This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 777533.

PROviding Computing solutions for ExaScale ChallengeS

D8.4	Final release of PROCESS software		
Project:	PROCESS H2020 – 777533	**Start / Duration:**	01 November 2017 36 Months
Dissemination[1]:	Public	**Nature[2]:**	**R**
Due Date:	31 October 2020	**Work Package:**	**WP 8**
Filename[3]	PROCESS_D8.4_FinalReleasePROCESSSoftware_v1.0.docx		

ABSTRACT

This deliverable accompanies the final release of PROCESS. We provide an overview of the released services, tools and user interface of PROCESS; example orchestration files to start these services in unison; the use case benchmarks, workflows and data sets; several additional software releases, and a packaged version of a production subset of the PROCESS ecosystem.

[1] PU = Public; CO = Confidential, only for members of the Consortium (including the EC services).
[2] R = Report; R+O = Report plus Other. Note: all "O" deliverables must be accompanied by a deliverable report.
[3] eg DX.Y_name to the deliverable_v0xx. v1 corresponds to the final release submitted to the EC.

Deliverable Contributors:	Name	Organization	Role / Title
Deliverable Leader[4]	Maassen J.	NLESC	Deliverable coordinator
Contributing Authors[5]	Cushing, R.	UvA	Writer
	Meizner, J.	AGH	Writer
	Nowakowski, P.	AGH	Writer
	Schmidt, J.	LMU	Writer
	Somoskoi, B.	LSY	Writer
	Madougou, S.	NLESC	Writer
	Graziani, M	HES-SO	Writer
	Tran, V.	UISAV	Writer
Reviewer(s)[6]	Höb, M.	LMU	Reviewer
	Bobák, M.	USIAV	Reviewer
Final review and approval	Höb, M.	LMU	Coordinator

Document History

Release	Date	Reasons for Change	Status[7]	Distribution
0.1	02.10.2020	Initial structure of document	Draft	Working Group
0.2	07.10.2020	Abstract + introduction added	Draft	Working Group
0.3	20.10.2020	Content updated	Draft	Working Group
0.4	25.10.2020	Fixed TOC/LOF/LOT	Draft	Working Group
0.5	26.10.2020	Ready for review	In Review	Consortium
0.6	29.10.2020	Updated LOBCDER section	In Review	Consortium
1.0	31.10.2020	Final Version	Released	Public

[4] Person from the lead beneficiary that is responsible for the deliverable.
[5] Person(s) from contributing partners for the deliverable.
[6] Typically, person(s) with appropriate expertise to assess the deliverable quality.
[7] Status = “Draft”; “In Review”; “Released”.

Table of Contents

Executive Summary

This deliverable accompanies the final release of the PROCESS software. We provide an overview of the released services, tools and user interfaces; example orchestration files to start these services in unison; the use case benchmarks codes, workflows and data sets; several additional software releases; and a packaged version of a production subset of the PROCESS ecosystem.

For all the services and use case codes we follow the FAIR software recommendations as published on https://fair-software.eu, and share the software on our online research software directory at https://software.process-project.eu.

Next to the codes developed within the PROCESS project, we also make use of several off-the-shelf open-source components developed externally. For these components, we refer to the relevant releases used within the PROCESS software stack. We also provide the technical readiness level (TRL) of these services.

The essential details of the releases are described in this deliverable. More information can be found in our software directory, which will also be kept up-to-date after this deliverable has been released.

List of Figures

List of Tables

1 Introduction

This document accompanies the final release of the PROCESS. This final release consists of the following:

1. Individually packaged releases of the PROCESS services
2. Orchestration files to create a stand-alone instances of (a subset of) the PROCESS services for testing and demonstration purposes
3. Releases of the use case benchmark software, workflows, and datasets (where applicable)
4. Releases of additional software.

The PROCESS services play a central role in the final PROCESS architecture. This final architecture is shown in Figure 1 and described in detail in D4.5. The dashed red boxes in Figure 1 highlight the service components in PROCESS: IEE, Rimrock, Cloudify, LOBCDER (with its micro-infrastructure containers), and DataNet, which will be part of the final release. These individual releases are described in detail in Section 2.

In addition to the individual releases of the services, we also release the necessary orchestration files to create a virtual machine (VM) images which enables a straightforward deployment of the PROCESS services for testing and demonstration purposes. Details on this can be found in Section 3.

We release the software, workflows and datasets for the individual use cases used as benchmarks to develop the PROCESS services (wherever possible). Additionally, we also release some software packages which were developed or adapted during the PROCESS project. These will be described in Section 4.

For all of the codes we have developed ourselves we follow the FAIR software recommendations as published on https://fair-software.eu. This implies that next to providing a short description of each service and its role in PROCESS as well as links to documentation, we also ensure the source code is available in a public repository (http://github.com or https://gitlab.com) with an open source license. Each release is archived and made citable through https://zenodo.org, and the software is made findable by publishing it via our online research software directory at https://software.process-project.eu.

In this software directory, more details can be found for each of the software releases, such as the contributors, links to the source code repositories and documentation, a list of related scientific publications, presentation, blogs, videos, deliverables, etc. As this software directory will be updated after the release of this deliverable, it may in some cases contain more up-to-date information than what is shown below.

