen component types exist

getAllSignalMagnitudes

This service obtains all magnitudes available for signal data (e.g. Force, displacement, time, etc), and they have a type and a unit. This service is public and every user is able to see this information.

Parameters

None

Response

magnitudes List(Magnitude) The list of magnitudes stored in the datasource. The magnitude ID is defined inside the *type* data

Exceptions

No Data NoExistingDataException When no signal magnitudes exist

getAllDataFileRelations

This service obtains all possible relation types for data files. When an element such as a project, a specimen or an experiment has data files associated, they are related according to some relation. For example, a project can be related to a large file by means of the relation "Project summary", or a document can be related to a specimen by means of the relation "Construction documentation". There relations define in which manner a large file is associated with another object. This service provides all possible relations that can be then used to link elements and data files, by using the ID of the relation returned by this service. This service is public and every user is able to see this information.

Parameters

None

Response

dataFileRelations List(BasicWithParents) The list of data file relations stored in the datasource. Relations often define a "parent" relation, since they can be defined as a hierarchy. For example, the relation "Construction documentation" is a sub-relation of the relation "Specimen documentation". The parent of all data file relations is the generic "Data File" (see [Data files](#) for more information)

Exceptions

No Data NoExistingDataException When no organisations exist

This service returns the download address (normally, a URL) of a data file. By accessing such address, the actual data of the data file can be downloaded. For security reasons, only administrators can use this service.

Parameters

- **id** The ID of the datafile. The ID of a data file can be obtained with services such as [getProject](#), [getSpecimen](#), [getExperiment](#) or similar, which present all the datafiles associated to the project/specimen/experiment/etc. Note that normally such services will already provide the download address of the data file

Response

- downloadAddress** The address (normally, a URL) that can be used to download the actual data of the data file

Exceptions

- No Data** When no signal magnitudes exist
- Invalid ID** When the data file ID has not been passed as parameter
- No permission** When the user does not have enough permissions (*i.e.* the user is not an administrator)
- General** When the data file ID passed as parameter is erroneous (malformed)

createProject

This service creates an empty project in the data source. For security reasons, only administrators can use this service.

Parameters

None

Response

- id** The ID of the project just created

Exceptions

- No permission** When the user does not have enough permissions to create projects (*i.e.* the user is not an administrator)

saveProject

This service modifies the data of an existing project. This allows the update of current project data (*e.g.* name, description, dates, keywords, etc) as well as project relations, such as participants, data rights or the "owner" of the data. Note that this service only creates such links between the project and some other object, the actual object is created with different methods, such as [createPerson](#) for persons that can be participants, [createDataRight](#) for data rights or [createOrganisation](#) for both organisations that can be participants and organisations that can have the intellectual property rights over the project data. To unlink relationships, **CHECK THIS WITH JM.**

Similarly, any upload of files for the project is managed with different services, such as [uploadDataFileForProject](#) or [uploadIdentityImageForProject](#). Removal of project data files is managed with services such as [delDataFileForProject](#) or [delIdentityImagesForProject](#).

manner, since such relations are created from the creation service of the corresponding experiment or specimen. Therefore, created experiments and specimens can belong to only one project. Removal of this link is achieved by deleting the experiment or specimen.

Parameters

- project** The project to be modified. The project must exist and have the ID set (which could have been retrieved with methods such as [createProject](#), [getProjectsIds](#), [getProjectByName](#), and so on). Relations (e.g. *intellectualPropertyOwners*, *dataRights*, *participants*, etc) do not need to include the whole linked object, only its ID will be used. So, for example, it is only necessary the ID of the data right when adding it on the list of *dataRights*, but not other data such as the data right name or description. The ID of data rights, persons and organisations can be retrieved via [getAllDataRights](#), [getAllPersons](#) and [getAllOrganisations](#), respectively

Response

- id** The ID of the project that has been modified

Exceptions

- No Data** When the project does not exist
- Invalid ID** When the project ID has not been passed as parameter
- Invalid Input** When the user does not have write permissions over the project
- Security constrain violation on Incorrect DataFile relation** When a new data file relation has been specified, and it is not an existing/valid data file relation ID. Valid data file relation IDs can be obtained with the service [getAllDataFileRelations](#)
- Security constrain violation on non related DataFile** When a data file specified is not related to a project (meaning that the data file has not been previously upload for the project via [uploadDataFileForProject](#) or it has been already deleted). Note that this situation can occur also if either the project or the data file does not exist
- General** When the project ID passed as parameter is erroneous (malformed)

createSpecimenForProject

This service creates an empty specimen in the data source, linked to a specific project.

Parameters

- id** The ID of project under which the specimen will be created

Response

- id** The ID of the specimen just created

Exceptions

- Invalid ID** When the project ID has not been passed as parameter
- Invalid Input** When the user does not have write permissions over the project
- General** When the project ID passed as parameter is erroneous (malformed)

createSpecimenForExperiment

This service creates an empty specimen in the data source, linked to a specific experimental activity (so the experiment uses the specimen). The specimen will be also linked to the project that conducts the experimental activity.

Parameters

be created

Response

id String The ID of the specimen just created

Exceptions

Invalid ID InvalidInputException When the experiment ID has not been passed as parameter

Invalid Input InvalidInputException When the user does not have write permissions over the project that the experiment belongs to

General Exception When the experiment ID passed as parameter is erroneous (malformed)

saveSpecimen

This service modifies the data of an existing specimen. This allows the update of current specimen data (e.g.name, description, dates, keywords, etc) as well as specimen relations, such as participants, data rights or the "owner" of the data. Note that this service only creates such links between the specimen and some other object, the actual object is created with different methods, such as [createPerson](#) for persons that can be participants, [createDataRight](#) for data rights or [createOrganisation](#) for both organisations that can be participants and organisations that can have the intellectual property rights over the specimen data. To unlink relationships, **CHECK THIS WITH JM.**

Similarly, any upload of files for the specimen is managed with different services, such as [uploadDataFileForSpecimen](#) or [uploadIdentityImageForSpecimen](#). Removal of specimen data files is managed with services such as [delDataFileForSpecimen](#) or [delIdentityImagesForSpecimen](#).

The relation of the specimen with projects and experiments are not managed by this service. The link between a specimen and a project is created either by direct creation of a specimen under a project via [createSpecimenForProject](#) or by implicit creation of a specimen used by an experiment via [createSpecimenForExperiment](#), where the specimen inherits the project of the experiment. Unlike with projects, relations between specimens and experiments is quite flexible and can be created freely. A direct link between a specimen and an experiment is created via [createSpecimenForExperiment](#), but at any time relations between these two objects can be created and removed with [createExperimentSpecimenRelation](#) and [delExperimentSpecimenRelation](#) respectively.

Parameters

● **specimen** Specimen The specimen to be modified. The specimen must exist and have the ID set (which could have been retrieved with methods such as [getProject](#) that provides a list of associated specimens, or [getSpecimensIds](#)). Relations (e.g. *intellectualPropertyOwners*, *dataRights*, *participants*, etc) do not need to include the whole linked object, only its ID will be used. So, for example, it is only necessary the ID of the data right when adding it on the list of *dataRights*, but not other data such as the data right name or description. The ID of data rights, persons and organisations can be retrieved via [getAllDataRights](#), [getAllPersons](#) and [getAllOrganisations](#), respectively

Response

id String The ID of the specimen that has been modified

Exceptions

No Data NoExistingDataException When the specimen does not exist

Invalid ID InvalidInputException When the specimen ID has not been passed as parameter

Invalid Input InvalidInputException When the user does not have write permissions over the specimen

Security constrain violation on Incorrect DataFile relation InvalidInputException When a new data file relation has been specified, and it is not an existing/valid data file relation ID. Valid data file relation IDs can be obtained with the service [getAllDataFileRelations](#)

Security constrain violation on non related DataFile InvalidInputException When a data file specified is not related to a specimen (meaning that the data file has not been previously upload for the specimen via [uploadDataFileForSpecimen](#) or it has been already deleted). Note that this situation can occur also if either the specimen or the data file does not exist

General Exception When the specimen ID passed as parameter is erroneous (malformed)

This service creates an empty physical experimental activity in the data source, linked to a specific project.

Parameters

● **id** The ID of project under which the experiment will be created

Response

id The ID of the physical experimental activity just created

Exceptions

Invalid ID When the project ID has not been passed as parameter

Invalid Input When the user does not have write permissions over the project

General When the project ID passed as parameter is erroneous (malformed)

createNumericalExperimentForProject

This service creates an empty numerical experimental activity in the data source, linked to a specific project.