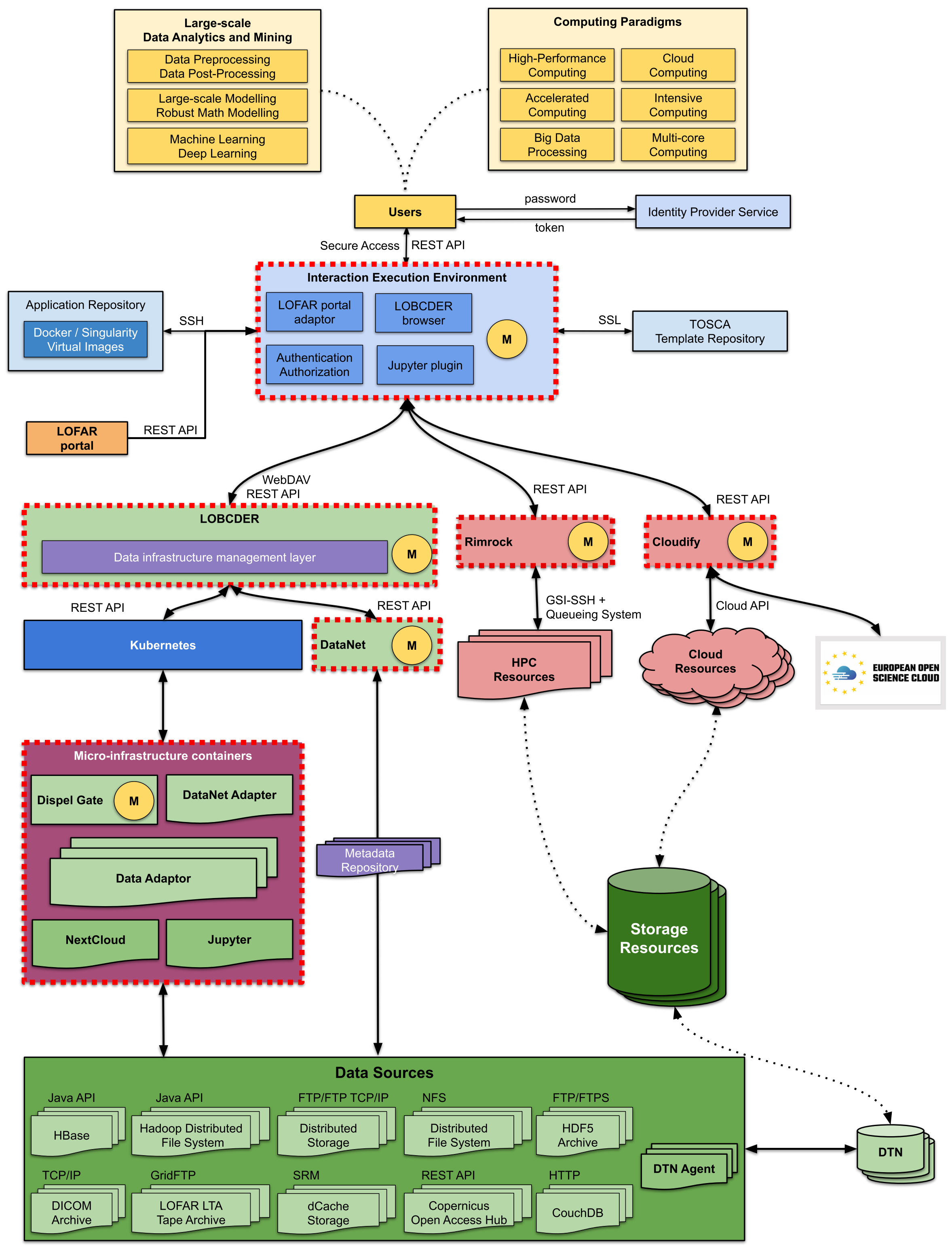


Figure 1: PROCESS architecture

2 Final release of the PROCESS services

This section provides the relevant information on the final release of the PROCESS services. In Figure 1, the dashed red boxes highlight the PROCESS service components.

The IEE, RIMROCK and LOBCDER services have been developed independently in projects predating PROCESS. In PROCESS, they have been further developed and modified to meet the PROCESS requirements and improve their TRL level. For each of these services, we have released a version specifically for the PROCESS production prototype. These releases are described in more detail below.

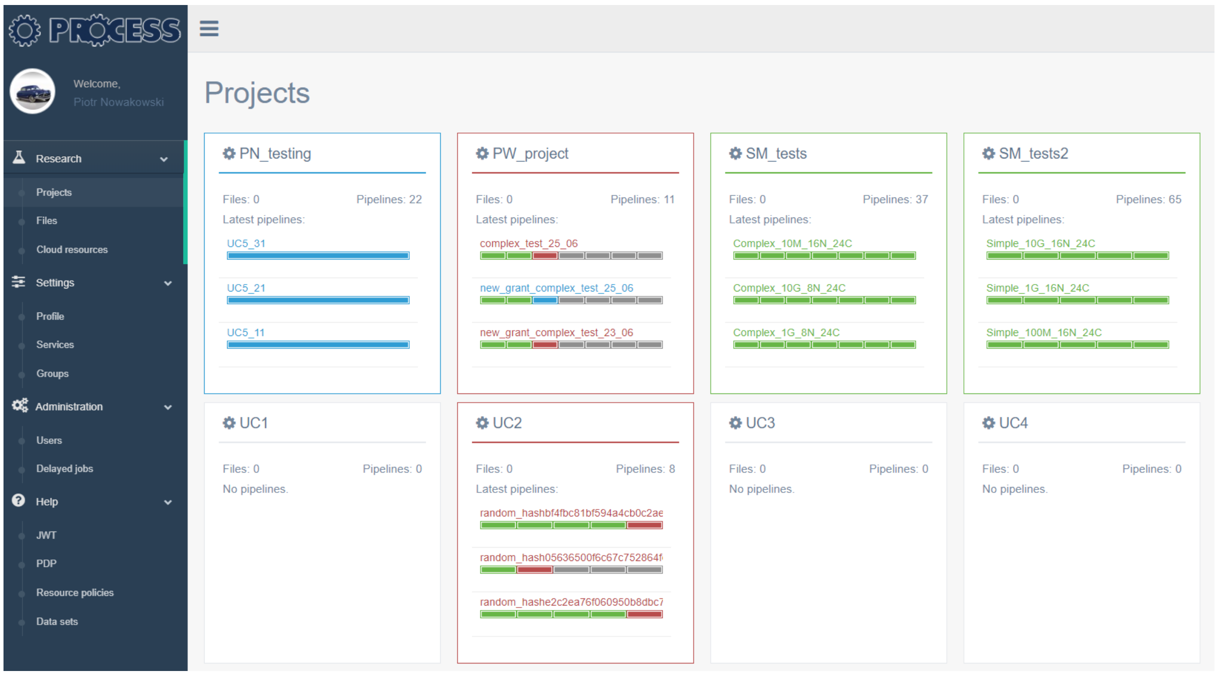
Cloudify is an off-the-shelf component which is developed externally and is used extensively in commercial cloud datacentres. It is developed and supported by Cloudify Ltd., and available in both a commercial and open source community version. In the process software stack, we use the unmodified community version, whose details are described below. We will also provide the necessary configuration files and service blueprints used to integrate the PROCESS services with Cloudify.

The services are also described in more detail on the PROCESS project website at: https://www.process-project.eu/software/. For all services developed or modified in the PROCESS project, we follow the FAIR software recommendations for releases, as described in the introduction. We also estimate the technical readiness level (TRL) of each of the released services.

In this release, we also include the necessary orchestration and configuration files needed to run the PROCESS software on the PROCESS testbed. In addition, we also release a set of *services demonstrators*, which provide virtualised, stand-alone instances of the PROCESS services. These demonstrators do not rely on the PROCESS testbed and can be used for testing and demonstration purposes, as well as serve as a starting point for new users.

2.1 The Interactive Execution Environment (IEE)

The Interactive Execution Environment (IEE) is a graphical web interface (shown in Figure 2), which provides a one-stop entry point to the PROCESS infrastructure and execution of computational tasks on the underlying HPC and cloud resources. The IEE manages the required delegation of security credentials and abstracts away all details related to interaction with various types of computational resources.



Projects

PN_testing
Files: 0 Pipelines: 22
Latest pipelines:
UC5_31
UC5_21
UC5_11

PW_project
Files: 0 Pipelines: 11
Latest pipelines:
complex_test_25_06
new_grant_complex_test_25_06
new_grant_complex_test_23_06

SM_tests
Files: 0 Pipelines: 37
Latest pipelines:
Complex_10M_16N_24C
Complex_10G_8N_24C
Complex_1G_8N_24C

SM_tests2
Files: 0 Pipelines: 65
Latest pipelines:
Simple_10G_16N_24C
Simple_1G_16N_24C
Simple_100M_16N_24C

UC1
Files: 0 Pipelines: 0
No pipelines.

UC2
Files: 0 Pipelines: 8
Latest pipelines:
random_hashbf4fbc81bf594a4cb0c2ae
random_hash05636500f6c67c752864f
random_hashe2c2ea76f060950b8dbc7

UC3
Files: 0 Pipelines: 0
No pipelines.

UC4
Files: 0 Pipelines: 0
No pipelines.

Figure 2: The IEE user interface

IEE works by providing access to configurable computational pipelines which represent individual use cases. Each pipeline may be launched an arbitrary number of times, with varying parameters. The IEE takes care of acquiring the required resources, scheduling data transfers (via LOBCDER) and executing the computational steps which build the use cases. IEE can schedule computations on traditional HPC compute clusters (via RimRock) and cloud resources (via the Cloudify extension). It can also communicate with bespoke REST interfaces offered by providers of external computational services - such as with the AgroCopernicus use case (UC#5). In addition, it also allows access through a REST API for use cases that require a use-case-specific user interface, such as the LOFAR use case (UC#2).

The IEE was originally developed in the EuroValve[8] project and is still being used there. For PROCESS, it was adapted and extended to meet the requirements of our use cases. We therefore estimate its TRL as 8.

Table 1: IEE - PROCESS final release

Software catalogue	https://software.process-project.eu/software/iee
Source code repository	https://gitlab.com/cyfronet/iee/-/tree/1.0.0
Install documentation	https://gitlab.com/cyfronet/iee/-/blob/1.0.0/README.md
User documentation	Available within GUI environment
DOI	10.5281/zenodo.3950336
Version	1.0.0
License	MIT
TRL	8

[8] http://www.eurvalve.eu

2.2 LOBCDER

Moving data in the exascale era is very time consuming because moving data is bound by the underlying limitations of the network infrastructure. One exabyte of data takes around 3 years to transfer on the faster, state of the art interconnects. At the same time projects like the Square Kilometre Array are expected to generate an exabyte of data every day. This puts a lot of strain on how to handle and process data which is globally distributed and difficult to move.

Our approach in PROCESS, is that the data services cannot be viewed as just software components. Instead a more holistic approach is needed which takes into consideration the underlying infrastructure, the flow of data, the processing and the services. This leads to a smart infrastructure where we can, for example, move the computation to the data instead of moving the data or split the processing of data into several parts, each being executed on different parts of the network.

LOBCDER introduces the concept of a micro-infrastructure which uses container technologies to combine data services with infrastructure to create a smart data infrastructure. This means that development of services now consists of two parts; the description of the underlying container infrastructure (micro-infrastructure) and the development of the different software services that fit into the micro-infrastructure.

In PROCESS, LOBCDER is used by the IEE to deploy use case specific data services, which allow use cases to adapt the data access and transfers to their needs. To enable this, LOBCDER consists of a large collection of *core, service* and *adaptor* components, which can be assembled to fit the individual use cases. The core component provide the basic implementation of the micro-infrastructure, while the services and adaptors provide the functionality needed by the use cases. Where the services offer a remotely accessible API, the adaptors are only used internally.

LOBCDER has significantly extended to fit the requirements of the use cases and the testbed. Therefore, we estimate it to be at TRL 7.

2.2.1 Core components

The LOBCDERS *core-infra* component is the heart or the micro-infrastructure and responsible for starting and managing the other components, including the use case specific services.

Table 2: LOBCDER core-infra – PROCESS final release

Software catalogue	https://software.process-project.eu/software/lobcder-core-infra
Source code repo	https://github.com/micro-infrastructure/core-infra
Documentation	https://github.com/micro-infrastructure/documentation
DOI	10.5281/zenodo.4154379
Version	0.1.0
License	MIT
TRL	7

LOBCDER *core-proxy* is a component that exposes functions provided by the service containers to the internet through a REST API. This provides a standardized way to call routines in service containers running as part of LOBCDER, without having to fully expose every individual container to the internet.