Parameters

● **id** The ID of project under which the experiment will be created

Response

id The ID of the numerical experimental activity just created

Exceptions

Invalid ID When the project ID has not been passed as parameter

Invalid Input When the user does not have write permissions over the project

General When the project ID passed as parameter is erroneous (malformed)

createExperimentForSpecimen

This service creates an empty physical experimental activity in the data source, linked to a specific specimen (so the experiment uses the specimen). The experiment will be also linked to the project that the specimen belongs to.

Parameters

● **id** The ID of the specimen that the physical experiment uses, and which will provide the project context to the experiment that will be created

Response

id The ID of the physical experimental activity just created

Exceptions

Invalid ID When the specimen ID has not been passed as parameter

General Exception When the specimen ID passed as parameter is erroneous (malformed)

createNumericalExperimentForSpecimen

This service creates an empty numerical experimental activity in the data source, linked to a specific specimen (so the numerical experiment uses the specimen). The numerical experiment will be also linked to the project that the specimen belongs to.

Parameters

● **id** String The ID of the specimen that the numerical experiment uses, and which will provide the project context to the numerical experiment that will be created

Response

id String The ID of the numerical experimental activity just created

Exceptions

Invalid ID InvalidInputException When the specimen ID has not been passed as parameter

Invalid Input InvalidInputException When the user does not have write permissions over the project that the specimen belongs to

General Exception When the specimen ID passed as parameter is erroneous (malformed)

saveExperiment

This service modifies the data of an existing experiment. This allows the update of current experiment data (e.g.name, description, dates, keywords, etc) as well as experiment relations, such as participants, data rights or the "owner" of the data. Note that this service only creates such links between the experiment and some other object, the actual object is created with different methods, such as [createPerson](#) for persons that can be participants, [createDataRight](#) for data rights or [createOrganisation](#) for both organisations that can be participants and organisations that can have the intellectual property rights over the specimen data. To unlink relationships, **CHECK THIS WITH JM.**

Similarly, any upload of files for the experiment is managed with different services, such as [uploadDataFileForExperimentalActivity](#) or [uploadIdentityImageForExperiment](#). Removal of experiment data files is managed with services such as [delDataFileForExperiment](#) or [delIdentityImagesForExperiment](#).

Signals of an experiment can be created with the services [uploadInputSignals](#) or [uploadOutputSignals](#). The upload of signals uses a file to upload signals in bulk. To delete signals, however, the service [delSignal](#) must be used.

The relation of the experiment with projects and specimens are not managed by this service. The link between an experiment and a project is created either by direct creation of an experiment under a project via [createExperimentForProject](#) and [createNumericalExperimentForProject](#) or by implicit creation of an experiment that uses a specimen via [createExperimentForSpecimen](#) and [createNumericalExperimentForSpecimen](#), where the experiment inherits the project of the specimen. Unlike with projects, relations between specimens and experiments is quite flexible and can be created freely. A direct link between a specimen and an experiment is created via [createExperimentForSpecimen](#) and [createNumericalExperimentForSpecimen](#), but at any time relations between these two objects can be created and removed with [createExperimentSpecimenRelation](#) and [delExperimentSpecimenRelation](#) respectively.

Parameters

● **experiment** Experiment The experiment to be modified. The experiment must exist and have the ID set (which could have been retrieved with methods such as [getProject](#) that provides a list of associated experiments, or [getExperimentsIds](#)). Relations (e.g. *intellectualPropertyOwners*, *dataRights*, *participants*, etc) do not need to include the whole linked object, only its ID will be used. So, for example, it is only necessary the ID of the data right when adding it on the list of *dataRights*, but not other data such as the data right name or description. The ID of data rights, persons and organisations can be retrieved via [getAllDataRights](#), [getAllPersons](#) and [getAllOrganisations](#), respectively

Response

Exceptions

- No Data** `NoExistingDataException` When the experiment does not exist

- Invalid ID** `InvalidInputException` When the experiment ID has not been passed as parameter

- Invalid Input** `InvalidInputException` When the user does not have write permissions over the experiment

- Security constrain violation on Incorrect DataFile relation** `InvalidInputException` When a new data file relation has been specified, and it is not an existing/valid data file relation ID. Valid data file relation IDs can be obtained with the service [getAllDataFileRelations](#)

- Security constrain violation on non related DataFile** `InvalidInputException` When a data file specified is not related to an experiment (meaning that the data file has not been previously upload for the experiment via [uploadDataFileForExperimentalActivity](#), or it has been already deleted). Note that this situation can occur also if either the experiment or the data file does not exist

- General** `Exception` When the experiment ID passed as parameter is erroneous (malformed)

createExperimentSpecimenRelation

This service links a specimen and an experimental activity by creating a relation between them. Both the specimen and the experimental activity must belong to the same project to be able to establish the relation. To remove the created relation, the service [delExperimentSpecimenRelation](#) can be used.

Parameters

- **experiment** `Experiment` The experimental activity in the relation. Note that only the ID of the experimental activity is required, the rest of the experiment data can be empty

- **specimen** `Specimen` The specimen in the relation. Note that only the ID of the specimen is required, the rest of the specimen data can be empty

Response

- result** `Boolean` A symbolic response meaning that no error was raised. The service will always return `true` unless an exception happens

Exceptions

- Invalid ID** `InvalidInputException` When either the experiment ID or the specimen ID has not been passed as parameter

- Different projects** `InvalidInputException` When the experiment and the specimen belong to different projects

- Invalid Input** `InvalidInputException` When the user does not have write permissions over the project that both the experiment and the specimen belong to

- General** `Exception` When either the experiment ID or the specimen ID passed as parameter is erroneous (malformed)

createPerson

This service creates a person in the data source. Persons are normally used as participants of projects, experiments and specimens. For security reasons, only administrators can use this service.

Parameters

- **forename** `String` The forename of the person

- **familyName** `String` The family name of the person

Response

Exceptions

Invalid parameter `InvalidInputException` When either the forname or the family name is empty

No permission `InvalidInputException` When the user does not have enough permissions (*i.e.* the user is not an administrator)

savePerson

This service modifies the data of an existing person. The person must have been previously created by means of [createPerson](#). This service only creates persons. To link them as participants to projects, experiments or specimens, the services [saveProject](#), [saveExperiment](#), or [saveSpecimen](#) must be used, respectively. For security reasons, only administrators can use this service.

Parameters

● **person** `Person` The person to be modified, which must exist and have the ID set (which could have been retrieved with methods such as [getAllPersons](#) or from [createPerson](#)).

Response

id `String` The ID of the person that has been modified

Exceptions

No Data `NoExistingDataException` When the person does not exist

Invalid ID `InvalidInputException` When the person ID has not been set

No permission `InvalidInputException` When the user does not have enough permissions (*i.e.* the user is not an administrator)

General `Exception` When the person ID set is erroneous

createOrganisation

This service creates an organisation in the data source. Organisations are mainly used as participants of projects, experiments and specimens, and as "owners" or intellectual property holders of some data. For security reasons, only administrators can use this service.

Parameters

● **name** `String` The name of the organisation

Response

id `String` The ID of the organisation just created

Exceptions

Invalid parameter `InvalidInputException` When either the name is empty

No permission `InvalidInputException` When the user does not have enough permissions (*i.e.* the user is not an administrator)

saveOrganisation

This service modifies the data of an existing organisation. The organisation must have been previously created by means of [createOrganisation](#). This service only creates organisations. To link them as participants or data owners to projects, experiments or specimens,

service.

Parameters

- **organisation** The organisation to be modified, which must exist and have the ID set (which could have been retrieved with methods such as [getAllOrganisations](#) or from [createOrganisation](#)).

Response

- id** The ID of the organisation that has been modified

Exceptions

- No Data** When the organisation does not exist
- Invalid ID** When the organisation ID has not been set
- No permission** When the user does not have enough permissions (*i.e.* the user is not an administrator)
- General** When the organisation ID set is erroneous

createDataRight

This service creates a "data right" in the data source, which defines a set of a document, a text and/or a reference that state how some dataset can be used, and any other considerations to take into account. Herein, data rights are flexible data structures that can be used for example to define "data policies", assert about "data quality/data curation", "data legal terms", etc. Data rights are normally specified by project, but can be specified for experiments and specimens too. For security reasons, only administrators can use this service.

Parameters

- **dataright** The data right information. All data (*name*, *description* and *url*) are optional, and any combination of the three can be used. The description can be used to provide the full text of the data right, or to provide a summary that is further complemented with a URL (*e.g.* pointing to an external Web page or a document to be downloaded). The ID is ignored since a new one will be created and assigned

Response

- id** The ID of the data right just created

Exceptions

- Invalid parameter** When the data right itself is empty. Note that the data inside (*name*, *description* and *url*) can be empty
- No permission** When the user does not have enough permissions (*i.e.* the user is not an administrator)

saveDataRight

This service modifies the data of an existing data right. The data right must have been previously created by means of [createDataRight](#). This service only creates data rights. To link them to projects, experiments or specimens, the services [saveProject](#), [saveExperiment](#), or [saveSpecimen](#) must be used, respectively. For security reasons, only administrators can use this service.

Parameters

- **dataright** The data right to be modified, which must exist and have the ID set (which could have been retrieved with methods such as [getAllDataRights](#) or from [createDataRight](#)).

Response

Exceptions

- No Data** `NoExistingDataException` When the data right does not exist
- Invalid ID** `InvalidInputException` When the data right ID has not been set
- No permission** `InvalidInputException` When the user does not have enough permissions (*i.e.* the user is not an administrator)
- General** `Exception` When the data right ID set is erroneous

createPhysicalPropertyType

This service creates a new kind of physical property. This enables users to expand the list of default physical property types, which are: Mass, Height, Width, Length, Thickness, Radius, Ex Eccentricity, Ey Eccentricity and Ez Eccentricity (the full list can be obtained at any time with [getAllPhysicalPropertyTypes](#)). Physical properties are used to describe both specimens and specimen components (which are sub-parts of specimens), and they can be set by using [saveSpecimen](#). For security reasons, only administrators can use this service.

Parameters

- **physicalPropertyType** `TypeAndUnit` The physical property type information. This type contains a **type**, which must contain a name and, optionally, a description, and a **unit**, which is optional. If the unit is not specified, a "dimensionless unit" will be set for the physical property type. If the unit is specified, it must contain an existing unit ID (other unit data will be ignored). The unit ID of the desired unit can be obtained by means of [getAllUnits](#)

Response

- id** `String` The ID of the physical property type just created

Exceptions

- Invalid parameter** `InvalidInputException` When the physical property type itself is empty or the type name is empty
- No permission** `InvalidInputException` When the user does not have enough permissions (*i.e.* the user is not an administrator)

createMaterialType

This service creates a new kind of material. This enables users to expand the list of default material types, which are: Glass, Steel, Sand, Concrete, etc (the full list can be obtained with [getAllMaterialTypes](#)). Material types must be classified under the category **STRUCTURAL** material type or **GROUND** material type. The material type is used when creating materials for specimens or specimen components by means of [saveSpecimen](#). For security reasons, only administrators can use this service.

Parameters

- **materialType** `MaterialType` The material type information. This type must contain a name and a category (that can be either **STRUCTURAL** or **GROUND**), as well as an optional description

Response

- id** `String` The ID of the material type just created

Exceptions

- Invalid parameter** `InvalidInputException` When any of the material type itself, the name or the category is empty
- No permission** `InvalidInputException` When the user does not have enough permissions (*i.e.* the user is not an administrator)

This service creates a new relation that can be used between any element that contains large files (e.g. projects, experiments, specimens, etc) and the large file itself. There exist a predefined list of data file relations, such as **Deliverable** to indicate that the file is a deliverable for some element, **Construction documentation** to indicate that the file is documenting the construction process of a specimen, **Publication** to indicate that the file has been submitted to a journal or a conference, and many others. This service enables users to expand such list of relations. For security reasons, only administrators can use this service.

Parameters

- **property** Basic The new relation to be created. The relation must contain a representative name and a description explaining in which situation the relation can be used
- **parentPropertyId** String The ID of an existing relation that will be used as parent in the relation hierarchy. The IDs of all existing relations can be obtained via [getAllDataFileRelations](#). If this parameter is not set, the parent root of all relations (the generic relation **Data File**) will be used

Response

- id** String The ID of the data file relation just created

Exceptions

- Invalid parameter** InvalidInputException When any of the property itself, the name or the description is empty
- No permission** InvalidInputException When the user does not have enough permissions (i.e. the user is not an administrator)

uploadDataFileForProject

This service uploads a data file for a project, under a specific relation. The large file is "converted" into a Data File inside the system. Once a data file is uploaded, it can be deleted via [delDataFileForProject](#). After uploading, the existing relation between a data file and the project can be modified by using [saveProject](#).

Parameters

- **upFile** UploadFile A self contained object with all the information to upload a file. It contains the following data: (i) **relatedId** (required), which specifies the project ID that will contain the data file, (ii) **relatedRelation** (required), which contains the relation ID to specify how the project and the large file are related. The relation ID desired can be obtained via [getAllDataFileRelations](#), (iii) **filename** (optional), which is a descriptive name for the file - if not set, an arbitrary one will be used, (iv) **description** (optional), an optional description for the large file, (v) **lastModifiedDate** (optional), the date to be registered for the large file - if none is set, the current date will be used, (vi) **data** (required), the actual large file data

Response

- id** String The ID of the data file just created

Exceptions

- Invalid parameter** InvalidInputException When the related ID is empty or the data is empty
- No permission** InvalidInputException When the related project does not exist or the user does not have write permissions over it
- Security constraint violation** InvalidInputException When the relation specified is not an existing/valid data file relation ID. Valid data file relation IDs can be obtained with the service [getAllDataFileRelations](#)
- Wrong date format** InvalidInputException When the "last modified date" is incorrect. The date must be specified in the format *yyyy-MM-dd*
- Drive error** IOException When a drive error has occurred while processing the upload of the large file
- General** Exception When the related ID passed as parameter is erroneous (malformed)

This service uploads a data file for a specimen, under a specific relation. The large file is "converted" into a Data File inside the system. Once a data file is uploaded, it can be deleted via [delDataFileForSpecimen](#). After uploading, the existing relation between a data file and the specimen can be modified by using [saveSpecimen](#).

Parameters

● **upFile** A self contained object with all the information to upload a file. It contains the following data: (i) **relatedId** (required), which specifies the specimen ID that will contain the data file, (ii) **relatedRelation** (required), which contains the relation ID to specify how the specimen and the large file are related. The relation ID desired can be obtained via [getAllDataFileRelations](#), (iii) **filename** (optional), which is a descriptive name for the file - if not set, an arbitrary one will be used, (iv) **description** (optional), an optional description for the large file, (v) **lastModifiedDate** (optional), the date to be registered for the large file - if none is set, the current date will be used, (vi) **data** (required), the actual large file data

Response

id The ID of the data file just created

Exceptions

Invalid parameter When the related ID is empty or the data is empty

No permission When the related specimen does not exist or the user does not have write permissions over it

Security constraint violation When the relation specified is not an existing/valid data file relation ID. Valid data file relation IDs can be obtained with the service [getAllDataFileRelations](#)

Wrong date format When the "last modified date" is incorrect. The date must be specified in the format *yyyy-MM-dd*

Drive error When a drive error has occurred while processing the upload of the large file

General When the related ID passed as parameter is erroneous (malformed)

uploadDataFileForExperimentalActivity

This service uploads a data file for an experiment, under a specific relation. The large file is "converted" into a Data File inside the system. Once a data file is uploaded, it can be deleted via [delDataFileForExperiment](#). After uploading, the existing relation between a data file and the experiment can be modified by using [saveExperiment](#).

Parameters

● **upFile** A self contained object with all the information to upload a file. It contains the following data: (i) **relatedId** (required), which specifies the experiment ID that will contain the data file, (ii) **relatedRelation** (required), which contains the relation ID to specify how the experiment and the large file are related. The relation ID desired can be obtained via [getAllDataFileRelations](#), (iii) **filename** (optional), which is a descriptive name for the file - if not set, an arbitrary one will be used, (iv) **description** (optional), an optional description for the large file, (v) **lastModifiedDate** (optional), the date to be registered for the large file - if none is set, the current date will be used, (vi) **data** (required), the actual large file data

Response

id The ID of the data file just created

Exceptions

Invalid parameter When the related ID is empty or the data is empty

No permission When the related experiment does not exist or the user does not have write permissions over it

Security constraint violation When the relation specified is not an existing/valid data file relation ID. Valid data file relation IDs can be obtained with the service [getAllDataFileRelations](#)

Wrong date format When the "last modified date" is incorrect. The date must be specified in the format *yyyy-MM-dd*

General Exception When the related ID passed as parameter is erroneous (malformed)

uploadIdentityImageForProject

This service updates the identity image (which can be considered as a logo or an icon) of a project. Any existing identity images for the project will be removed before the new image is uploaded. Large images are resized automatically. Valid image formats are **PNG, JPG, BMP, GIF** and **TIFF** (in RGB, but not in CMYK format). Once the identity image is uploaded, it can be removed via [delIdentityImagesForProject](#) or updated via this method.