Table 3: LOBCDER core-proxy – PROCESS final release

Software catalogue	https://software.process-project.eu/software/lobcder-core-proxy
Source code repo	https://github.com/micro-infrastructure/core-proxy
Documentation	https://github.com/micro-infrastructure/documentation
DOI	10.5281/zenodo.4156351
Version	0.1.2
License	MIT
TRL	7

Finally, the *core-mongo* component contains a mongoDB-based database used by the core-infra component for persistent storage.

Table 4: LOBCDER core-mongo – PROCESS final release

Software catalogue	https://software.process-project.eu/software/lobcder-core-mongo
Source code repo	https://github.com/micro-infrastructure/core-mongo
Documentation	https://github.com/micro-infrastructure/documentation
DOI	10.5281/zenodo.4156357
Version	0.1.2
License	MIT
TRL	7

2.2.2 Services

The *scp2scp* service for the LOBCDER micro-infrastructure provides file transfers between HPC cluster using the SCP protocol. This service is used by IEE in several use cases to transfer in and output data to and from HPC sites.

Table 5: LOBCDER service-scp2scp -- PROCESS final release

Software catalogue	https://software.process-project.eu/software/lobcder-service-scp2scp
Source code repo	https://github.com/micro-infrastructure/service-scp2scp
Documentation	https://github.com/micro-infrastructure/documentation
DOI	10.5281/zenodo.4154379
Version	0.2.7
License	MIT

The *urlshare* service provides plain URL based file sharing. This service is used by UC2 to access stage-out the output data of workflows.

Table 6: LOBCDER service-urlshare -- PROCESS final release

Software catalogue	https://software.process-project.eu/software/lobcder-service-urlshare
Source code repo	https://github.com/micro-infrastructure/service-urlshare
Documentation	https://github.com/micro-infrastructure/documentation
DOI	10.5281/zenodo.4154867
Version	0.1.2
License	MIT

The *webdavserver* repository contains a service for the LOBCDER micro-infrastructure which provides file sharing based on the WebDAV protocol. This service is use cases to view remote data and manipulate small datasets.

Table 7: LOBCDER service-webdavserver -- PROCESS final release

Software catalogue	https://software.process-project.eu/software/lobcder-service-webdavserver
Source code repo	https://github.com/micro-infrastructure/service-webdavserver
Documentation	https://github.com/micro-infrastructure/documentation
DOI	10.5281/zenodo.4154875
Version	0.1.2
License	MIT

2.2.3 Adaptors

The *sshfs* adaptor allows LOBCDER to mount remote file systems using SSHFS. This is often used in combination with the WebDAV service to expose file systems of remote HPC clusters.

Table 8: LOBCDER adaptor-sshfs -- PROCESS final release

Software catalogue	https://software.process-project.eu/software/lobcder-adaptor-sshfs
Source code repo	https://github.com/micro-infrastructure/adaptor-sshfs
Documentation	https://github.com/micro-infrastructure/documentation
DOI	10.5281/zenodo.4154703
Version	0.1.0
License	MIT

The *gridftp* adaptor allows LOBCDER to perform data transfers between two sites using the GridFTP protocol, which supports third-party copy and parallel TCP streams for increased performance. The *gridftp* adaptor is used internally by the *scp2scp* service for data transfers between sites that support the GridFTP protocol.

Table 9: LOBCDER adaptor-gridftp -- PROCESS final release

Software catalogue	https://software.process-project.eu/software/lobcder-adaptor-gridftp
Source code repo	https://github.com/micro-infrastructure/adaptor-gridftp
Documentation	https://github.com/micro-infrastructure/documentation
DOI	10.5281/zenodo.4154732
Version	0.1.0
License	MIT

The *srm2local* adaptor allows LOBCDER adaptor to data from servers using the SRM protocol, such as the LOFAR Long Term Archive (LTA) used in UC#2.

Table 10: LOBCDER adaptor-srm2local -- PROCESS final release

Software catalogue	https://software.process-project.eu/software/lobcder-adaptor-srm2local
Source code repo	https://github.com/micro-infrastructure/adaptor-srm2local
Documentation	https://github.com/micro-infrastructure/documentation
DOI	10.5281/zenodo.4154794
Version	0.1.0
License	MIT

The *lofar-stage* adaptor perform staging of data on the LOFAR LTA. When staging, data is moved in the archive from tape to temporary disk, after which this data can be retrieved using the srm2local adaptor.

Software catalogue	https://software.process-project.eu/software/lobcder-adaptor-lofar-stage
Source code repo	https://github.com/micro-infrastructure/adaptor-lofar-stage
Documentation	https://github.com/micro-infrastructure/documentation
DOI	10.5281/zenodo.4154768
Version	0.1.0
License	MIT

The *fdt* adaptor can be used to perform data transfers between remote sites using the fdt library which supports high-performance data transfers using parallel TCP streams. The *fdt* adaptor is used internally by the *scp2scp* service for data transfers between sites that support the *fdt* protocol.

Table 11: LOBCDER adaptor-fdt -- PROCESS final release

Software catalogue	https://software.process-project.eu/software/lobcder-adaptor-fdt
Source code repo	https://github.com/micro-infrastructure/adaptor-fdt
Documentation	https://github.com/micro-infrastructure/documentation
DOI	10.5281/zenodo.4154750
Version	0.1.0
License	MIT

2.3 RIMROCK

Rimrock provides a REST API to existing scientific compute infrastructures, such as HPC clusters. By providing a REST interface, access to compute services, applications and advanced scripts deployed on the infrastructure becomes independent of the local software stack. The main advantage of Rimrock is the ability to connect to any underlying software technology and provide an integrated solution for secure access to computing infrastructures.

By providing a REST interface, the Rimrock service allows to use its functionalities independently of any programming language chosen to build applications on top of the computing infrastructure. It is therefore possible to create web and desktop applications as well as prepare advanced computation scripts. An interesting approach, also supported by the service, is the ability to develop web applications, which can be run solely in the user's web browser, minimizing the role of server-side software.

All data exchanged with the Rimrock service is fully secured with an encrypted connection and for user authorisation a temporary user certificate is used, ensuring that long-lived credentials would not leak, giving permanent access to the attacker.

In PROCESS, Rimrock is used by the IEE to access the HPC compute infrastructure of the PROCESS testbed. To do so, it was extended with support for container-based applications and workflows, which is used by UC#1 and UC#2.

Before PROCESS, Rimrock had already been successfully used during the development of a web application in the domain of the energy sector. Thereby, allowing for harnessing of the computing power of the PLGrid infrastructure for the analysis of different scenarios when building a national power grid and the influence on the environment and human health. We therefore estimate its current TRL level to be 8.

Table 12: RimRock - PROCESS final release

Process website	https://www.process-project.eu/software/
Software catalogue	https://software.process-project.eu/software/rimrock
Source code repo	https://gitlab.com/cyfronet/rimrock
Install documentation	https://gitlab.com/cyfronet/rimrock/-/blob/master/doc/install/installation.md
User documentation	https://rimrock.plgrid.pl/jobs
DOI	10.5281/zenodo.3949651
Version	2.0.3
License	MIT
TRL	8

2.4 Cloudify

Service orchestration is a process for automated configuration, deployment and management of services and applications in the cloud. It can automate the execution of different service workflows including deployment, initialisation, start/stop, scaling, healing of services based on standardised descriptions of composed services, relations between components and their requirements. In the PROCESS project, we use the OASIS TOSCA standard for service description and use Cloudify for orchestration.

Cloudify is an externally developed open-source cloud orchestration platform, designed to automate the deployment, configuration and remediation of application and network services across hybrid cloud and stack environments. It uses OASIS TOSCA templates written in YAML (called blueprints in Cloudify) for defining applications, services and dependencies among them. These blueprint files describe the execution plans for the lifecycle of the application for installing, starting, terminating, orchestrating and monitoring the application stack. Cloudify uses the blueprint as input that describes the deployment plan and is responsible for executing it on the cloud environment.

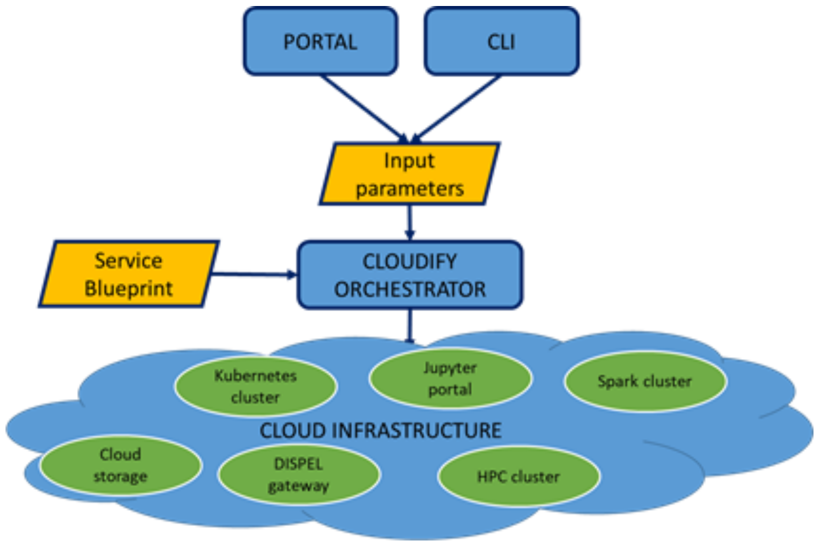


Figure 3: Cloudify orchestration

In PROCESS, Cloudify is used by the IEE to access the cloud infrastructure available as part of our testbed. The role of Cloudify is similar to that of Rimrock. While Rimrock provides unified access to the HPC compute infrastructure, Cloudify provides this access to the cloud infrastructure. This cloud infrastructure is used in UC#4.

As Cloudify is developed commercially and used extensively in commercial cloud data centres, we estimate its TRL level to be 9.

Table 13: Cloudify – PROCESS final release

Process website	https://www.process-project.eu/software/
Software catalogue	https://software.process-project.eu/software/cloudify
Source code repo	https://github.com/cloudify-cosmo
Install documentation	https://docs.cloudify.co/latest/install_maintain/
User documentation	https://docs.cloudify.co/latest/working_with/
DOI	N/A (externally developed)
Version	18.10.4-community
License	Apache-2
TRL	9

We also release the necessary configuration files and orchestration blueprints which define the PROCESS service and use case codes. The relevant details are listed in the table below