Parameters

● **upFile** UploadFile A self contained object with all the information to upload a file. It contains the following data: (i) **relatedId** (required), which specifies the project ID for which the identity image will be set, and (ii) **data** (required), the actual image data. The parameters filename, description, lastModifiedDate and relatedRelation are not relevant and must be left empty

Response

downloadIri String The IRI (normally, a URL) to download the identity image data just created

Exceptions

Invalid parameter InvalidInputException When the related ID is empty or the data is empty

No permission InvalidInputException When the related project does not exist or the user does not have write permissions over it

Invalid image InvalidInputException When the image is not recognised as such, because it has an incorrect or unsupported format

Wrong date format InvalidInputException When the "last modified date" is incorrectly set. The date should be left empty or must be specified in the format *yyyy-MM-dd*

Drive error IOException When a drive error has occurred while processing the upload of the image

General Exception When the related ID passed as parameter is erroneous (malformed)

uploadIdentityImageForSpecimen

This service updates the identity image (which can be considered as a logo or an icon) of a specimen. Any existing identity images for the specimen will be removed before the new image is uploaded. Large images are resized automatically. Valid image formats are **PNG, JPG, BMP, GIF** and **TIFF** (in RGB, but not in CMYK format). Once the identity image is uploaded, it can be removed via [delIdentityImagesForSpecimen](#) or updated via this method.

Parameters

● **upFile** UploadFile A self contained object with all the information to upload a file. It contains the following data: (i) **relatedId** (required), which specifies the specimen ID for which the identity image will be set, and (ii) **data** (required), the actual image data. The parameters filename, description, lastModifiedDate and relatedRelation are not relevant and must be left empty

Response

downloadIri String The IRI (normally, a URL) to download the identity image data just created

Exceptions

Invalid parameter InvalidInputException When the related ID is empty or the data is empty

No permission InvalidInputException When the related specimen does not exist or the user does not have write permissions over it

Invalid image InvalidInputException When the image is not recognised as such, because it has an incorrect or unsupported format

specified in the format *yyyy-MM-dd*

Drive error When a drive error has occurred while processing the upload of the image

General When the related ID passed as parameter is erroneous (malformed)

uploadIdentityImageForExperiment

This service updates the identity image (which can be considered as a logo or an icon) of an experiment. Any existing identity images for the experiment will be removed before the new image is uploaded. Large images are resized automatically. Valid image formats are **PNG, JPG, BMP, GIF** and **TIFF** (in RGB, but not in CMYK format). Once the identity image is uploaded, it can be removed via [delIdentityImagesForExperiment](#) or updated via this method.

Parameters

● **upFile** A self contained object with all the information to upload a file. It contains the following data: (i) **relatedId** (required), which specifies the experiment ID for which the identity image will be set, and (ii) **data** (required), the actual image data. The parameters filename, description, lastModifiedDate and relatedRelation are not relevant and must be left empty

Response

downloadIri The IRI (normally, a URL) to download the identity image data just created

Exceptions

Invalid parameter When the related ID is empty or the data is empty

No permission When the related experiment does not exist or the user does not have write permissions over it

Invalid image When the image is not recognised as such, because it has an incorrect or unsupported format

Wrong date format When the "last modified date" is incorrectly set. The date should be left empty or must be specified in the format *yyyy-MM-dd*

Drive error When a drive error has occurred while processing the upload of the image

General When the related ID passed as parameter is erroneous (malformed)

uploadIdentityImageForPerson

This service updates the identity image (similar to a profile picture) of a person. Any existing identity images for the person will be removed before the new image is uploaded. Large images are resized automatically. Valid image formats are **PNG, JPG, BMP, GIF** and **TIFF** (in RGB, but not in CMYK format). Once the identity image is uploaded, it can be removed via [delIdentityImagesForPerson](#) or updated via this method. For security reasons, only administrators can use this service.

Parameters

● **upFile** A self contained object with all the information to upload a file. It contains the following data: (i) **relatedId** (required), which specifies the person ID for which the identity image will be set, and (ii) **data** (required), the actual image data. The parameters filename, description, lastModifiedDate and relatedRelation are not relevant and must be left empty

Response

downloadIri The IRI (normally, a URL) to download the identity image data just created

Exceptions

Invalid parameter When the related ID is empty or the data is empty

No permission When the user does not have enough permissions (*i.e.* the user is not an administrator)

Wrong date format `InvalidInputException` When the "last modified date" is incorrectly set. The date should be left empty or must be specified in the format `yyyy-MM-dd`

Drive error `IOException` When a drive error has occurred while processing the upload of the image

General `Exception` When the related ID passed as parameter is erroneous (malformed)

uploadIdentityImageForOrganisation

This service updates the identity image (normally, the organisation logo) of an organisation. Any existing identity images for the organisation will be removed before the new image is uploaded. Large images are resized automatically. Valid image formats are **PNG, JPG, BMP, GIF** and **TIFF** (in RGB, but not in CMYK format). Once the identity image is uploaded, it can be removed via [delIdentityImagesForOrganisation](#) or updated via this method. For security reasons, only administrators can use this service.

Parameters

● **uploadFile** `UploadFile` A self contained object with all the information to upload a file. It contains the following data: (i) **relatedId** (required), which specifies the organisation ID for which the identity image will be set, and (ii) **data** (required), the actual image data. The parameters filename, description, lastModifiedDate and relatedRelation are not relevant and must be left empty

Response

downloadIri `String` The IRI (normally, a URL) to download the identity image data just created

Exceptions

Invalid parameter `InvalidInputException` When the related ID is empty or the data is empty

No permission `InvalidInputException` When the user does not have enough permissions (*i.e.* the user is not an administrator)

Invalid image `InvalidInputException` When the image is not recognised as such, because it has an incorrect or unsupported format

Wrong date format `InvalidInputException` When the "last modified date" is incorrectly set. The date should be left empty or must be specified in the format `yyyy-MM-dd`

Drive error `IOException` When a drive error has occurred while processing the upload of the image

General `Exception` When the related ID passed as parameter is erroneous (malformed)

uploadInputSignals

This service uploads a file with signal data, parses it and converts it into different Signal Data inside the system, associated to an experiment. The current format supported for the file with signal data is explained in the Section [Creating Experiments](#). A template of the file [can be downloaded](#).

This service associate signals as **INPUT** signals used in an experiment, meaning the signals that were fed to the experiment prior or during the execution of the experiment to be actually used to conduct the experiment. Signals that came as result of the experiment are uploaded as OUTPUT signals by using [uploadOutputSignals](#).

There is an initial parsing of the file. Once this is done, signals will be uploaded one by one sequentially. If an error happens, the process will be stopped and the user will be informed. This has to be taken into consideration since some signal data might have been correctly uploaded already. For the system, signals are not identified by a name, but by a unique ID that is assigned at uploading time, so uploading the exact same file (with the fixed errors) might result in some duplication of the signals that were correctly uploaded. Signals that have been uploaded can be retrieved via [getExperiment](#) or [getSignalsOfExperiment](#).

Parameters

● **experimentId** `String` The ID of the experiment that will contain the signal data

with all the information for one or more signals

Response

id The IDs of every signal that has just been created

Exceptions

Invalid ID When the experiment ID is empty or the data is empty

No permission When the experiment does not exist or the user does not have write permissions over it

Invalid header When the signal data file cannot be correctly parsed because a possible error in the supplied format

Number error When some value of a signal cannot be recognised as a valid number

Invalid magnitude When the magnitude specified for some signal data is not recognised. Valid magnitudes names can be obtained via [getAllSignalMagnitudes](#). Note that the name, and not the ID, must be used

Drive error When a drive error has occurred while processing the upload of the signal data

No signals found When no signal data has been identified in the file

Unsupported format When the signal file is unsupported. A CSV file as explained above is the recommended method to upload signal data

General When the related ID passed as parameter is erroneous (malformed)

uploadOutputSignals

This service uploads a file with signal data, parses it and converts it into different Signal Data inside the system, associated to an experiment. The current format supported for the file with signal data is explained in the Section [Creating Experiments](#). A template of the file [can be downloaded](#).

This service associate signals as **OUTPUT** signals used in an experiment, meaning the signals that were obtained as result of the experiment execution. Signals that were used for the execution of the experiment are uploaded as INPUT signals by using [uploadInputSignals](#).

There is an initial parsing of the file. Once this is done, signals will be uploaded one by one sequentially. If an error happens, the process will be stopped and the user will be informed. This has to be taken into consideration since some signal data might have been correctly uploaded already. For the system, signals are not identified by a name, but by a unique ID that is assigned at uploading time, so uploading the exact same file (with the fixed errors) might result in some duplication of the signals that were correctly uploaded. Signals that have been uploaded can be retrieved via [getExperiment](#) or [getSignalsOfExperiment](#).