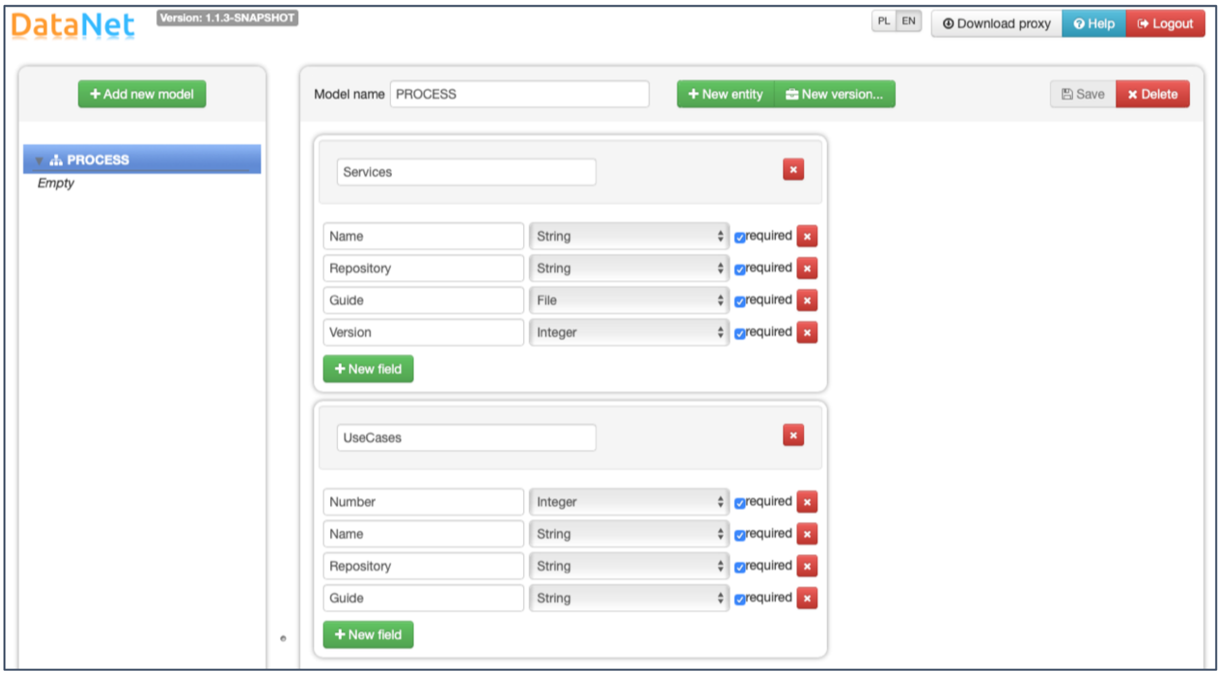
Table 14: Cloudify blueprints – PROCESS final release

Process website	https://www.process-project.eu/software/
Software catalogue	https://software.process-project.eu/software/cloudify-blueprints
Source code repo	https://github.com/process-project/cloudify-blueprints/
User documentation	https://github.com/process-project/cloudify-blueprints/blob/main/docs/Cloudify_integration.pdf
DOI	10.5281/zenodo.4126973
Version	1.0.1
License	Apache-2

2.5 DataNet

DataNet enables lightweight metadata management backed by a flexible database which allows convenient access to the stored objects. One of the main goals of DataNet is to make it usable from the largest set of languages and platforms possible. That is why it uses HTTP as a basis for transferring data between backing servers and the service, and − for improved compatibility and automation− we applied the REST methodology to structure the messages sent to and from the repositories, which makes the integration process straightforward.

DataNet also features in-transit encryption to ensure the security of metadata while being moved between components. The pluggable security mechanism also ensures that access to the platform is restricted to the appropriate group of people to prevent leakage of stored data. DataNet's API enables both, the integration with other PROCESS components such as the IEE portal as well as direct access from external components.



Model name PROCESS

Services

Name	String	required
Repository	String	required
Guide	File	required
Version	Integer	required

UseCases

Number	Integer	required
Name	String	required
Repository	String	required
Guide	String	required

Figure 4: DataNet user interface

Originally, DataNet was developed and tested in the scope of the PLGrid project. DataNET is currently running in production at: https://datanet.plgrid.pl

Table 15: DataNet – PROCESS final release

Process website	https://www.process-project.eu/software/
Software catalogue	https://software.process-project.eu/software/datanet
Source code repo	https://github.com/process-project/datanet
Install documentation	https://github.com/process-project/datanet/blob/master/README.md
User documentation	https://datanet.plgrid.pl/documentation/tutorial/
DOI	10.5281/zenodo.4134169
Version	1.1.3-process
License	MIT
TRL	8

3 PROCESS Service Demonstrator Release

To demonstrate the interface and possibilities of the PROCESS services stack we have created a virtual machine image called *mini-process* which runs (a subset of) the services for demonstration and test purposes. This virtual machine can be used independently of the PROCESS testbed, and does not require the configuration and setup needed to install the services on a local infrastructure. While it does not provide significant compute and storage resources, it does provide an easy-to-use way for potential users to explore the possibilities of the PROCESS services.

Table 16: Mini-process Demonstrator – PROCESS final release

Software catalogue	https://software.process-project.eu/software/mini-process
Source code repo	https://github.com/micro-infrastructure/mini-process
Documentation	https://github.com/micro-infrastructure/mini-process/blob/master/README.md
DOI	10.5281/zenodo.4154490
Version	0.1.0
License	MIT

In addition to the complete virtual machine image offered in *mini-process* the individual services are also available as stand-alone virtual machines. The data service is contained in a *mini-lobcder* release.

Table 17: Mini-lobcder Demonstrator – PROCESS final release

Software catalogue	https://software.process-project.eu/software/mini-lobcder
Source code repo	https://github.com/micro-infrastructure/mini-lobcder
Documentation	https://github.com/micro-infrastructure/mini-lobcder/blob/master/README.md
DOI	10.5281/zenodo.4154714
Version	0.1.0
License	MIT

The compute service and PROCESS web portal is offered in the *mini-IEE* virtual machine.

Table 18: Mini-IEE Demonstator – PROCESS final release

Software catalogue	https://software.process-project.eu/software/mini-iee
Source code repo	https://gitlab.com/cyfronet/mini-iee
Documentation	https://gitlab.com/cyfronet/mini-iee/-/blob/master/README.md
DOI	10.5281/zenodo.4162112
Version	0.9.0
License	MIT

Finally, a *mini-slurm* virtual machine is also provided, which simulates a small SLURM cluster to which jobs can be submitted. It is used by mini-IEE and mini-process as a virtual target for job submission.

Table 19: Mini-slurm Demonstrator – PROCESS final release

Software catalogue	https://software.process-project.eu/software/mini-slurm
Source code repo	https://github.com/micro-infrastructure/mini-slurm
Documentation	https://github.com/micro-infrastructure/mini-slurm/blob/master/README.md
DOI	10.5281/zenodo.4154714
Version	0.1.0
License	MIT

4 Release of use case software, workflows and data

In this section we will describe the releases of source codes, workflows, and orchestration files for the use cases. For each use case, we will provide a brief introduction describing the main outcome of the use case, and provide an overview of all released components. For some of the use cases (UC4 and UC5) not all software can be released as open source, as this software is owned by companies.

For each of the released software components, we have also created an entry in the PROCESS software directory (https://software.process-project.eu/) where more details can be found for each of the software releases, such as the contributors, links to the source code repositories and documentation, a list of related scientific publications, presentation, blogs, videos, deliverables, etc. As this directory will be updated after the release of this deliverable, it may contain more up-to-date information than shown below.

4.1 Use case 1

https://www.process-project.eu/exascale-learning-on-medical-image-data/

The main outcome of UC1 is the development of a three-layered software architecture, CamNet, for training different deep neural network models. The architecture and functionality are described in detail in D1.3. The software for each layer is released separately, as described below. In addition, easy-to-deploy Docker containers are also released for each layer. In the following section, we will provide a short summary of each component, along with all the relevant information on the software release.

4.1.1 CamNet Layer 1

The first layer of the application focuses on the patch extraction and data pre-processing of the WSIs, and was described and evaluated in D4.5.

Table 20: CamNet Layer 1 – PROCESS final release

Software catalogue	https://software.process-project.eu/software/camnet-layer-1
Source code repo	https://github.com/medgift/PROCESS_L1
Documentation	https://github.com/medgift/PROCESS_L1/blob/master/README.md
DOI	10.5281/zenodo.4148054
Version	1.0.0
License	MIT

4.1.2 CamNet Layer 2

This second layer uses the patches and labels extracted by layer 1 to train a convolutional neural network to classify the two patch types. Different models can be chosen by a configuration parameter. Keras and Tensorflow are used to implement this layer.

Table 21: CamNet Layer 2 – PROCESS final release

Software catalogue	https://software.process-project.eu/software/camnet-layer-2
Source code repo	https://github.com/medgift/PROCESS_L2
Documentation	https://github.com/medgift/PROCESS_L2/blob/master/README.md
DOI	10.5281/zenodo.4148064
Version	1.0.0
License	MIT

4.1.3 CamNet Layer 3

The third layer focuses on determining the robustness and interpretability of the convolutional neural network trained by layer 2. This is an important feature, as the physicians need to understand why the network labels certain patches as containing tumors or being clear of tumors.

Table 22: CamNet Layer 3 – PROCESS final release

Software catalogue	https://software.process-project.eu/software/camnet-layer-3
Source code repo	https://github.com/medgift/PROCESS_L3
Documentation	https://github.com/medgift/PROCESS_L3/blob/master/README.md
DOI	10.5281/zenodo.4153991
Version	0.9.0
License	MIT

4.1.4 Regression Concept Vectors (RCVs)

This repository contains the software developed for the interpretability and perturbation robustness for the analysis of histopathology images by convolutional neural networks.

Table 23: Regression Concept Vectors – PROCESS final release

Software catalogue	https://software.process-project.eu/software/rcv
Source code repo	https://github.com/medgift/iMIMIC-RCVs
Documentation	https://medium.com/research-at-medgift/deep-learning-interpretability-for-medical-imaging-regression-concept-vectors-implementation-and-a1bf979ea63a
DOI	10.5281/zenodo.4157223
Version	1.0.0
License	MIT

4.1.5 CamNet Processing Containers

For portability, the CamNET software stack is also provided as a set of docker containers, whose recipe files are gathered in the UC1 Processing containers repository.

Table 24: CamNet processing containers – PROCESS final release

Software catalogue	https://software.process-project.eu/software/camnet-containers
Source code repo	https://github.com/ieggel/process-uc1-integration
Documentation	https://github.com/ieggel/process-uc1-integration/blob/master/README.md
DOI	10.5281/zenodo.4154758
Version	1.0.0
License	MIT

4.2 Use case 2

https://www.process-project.eu/square-kilometre-array-lofar-ska/

The main outcome of UC2 is a workflow that enables "point-and-click" processing of radio astronomy data from the LOFAR Long Term Archive (LTA). The implementation consists of several components, including a user facing web portal, a pipeline execution backend, analysis workflows, and software containers, which are described in more detail below.

4.2.1 LTACAT Web portal

The LTACAT web portal provides access to the accessible observations within the LOFAR LTA. It retrieves an overview of the available observations from a database backend (which implements the technical details of connecting to the LTA servers), and allows the user to search for the desired observations. It also connects to the LTACAT pipeline backend (described below) to select, configure and execute the pipelines on the selected observation data. An initial version of this web portal has been developed as part of the EOSC-pilot project[9]. In turn, this application was based on the FRBCat[10] portal developed in the AA-Alert project[11].

Table 25: LTACAT Web Portal – PROCESS final release

Software catalogue	https://software.process-project.eu/software/lofar-lta-one-click-processing
Source code repo	https://github.com/process-project/ltacat_UC2
Documentation	https://github.com/process-project/ltacat_UC2/README.md
DOI	10.5281/zenodo.4157276
Version	1.0.0
License	Apache 2.0

4.2.2 The LTACAT pipeline backend

The LTACAT pipeline backend provides the functionality to plug in multiple pipeline execution engines which can be used to configure and launch the processing pipelines on selected observations. Currently, two execution engines are supported to run the processing

[9] https://eoscpilot.eu

[10] http://www.frbcat.org

[11] http://www.esciencecenter.nl/project/aa-alert

pipelines. Initially, the *xenon-flow* platform was used to move data and execute workflows on remote compute sites. This was later replaced by an engine that submits pipelines to the REST API of the IEE. Both execution engines are part of this release.

To configure and launch a pipeline, a JSON configuration file specifying the pipeline parameters must be provided. These are released separately below as the *xenon-flow pipeline* and *IEE pipeline.*

Table 26: LTACAT Pipeline backend – PROCESS final release

Software catalogue	https://software.process-project.eu/software/lta-pipeline-backend
Source code repo	https://github.com/process-project/UC2_workflow_api
Documentation	https://github.com/process-project/UC2_workflow_apireadme.md
DOI	10.5281/zenodo.4159301
Version	1.0.0
License	Apache 2.0

4.2.3 CWL Pipeline Definitions

This repository contains a set of processing pipeline descriptions in the Common Workflow Language (CWL) format. These define the processing pipelines needed to run the UC2 analysis workflows in an automated fashion. The CWL workflow definitions contain steps corresponding to the actual LTA data reduction pipeline steps, plus some extra steps needed to create the appropriate directory structure and adapt the configuration files to the context and the selected data.