Parameters

● **experimentId** The ID of the experiment that will contain the signal data

● **dataBytes** The actual data of the large file containing the signal data. This is normally a CSV (Comma Separated Value) file with all the information for one or more signals

Response

id The IDs of every signal that has just been created

Exceptions

Invalid ID When the experiment ID is empty or the data is empty

No permission When the experiment does not exist or the user does not have write permissions over it

Invalid header When the signal data file cannot be correctly parsed because a possible error in the supplied format

Number error When some value of a signal cannot be recognised as a valid number

be obtained via [getAllSignalMagnitudes](#). Note that the name, and not the ID, must be used

Drive error `IOException` When a drive error has occurred while processing the upload of the signal data

No signals found `InvalidInputException` When no signal data has been identified in the file

Unsupported format `NotImplementedException` When the signal file is unsupported. A CSV file as explained above is the recommended method to upload signal data

General `Exception` When the related ID passed as parameter is erroneous (malformed)

delProject

This service deletes a project from the data source, including all its dependent specimens, experiments, data files, and so on. Elements that are simply linked to the project, such as data rights, persons and organisations, are not affected. The operation is not atomic, so if an error happens in the middle of the deletion process, it is possible that the project is only partially deleted.

Parameters

● **project** `Project` The project to be deleted. Only the project ID is required

Response

id `String` The ID of the project just deleted

Exceptions

Invalid parameter `InvalidInputException` When the project ID is empty or the data is empty

No data `NoExistingDataException` When the project does not exist

No permission `InvalidInputException` When the user does not have write permissions over the project or it does not exist (for non-admin users)

Drive error `IOException` When a drive error has occurred while deleting some physical large file

General `Exception` When the project ID passed as parameter is erroneous (malformed)

delSpecimen

This service deletes a specimen from the data source, including all its dependent materials, specimen components, data files, and so on. Elements that are simply linked to the specimen, such as data rights, persons and organisations, are not affected. The operation is not atomic, so if an error happens in the middle of the deletion process, it is possible that the specimen is only partially deleted.

Parameters

● **specimen** `Specimen` The specimen to be deleted. Only the specimen ID is required

Response

id `String` The ID of the specimen just deleted

Exceptions

No data `NoExistingDataException` When the specimen does not exist

No permission `InvalidInputException` When the user does not have write permissions over the specimen or it does not exist (for non-admin users)

Drive error `IOException` When a drive error has occurred while deleting some physical large file

General `Exception` When the specimen ID passed as parameter is erroneous (malformed)

delExperiment

This service deletes an experiment from the data source, including all its dependent signal data, data files, and so on. Elements that are simply linked to the experiment, such as devices, data rights, persons and organisations, are not affected. The operation is not atomic, so if an error happens in the middle of the deletion process, it is possible that the experiment is only partially deleted.

Parameters

● **experiment** `ExperimentalActivity` The experiment to be deleted. Only the experiment ID is required

Response

id `String` The ID of the experiment just deleted

Exceptions

Invalid parameter `InvalidInputException` When the experiment ID is empty or the data is empty

No data `NoExistingDataException` When the experiment does not exist

No permission `InvalidInputException` When the user does not have write permissions over the experiment or it does not exist (for non-admin users)

Drive error `IOException` When a drive error has occurred while deleting some physical large file

General `Exception` When the experiment ID passed as parameter is erroneous (malformed)

delExperimentSpecimenRelation

This service deletes the link between a specimen and an experimental activity by removing a relation between them. To create again a relation, the service [createExperimentSpecimenRelation](#) can be used.

Parameters

● **experiment** `Experiment` The experimental activity in the relation. Note that only the ID of the experimental activity is required, the rest of the experiment data can be empty

● **specimen** `Specimen` The specimen in the relation. Note that only the ID of the specimen is required, the rest of the specimen data can be empty

Response

result `Boolean` A symbolic response meaning that no error was raised. The service will always return `true` unless an exception happens

Exceptions

Invalid ID `InvalidInputException` When either the experiment ID or the specimen ID has not been passed as parameter

Different projects `InvalidInputException` When the experiment and the specimen belong to different projects (note that, although the service [createExperimentSpecimenRelation](#) would have refused to create a relation in such circumstances, the check is also performed in this service)

specimen belong to

General When either the experiment ID or the specimen ID passed as parameter is erroneous (malformed)

delSignal

This service deletes a signal and its data from the data source.

Parameters

● **signal** The signal to be deleted. Only the signal ID is required

Response

id The ID of the signal just deleted

Exceptions

Invalid parameter When the signal ID is empty or the data is empty

No data When the signal does not exist or it does not belong to an experiment

No permission When the user does not have write permissions over the experiment that is related to the signal

Drive error When a drive error has occurred while deleting some physical signal data

General When the signal ID passed as parameter is erroneous (malformed)

delPerson

This service deletes a person from the data source. For security reasons, only administrators can use this service.

Parameters

● **person** The person to be deleted. Only the person ID is required

Response

id The ID of the person just deleted

Exceptions

Invalid parameter When the person ID is empty or the data is empty

No permission When the user does not have enough permissions (*i.e.* the user is not an administrator)

Drive error When a drive error has occurred while deleting some physical large file (like the identity image of the person)

General When the person ID passed as parameter is erroneous (malformed)

delOrganisation

This service deletes an organisation from the data source. For security reasons, only administrators can use this service.

Parameters

Response

id The ID of the organisation just deleted

Exceptions

Invalid parameter When the organisation ID is empty or the data is empty

No permission When the user does not have enough permissions (*i.e.* the user is not an administrator)

Drive error When a drive error has occurred while deleting some physical large file (like the identity image of the organisation)

General When the organisation ID passed as parameter is erroneous (malformed)

delDataRight

This service deletes a data right from the data source. For security reasons, only administrators can use this service.

Parameters

● **dataright** The data right to be deleted. Only the data right ID is required

Response

id The ID of the data right just deleted

Exceptions

Invalid parameter When the data right ID is empty or the data is empty

No permission When the user does not have enough permissions (*i.e.* the user is not an administrator)

General When the data right ID passed as parameter is erroneous (malformed)

delPhysicalPropertyType

This service deletes a physical property type from the data source. For security reasons, only administrators can use this service.

Parameters

● **physicalPropertyType** The physical property type to be deleted. Only the physical property type ID (located under the **type** data) is required

Response

id The ID of the physical property type just deleted

Exceptions

Invalid parameter When the physical property type ID is empty or the data is empty

No permission When the user does not have enough permissions (*i.e.* the user is not an administrator)

General When the physical property type ID passed as parameter is erroneous (malformed)

This service deletes a material type from the data source. For security reasons, only administrators can use this service.

Parameters

- **materialType** The material type to be deleted. Only the material type ID is required

Response

- id** The ID of the material type just deleted

Exceptions

- Invalid parameter** When the material type ID is empty or the data is empty
- No permission** When the user does not have enough permissions (*i.e.* the user is not an administrator)
- General** When the material type ID passed as parameter is erroneous (malformed)

delDataFileForProject

This service deletes a project data file.

Parameters

- **relatedId** The ID of the project that is related to the the data file
- **datafile** The data file to be deleted. Only the data file ID is required

Response

- id** The ID of the data file just deleted

Exceptions

- Invalid parameter** When the related ID is empty or the data file ID is empty
- No permission** When the user does not have write permissions the project that holds the data file
- Drive error** When a drive error has occurred while deleting the data file physical large file
- General** When the related ID or the data file ID passed as parameter are erroneous (malformed)

delDataFileForSpecimen

This service deletes a specimen data file.

Parameters

- **relatedId** The ID of the specimen that is related to the the data file
- **datafile** The data file to be deleted. Only the data file ID is required

Response

- id** The ID of the data file just deleted

Exceptions

Invalid parameter `InvalidInputException` When the related ID is empty or the data file ID is empty

No permission `InvalidInputException` When the user does not have write permissions the project that holds the data file

Drive error `IOException` When a drive error has occurred while deleting the data file physical large file

General `Exception` When the related ID or the data file ID passed as parameter are erroneous (malformed)

delDataFileForExperiment

This service deletes an experiment data file.

Parameters

● **relatedId** `String` The ID of the experiment that is related to the the data file

● **datafile** `DataFile` The data file to be deleted. Only the data file ID is required

Response

id `String` The ID of the data file just deleted

Exceptions

Invalid parameter `InvalidInputException` When the related ID is empty or the data file ID is empty

No permission `InvalidInputException` When the user does not have write permissions the project that holds the data file

Drive error `IOException` When a drive error has occurred while deleting the data file physical large file

General `Exception` When the related ID or the data file ID passed as parameter are erroneous (malformed)

delDataFileRelation

This service deletes an existing relation that is used between any element that contains large files (e.g. projects, experiments, specimens, etc) and the large file itself. For security reasons, only administrators can use this service.

Parameters

● **relation** `Basic` The existing relation to be deleted. Only the ID is required

Response

id `String` The ID of the data file relation just deleted

Exceptions

Invalid parameter `InvalidInputException` When the relation ID is empty

No permission `InvalidInputException` When the user does not have enough permissions (*i.e.* the user is not an administrator)

Not a data file relation security constraint violation `InvalidInputException` When the relation specified is not an existing/valid data file relation ID. Valid data file relation IDs can be obtained with the service [getAllDataFileRelations](#)

Still used data file relation security constraint violation `InvalidInputException` When the data file relation is used by at least one data file. The relation must not be in use to safely delete it

Is a parent data file relation security constraint violation `InvalidInputException` When the data file relation is a parent of other data file relations. The relation must not have any children relations to safely delete it

delIdentityImagesForProject

This service removes the identity images of a project. To update the identity image instead of remove it, or to upload a new identity image after deletion, the service [uploadIdentityImageForProject](#) can be used.

Parameters

- **id** The ID of the project to delete identity images from

Response

- imageIRIs** The IRIs (normally, URLs) of the identity images data just deleted

Exceptions

- Invalid parameter** When the project ID is empty
- No permission** When the project does not exist or the user does not have write permissions over it
- Drive error** When a drive error has occurred while processing the deletion of the image
- General** When the project ID passed as parameter is erroneous (malformed)

delIdentityImagesForSpecimen

This service removes the identity images of a specimen. To update the identity image instead of remove it, or to upload a new identity image after deletion, the service [uploadIdentityImageForSpecimen](#) can be used.

Parameters

- **id** The ID of the specimen to delete identity images from

Response

- imageIRIs** The IRIs (normally, URLs) of the identity images data just deleted

Exceptions

- Invalid parameter** When the specimen ID is empty
- No permission** When the specimen does not exist or the user does not have write permissions over it
- Drive error** When a drive error has occurred while processing the deletion of the image
- General** When the specimen ID passed as parameter is erroneous (malformed)

delIdentityImagesForExperiment

This service removes the identity images of an experiment. To update the identity image instead of remove it, or to upload a new identity image after deletion, the service [uploadIdentityImageForExperiment](#) can be used.

Parameters

- **id** The ID of the experiment to delete identity images from

imageIRIs The IRIs (normally, URLs) of the identity images data just deleted

Exceptions

Invalid parameter When the experiment ID is empty

No permission When the experiment does not exist or the user does not have write permissions over it

Drive error When a drive error has occurred while processing the deletion of the image

General When the experiment ID passed as parameter is erroneous (malformed)

delIdentityImagesForPerson

This service removes the identity images of a person. To update the identity image instead of remove it, or to upload a new identity image after deletion, the service [uploadIdentityImageForPerson](#) can be used. For security reasons, only administrators can use this service.

Parameters

● **id** The ID of the person to delete identity images from

Response

imageIRIs The IRIs (normally, URLs) of the identity images data just deleted

Exceptions

Invalid parameter When the person ID is empty

No permission When the person does not exist or the user does not have enough permissions (*i.e.* the user is not an administrator)

Drive error When a drive error has occurred while processing the deletion of the image

General When the person ID passed as parameter is erroneous (malformed)

delIdentityImagesForOrganisation

This service removes the identity images of an organisation. To update the identity image instead of remove it, or to upload a new identity image after deletion, the service [uploadIdentityImageForOrganisation](#) can be used. For security reasons, only administrators can use this service.

Parameters

● **id** The ID of the organisation to delete identity images from

Response

imageIRIs The IRIs (normally, URLs) of the identity images data just deleted

Exceptions

Invalid parameter When the organisation ID is empty

No permission When the organisation does not exist or the user does not have enough permissions (*i.e.* the user is not an administrator)

Drive error When a drive error has occurred while processing the deletion of the image

10.2 Using the API from Matlab

This section provides an example of how to use MathWorks Matlab with the CDV API. Please note that this information is just given as an indication to assist MATLAB users to use the API, and **no support is provided**. The setup of MATLAB has proven to be excessively tricky (aggravated by some limitations and bugs in Matlab) and this section intends to share some of the experiences gathered during the process.

10.2.1 Introduction

The initial setup involves two steps, which are necessary to do only once:

1. Securing the access and installation of possible user credentials
2. Creation of MATLAB code to support access to the API (required)

Securing the access entails the homologous process followed at [the Web access](#) but for MATLAB. This means that if the access to ELSADATA was marked as secure by your Web browser, you will not need to do additional steps to secure your access from MATLAB. Likewise, if you only require access to public data, no user credentials are needed. In any case, before starting is necessary you read the whole section [Secure access and private data](#) to get an idea of what you will do and its implications.

NOTE *Part of the behaviour of MATLAB for a secure access to CDV has changed at version 2016b (R2016a and before had more relaxed security constrains), so the notes included here will address later versions, specifically versions R2017a and R2018a.*

After the initial setup, then the API can be used. The next sections will explain in detail how to prepare MATLAB environment to access the API and how to call the services available.

10.2.2 Prerequisites

There are three prerequisites before the setup:

1. **Install a Java Development Kit (JDK)**. The version of JDK to install is not explicitly [specified by MATLAB](#), although they recommend to use [a version that matches](#) the version returned by the command:

```
>> version -java
```

In our experience, the old version [1.7.0_79](#) (Java SE Development Kit 7u79) worked correctly. It is always recommended to install the latest JDK, but the aforementioned version can be used if MATLAB does not accept the latest one.

2. **Install Apache CXF**. The version of CXF to install is not explicitly [specified by MATLAB](#). It is recommended to install the [latest version](#) although, similarly to the JDK, we have experienced problems and the old version [2.7.18](#) proved to work well with MATLAB and the JDK 1.7.0_79.
3. **Install Java Cryptography Extensions (JCE)**. By default, old versions of Java does not enable strong cryptography and some extensions have to be installed in the system. This includes almost all Java releases prior Java 9. The JCE should be downloaded according to the Java version of MATLAB returned by the command:

```
>> version -java
```

Normally, this will be the [JCE for the JDK 7](#) (1.7.x). Other versions, like the one for the [JDK 8](#) (1.8.x) can be easily found on the Web.

To install the extensions, the JCE ZIP file has to be extracted into the MATLAB folder returned by this command:

This step will probably require that you have or grant administrative rights to overwrite the existing files.

10.2.3 CelestinaWS script

In order to facilitate the setup process, [a script](#) has been developed. To use the script, copy it in a directory and edit it to modify some of the variables:

- JDK_DIR: The directory where the JDK is installed
- CXF_DIR: The directory where CXF is installed
- HOST: The hostname of ELSADATA. For example, if you accessed through "https://elsadata.jrc.ec.europa.eu/CelestinaDataViewer/" the hostname would be "elsadata.jrc.ec.europa.eu"
- CERTS_DIR: The directory where the certificates are located. This will be only used if the communication has to be secured (see below) and the user needs to use their credentials to access non public data. It can be any arbitrary directory, although it is recommended a subdirectory *certs* in the directory that the script is located
- SERVER_CERTALIAS: This is an alias. It is not relevant and you can set a single word like ELSADATA, UOXF, etc.
- SERVER_CERTFILE: The name of the file with the server certificate (when securing the connection is necessary). This file has to be downloaded in advance as explained in the [Secure access](#) section
- CLIENT_CERTFILE: The name of the file with the user credentials (when a user has to be used to access non public data). This file is downloaded from ELSADATA when the user is activated

Once edited, the script will be ready to be run with

```
>>celestinaWS
```

NOTE *This script comes with no support and no guarantee of working*

10.2.4 Initial setup - installation of server certificate and user credentials

Once the prerequisites have been satisfied, [the communication has to be secured](#). If your Web browser has marked your connection with ELSADATA automatically as secure, and you did not have to do any action to set the secure access (as explained in [Section 3.1](#)), then no additional steps are required to secure the communication. In this case, it is recommended that the script is edited and the line that calls the function *installServerCertificate* is commented (by using %) or removed. Otherwise, it will be necessary to do some steps to configure the secure access, similarly to what was done for the Web access. The script will do most of the work for you to configure the secure access, providing you set the right values in the script variables (specifically, the variable *SERVER_CERTFILE*).

When the celestinaWS script is run, it will first try to install the required certificates. An example of the output can be seen below:

```
>> celestinaWS

  _____
 | | / _ \ | | _ \ | | _ \ | | _ \ | | _ \ | | _ \ | | _ \ | | _ \
 | | / _ \ | | _ \ | | _ \ | | _ \ | | _ \ | | _ \ | | _ \ | | _ \
 | | / _ \ | | _ \ | | _ \ | | _ \ | | _ \ | | _ \ | | _ \ | | _ \
 | | / _ \ | | _ \ | | _ \ | | _ \ | | _ \ | | _ \ | | _ \ | | _ \

Installing Web Services for the first time
[*] Installing certificates -----
[.] Installing server certificate...
[.] Importing server certificate [/home/ilm/MATLAB/ilm/certs/elsadata.cer] into JKS keystore
[/home/ilm/MATLAB/ilm/certs/elsadata.cer.jks]...
Certificate was added to keystore
[.] Converting server certificate [/home/ilm/MATLAB/ilm/certs/elsadata.cer.jks] into PEM
[/home/ilm/MATLAB/ilm/certs/elsadata.cer.pem]...
Certificate stored in file </home/ilm/MATLAB/ilm/certs/elsadata.cer.pem>
[.] Certificates created. If there is an unexpected error before the
certificate installation is complete, please delete manually the files:
[/home/ilm/MATLAB/ilm/certs/elsadata.cer.jks]
[/home/ilm/MATLAB/ilm/certs/elsadata.cer.pem]
Otherwise, you will see a "keytool error" the next time you run this script
```

```
[*] Creating temporary java.opts file in current directory...
[?] Do you need to configure a user to access private data [y,n]: y
[?] Please input your password for the P12 file (/home/ilm/MATLAB/ilm/certs/iniadmin.p12)
Password: iniadmin
[.] File [java.opts] created...
[.] Moving file [java.opts] to [/home/ilm/MATLAB/R2017a/bin/glnxa64/java.opts]...
Note: This step involves the use of a java.opts file at [/home/ilm/MATLAB/R2017a/bin/glnxa64/java.opts]
which will change your standard keystore in order to be less intrusive.
However, this might change the way you interact with other secure systems
from MATLAB. If you want to restore the standard keystore and still use CDV,
just edit the java.opts file to remove trustStore lines and include the server
certificate [/home/ilm/MATLAB/ilm/certs/elsadata.cer] into [/home/ilm/MATLAB/R2017a/sys/certificates/ca/rootcerts.pem].
[.] The file [/home/ilm/MATLAB/R2017a/bin/glnxa64/java.opts] already exists. A backup will be made and
the file will be overwritten. Please, merge the backup file manually
if you consider it necessary.
[.] Backing up file [/home/ilm/MATLAB/R2017a/bin/glnxa64/java.opts] as
[/home/ilm/MATLAB/R2017a/bin/glnxa64/java.opts_2018-06-18_1927.org]...
Matlab needs to be restarted to apply the changes.
Now please restart Matlab and execute this script again
Press ENTER to exit Matlab...
```

At this point, everything has been configured successfully and MATLAB needs to be restarted to be able to use the new configuration. This step does not have to be repeated unless ELSADATA changes its server certificate (and it is not trusted by default) or new user credentials have to be installed.

10.2.5 Initial setup - generating MATLAB code helper

After restarting MATLAB, the script has to be run again. This time, it will generate the auxiliary classes that will be used by MATLAB to communicate with the API. These MATLAB code will be generated in your current directory. The script will detect that the certificates were installed successfully and will skip the step discussed in the previous section. An example of output can be seen below:

```
>> celestinaWS

  _ _ _ _ _
 / _ | _ | | _ _ | _ | _ | _ |
 | | / _ \ | / _ \ | _ | | ' _ \ / _ \ |
 | _ | _ | | _ \ _ \ | | | | | | | |
 \ _ \ | | \ _ \ | | \ _ \ | | | | \ _ \ |

Installing Web Services for the first time
[*] Installing certificates -----
Server and Client certificates seem to have been already created. To force recreation, delete the file [installed_cert]
and run the script again
[*] Creating Web Services in current directory -----

Created data.
./data.m
./+wsdl

In order to use data, you must run javaaddpath('./+wsdl/data.jar').
[*] Cleaning validations -----
- Basic.m
- BasicData.m
- BasicDataWithDataFiles.m
- BasicWithParents.m
- CdvDate.m
- DataFile.m
- DataRight.m
- Device.m
- ExperimentalActivity.m
- GeneralData.m
- Location.m
- Magnitude.m
- Material.m
- MaterialCategory.m
- MaterialType.m
- OntologyClass.m
```

```

- FileComponent.m
- Person.m
- PhysicalProperty.m
- Plan2D.m
- Point2D.m
- Project.m
- SignalData.m
- Specimen.m
- SpecimenComponent.m
- TypeAndUnit.m
- UploadFile.m

... INSTALLATION DONE!
[*] Listing services and object creation methods available:
Methods for class data:

createDataFileRelation          delIdentityImageForSpecimen    getSignalsOfExperiment
createDataRight                 delMaterialType                getSpecimen
createExperimentForProject      delOrganisation                getSpecimensIds
createExperimentForSpecimen    delPerson                       saveDataRight
createExperimentSpecimenRelation delPhysicalPropertyType        saveExperiment
createMaterialType             delProject                      saveOrganisation
createNumericalExperimentForProject delSignal                       savePerson
createNumericalExperimentForSpecimen delSpecimen                     saveProject
createOrganisation             display                          saveSpecimen
createPerson                    getAllDataFileRelations        testWS
createPhysicalPropertyType     getAllDataRights               testWSParamException
createProject                   getAllMaterialTypes            uploadDataFile
createSpecimenForExperiment    getAllOrganisations
uploadDataFileForExperimentalActivity
createSpecimenForProject      getAllPersons                   uploadDataFileForProject
data                           getAllPhysicalPropertyTypes    uploadDataFileForSpecimen
delDataFileForExperimentalActivity getAllSignalMagnitudes
uploadIdentityImageForExperimentalActivity
delDataFileForProject         getAllSpecimenComponentTypes
uploadIdentityImageForOrganisation
delDataFileForSpecimen        getAllUnits                      uploadIdentityImageForPerson
delDataFileRelation           getDataFileDownloadAddress      uploadIdentityImageForProject
delDataRight                  getExperiment                   uploadIdentityImageForSpecimen
delExperiment                  getExperimentsIds              uploadInputSignals
delExperimentSpecimenRelation getProject                       uploadOutputSignals
delIdentityImageForExperimentalActivity getProjectByName                 whoAmI
delIdentityImageForOrganisation getProjects
delIdentityImageForPerson      getProjectsIds
delIdentityImageForProject     getSignal

Static methods:

getBasic                        getMaterial                      getPoint2D
getCdvDate                      getMaterialCategory              getProjectObject
getDataFile                     getMaterialType                  getSignalData
getDataRight                    getOntologyClass                 getSpecimenComponent
getDevice                        getOrganisation                  getSpecimenObject
getExperimentalActivity          getPerson                         getTypeAndUnit
getLocation                      getPhysicalProperty              getUploadFile
getMagnitude                     getPlan2D

```

Once the installation is completed (informed by the message "INSTALLATION DONE"), the available services are listed and the API is ready to be used.

This step does not have to be repeated (*i.e.* the MATLAB classes do not have to be re-generated) unless a new CDV version that changes the API is installed at ELSADATA.

10.2.6 Using the API

The API is ready to be used now. In future uses of the API after the setup is completed, the script should be run before calling any service. This step is not really necessary because the script will run only one command, but it is a convenient way to have it executed. The script will inform that both the certificates and the Web Services have been already installed:

```

_ _ _ _ _
/ _ | _ | _ _ _ | _ ( ) _ _ _ _
| | / _ \ | / _ \ | _ | | ' _ \ / _ |
| | | _ / | _ \ \ | | | | | | ( | |
\ \ | | \ \ | | \ \ | | | | | \ \ |

```

Web Services (and certificates) seem to have been already installed.
 If you wish to reinstall everything, just delete the files with name
 "installed_ws" and "installed_cert" in the current directory
 [*] Listing services and object creation methods available:
 Methods for class data:
 ...

In order to see the available services, the following command can be executed from the current script directory:

```

>> methods(data)

Methods for class data:

createDataFileRelation          delIdentityImageForSpecimen      getSignalsOfExperiment
createDataRight                 delMaterialType                  getSpecimen
createExperimentForProject       delOrganisation                  getSpecimensIds
createExperimentForSpecimen     delPerson                         saveDataRight
createExperimentSpecimenRelation delPhysicalPropertyType          saveExperiment
createMaterialType              delProject                       saveOrganisation
createNumericalExperimentForProject delSignal                         savePerson
createNumericalExperimentForSpecimen delSpecimen                      saveProject
createOrganisation              display                          saveSpecimen
createPerson                    getAllDataFileRelations          testWS
createPhysicalPropertyType      getAllDataRights                 testWSParamException
createProject                   getAllMaterialTypes              uploadDataFile
createSpecimenForExperiment     getAllOrganisations
uploadDataFileForExperimentalActivity getAllPersons                     uploadDataFileForProject
createSpecimenForProject        getAllPhysicalPropertyTypes      uploadDataFileForSpecimen
data                             getAllSignalMagnitudes
delDataFileForExperimentalActivity getAllSpecimenComponentTypes
uploadIdentityImageForExperimentalActivity
delDataFileForProject
uploadIdentityImageForOrganisation
delDataFileForSpecimen         getAllUnits                       uploadIdentityImageForPerson
delDataFileRelation            getDataFileDownloadAddress        uploadIdentityImageForProject
delDataRight                   getExperiment                     uploadIdentityImageForSpecimen
delExperiment                   getExperimentsIds                 uploadInputSignals
delExperimentSpecimenRelation  getProject                        uploadOutputSignals
delIdentityImageForExperimentalActivity getProjectByName                  whoAmI
delIdentityImageForOrganisation getProjects
delIdentityImageForPerson       getProjectsIds
delIdentityImageForProject      getSignal

Static methods:

getBasic                        getMaterial                       getPoint2D
getCdvDate                     getMaterialCategory               getProjectObject
getDataFile                    getMaterialType                   getSignalData
getDataRight                   getOntologyClass                  getSpecimenComponent
getDevice                       getOrganisation                   getSpecimenObject
getExperimentalActivity         getPerson                          getTypeAndUnit
getLocation                    getPhysicalProperty               getUploadFile
getMagnitude                    getPlan2D

```

The available services can be generally called by using this format:

```
SERVICE_NAME(data, INPUT_PARAM1, INPUT_PARAM2 ...)
```

parameters as required by the service. It is recommended that the first time, a [testing service](#) is called, to verify everything is working correctly. For example, the [testWS service](#) receives no parameters and returns a String. It can be called this way:

```
>> testWS(data)
ans =

    'If you can see this, the Web Service has been called successfully'
```

As seen in the output, the call was successful. Possible problems are discussed below. Next, it is suggested to verify the user that CDV has authenticated from our MATLAB access by using the [whoAmI service](#):

```
whoAmI(data)

ans =

    'ADMIN INI'
```

In this case we are using a temporary administrator. If you have configured a user and you see "PUBLIC" as an answer, it means that something went wrong with the user credentials configuration.

For example, this code will retrieve the list of projects available and will show their respective names:

```
>> pids = getProjectsIds(data) % Retrieve the IDs of all projects

pids =

    4x1 cell array

    'http://jrc.ec.europa.eu/celestina#project193'
    'http://jrc.ec.europa.eu/celestina#project210'
    'http://jrc.ec.europa.eu/celestina#project249'
    'http://jrc.ec.europa.eu/celestina#project259'

>> for i=1:length(pids) % Loop over all ids
    pid = pids(i);
    project = getProject(data, pid{1}); % Get all data about a specific project
    disp(project.name); % Print its name
end
% Results:
Spear ELSA
PrecastEC8 ELSA
SAFECAST ELSA
SAFECLADDING ELSA
```

Sometimes it is necessary to create objects. Objects used by the API can be created by using this format:

```
wsdl.data.OBJECT_CLASS_NAME
```

For example, to create a *Project* object and assign it to a variable *p1*:

```
p1 = wsdl.data.Project
```

Or by using this format:

```
data.getOBJECT_METHOD
```

Which for the previous example will be:

```
p1 = data.getProjectObject
```

```
>> p1 = getProject(data, 'http://jrc.ec.europa.eu/celestina#project259')

p1 =

Project with properties:

    experimentalActs: [39x1 wsdl.data.ExperimentalActivity]
    specimens: [14x1 wsdl.data.Specimen]
    dataRights: []
    endDate: [1x1 wsdl.data.CdvDate]
    identityImageIris: []
    intellectualPropertyOwners: []
    keywords: {9x1 cell}
    participants: [1x1 wsdl.data.Organisation]
    purpose: []
    shortName: 'SAFECLADDING ELSA'
    startDate: [1x1 wsdl.data.CdvDate]
    datafiles: [2x1 wsdl.data.DataFile]
    description: 'Improved Fastening Systems of Cladding Panels for Precast Buildings in Seismic Zones'
    id: 'http://jrc.ec.europa.eu/celestina#project259'
    name: 'SAFECLADDING ELSA'
```

10.2.7 Data from signals and files

When using the API, only metadata is normally retrieved from objects. For example, if a document (a *DataFile*) is being accessed via the API, information about the document will be read, such as its name, description, creation date, and so on, but the actual document data will not be obtained. To get the actual data, the attribute *downloadIRI* must be used. If an element has data associated, such as data files and signals, the *downloadIRI* attribute will contain the URL from where the actual data can be downloaded. Downloading the actual data, once the *downloadIRI* has been obtained, is a simple process in MATLAB:

```
>> websave('example.csv', downloadIri)
```

Alternatively, a Web browser or software such as *curl* can be used.

10.2.8 Additional examples

NOTE *These scripts have been developed by third parties and they come with no support and no guarantee of working (especially if new versions of CDV are released and these scripts are not updated accordingly). The scripts have been kindly shared to provide examples and show how the API can be used.*

[Examples by Elias Strepelias \(University of Patras\)](#)

10.2.9 Common issues

PROBLEM

```
Error using data/whoAmI (line 1085)
Java exception occurred:
com.sun.xml.internal.ws.client.ClientTransportException: HTTP transport error: javax.net.ssl.SSLHandshakeException:
Received fatal alert: handshake_failure
    at com.sun.xml.internal.ws.transport.http.client.HttpClientTransport.getOutputStream(Unknown Source)
    at com.sun.xml.internal.ws.transport.http.client.HttpTransportPipe.process(Unknown Source)
    at com.sun.xml.internal.ws.transport.http.client.HttpTransportPipe.processRequest(Unknown Source)
    at com.sun.xml.internal.ws.transport.DeferredTransportPipe.processRequest(Unknown Source)
    at com.sun.xml.internal.ws.api.pipe.Fiber.__doRun(Unknown Source)
```

```

at com.sun.xml.internal.ws.api.pipe.Fiber.run(Unknown Source)
at com.sun.xml.internal.ws.api.pipe.Fiber.runSync(Unknown Source)
at com.sun.xml.internal.ws.client.Stub.process(Unknown Source)
at com.sun.xml.internal.ws.client.sei.SEIStub.doProcess(Unknown Source)
at com.sun.xml.internal.ws.client.sei.SyncMethodHandler.invoke(Unknown Source)
at com.sun.xml.internal.ws.client.sei.SyncMethodHandler.invoke(Unknown Source)
at com.sun.xml.internal.ws.client.sei.SEIStub.invoke(Unknown Source)
at com.sun.proxy.$Proxy47.whoAmI(Unknown Source)

```

Caused by: javax.net.ssl.SSLHandshakeException: Received fatal alert: handshake_failure

```

at sun.security.ssl.Alerts.getSSLException(Unknown Source)
at sun.security.ssl.Alerts.getSSLException(Unknown Source)
at sun.security.ssl.SSLSocketImpl.recvAlert(Unknown Source)
at sun.security.ssl.SSLSocketImpl.readRecord(Unknown Source)
at sun.security.ssl.SSLSocketImpl.performInitialHandshake(Unknown Source)
at sun.security.ssl.SSLSocketImpl.startHandshake(Unknown Source)
at sun.security.ssl.SSLSocketImpl.startHandshake(Unknown Source)
at sun.net.www.protocol.https.HttpsClient.afterConnect(Unknown Source)
at sun.net.www.protocol.https.AbstractDelegateHttpsURLConnection.connect(Unknown Source)
at sun.net.www.protocol.http.HttpURLConnection.getOutputStream(Unknown Source)
at sun.net.www.protocol.https.HttpsURLConnectionImpl.getOutputStream(Unknown Source)
... 14 more

```

SOLUTION It is possible that the handshake has failed for an incorrect ciphersuite. Make sure that JRE has been installed. If the problem persists, it is possible that MATLAB is not accepting any of the strong cryptography demanded by the application. In this case, use R2018 or superior, or update the JRE that comes with MATLAB.

PROBLEM

```
>> methods(data)
```

```
The class data has no Constant property or Static method named 'Data'.
```

```
Error in data (line 27)
```

```
wsdlService = data.Data();
```

SOLUTION Make sure the Java classpath has been correctly set

```
>> javaaddpath('.\+wsdl\data.jar')
```