Table 27: CWL Pipeline definitions – PROCESS final release

Software catalogue	https://software.process-project.eu/software/lta-cwl-pipeline-definitions
Source code repo	https://github.com/process-project/PREFACTOR-XENON-CWL
Documentation	https://github.com/process-project/PREFACTOR-XENON-CWL/README.md
DOI	10.5281/zenodo.4157310
Version	1.0.0
License	Apache 2.0

4.2.4 LTACAT Xenon-flow Pipeline Configuration

This repository contains a data processing pipeline configuration for the LTACAT pipeline backend based on the xenon-flow execution engine. This processing pipeline consists of a configuration file in JSON containing all parameters of the pipeline (which can be adjusted in the LTACAT web portal) and specifies which CWL files should be executed in the LTACAT pipeline tools container (described below). These CWL files can be found in the *CWL Pipeline Definitions* repository described above.

Table 28: Xenon-flow pipeline configuration – PROCESS final release

Software catalogue	https://software.process-project.eu/software/ltacat-xenon-flow-pipeline
Source code repo	https://github.com/process-project/UC2_pipeline
Documentation	https://github.com/process-project/UC2_pipeline/README.md
DOI	10.5281/zenodo.4159296
Version	1.0.1
License	Apache 2.0

4.2.5 LTACAT IEE Pipeline Configuration

This pipeline configuration uses the interactive execution environment (IEE) REST API calls to implement a workflow for calibrating and imaging LTA data as above. It also uses CWL under the hood, which are part of the *CWL pipeline definitions* repository described above.

Table 29: LTACAT IEE Pipeline Configuration – PROCESS final release

Software catalogue	https://software.process-project.eu/software/ltacat-iee-pipeline
Source code repo	https://github.com/process-project/LOFAR_IEE_pipeline
Documentation	https://github.com/process-project/LOFAR_IEE_pipeline/README.md
DOI	10.5281/zenodo.4159299
Version	1.0.1
License	Apache 2.0

4.2.6 LTA Processing Tool Containers

The data processing pipelines run scientific code developed at ASTRON. For portability, these codes are provided within Singularity containers whose recipe files are gathered in the LOFAR tool containers repository.

Table 30: LTA Processing Tool Containers – PROCESS final release

Software catalogue	https://software.process-project.eu/software/lta-tool-containers
Source code repo	https://github.com/process-project/containerisation
Documentation	https://github.com/process-project/containerisation/README.md
DOI	10.5281/zenodo.4157383
Version	1.0.0
License	Apache 2.0

4.2.7 LTACAT Web Portal Containers

This repository contains the necessary Docker files to create a Docker container image for the LTACAT Web Portal. This Docker container provides an easy to use alternative to installing the individual components.

Table 31: LTACAT Web Por – PROCESS final release

Software catalogue	https://software.process-project.eu/software/lofar-lta-one-click-processing-container
Source code repo	https://github.com/process-project/lofar-lta-one-click-processing
Documentation	https://github.com/process-project/lofar-lta-one-click-processing/blob/master/README.md
DOI	10.5281/zenodo.4157393
Version	1.0.0
License	Apache 2.0

4.3 Use case 3

https://www.process-project.eu/supporting-innovation-based-on-global-disaster-risk-data/

The main goal of UC3 is to enable more efficient data management. This was mainly realised within the project by the data services around LOBCDER. Through a containerized portal, it is possible to access and manage data. Thereby, this UC3-Portal container allows access to data stored on any resource location. A running version of the portal providing access to the UNISDR GAR dataset can be found at: https://gar.mnm-team.org

Table 32: GAR Portal – PROCESS final release

Software catalogue	https://software.process-project.eu/software/gar-portal
Source code repo	https://github.com/process-project/UC3-Portal
Documentation	https://github.com/process-project/UC3-Portal/blob/master/README.md
DOI	10.5281/zenodo.4159480
Version	1.0.0
License	MIT

4.4 Use case 4

https://www.process-project.eu/ancillary-pricing-for-airline-revenue-management/

The main outcome of UC4 are: a *generator* for creating ancillary pricing test datasets, an *ancillary data storage,* which is capable of storing the incoming ancillary data in a way that allows easy exploitation, and an *optimisation method* for pricing and revenue management that includes state-of-the-art machine learning methods.

Only a minor part of this work can be released as open software. The released module is responsible for generating arbitrary random test data for having a basis to set up the machine learning algorithms. The architecture and functionality of the different modules are described in detail in D1.3.

Table 33: Ancillary Pricing Data Generator – PROCESS final release

Software catalogue	https://software.process-project.eu/software/ancillary
Source code repo	https://github.com/process-project/UC4-AncillaryPricingDataGenerator
Documentation	https://github.com/process-project/UC4-AncillaryPricingDataGenerator/blob/master/README.md
DOI	10.5281/zenodo.4126773
Version	1.0.0
License	Apache 2.0

4.5 Use case 5

https://www.process-project.eu/agricultural-analysis-based-on-copernicus-data/

The main output of UC5 is a *process wrapper API* used to connect the proprietary PROMET (Processes of Mass and Energy Transfer) modelling tool to the PROCESS services. PROMET links multi-model simulations with observational Copernicus data sets to provide a combined modelling of the macroscale carbon cycle, water and energy inputs, key nutrients in the soil, plant metabolism and human behaviour.

To integrate this (proprietary and closed source) PROMET application with the PROCESS services, the *process wrapper API* was created to enable deployment of a PROMET run from the IEE on a specific target infrastructure. In addition, the configuration of input variables for PROMET can be performed from within the PROCESS portal (IEE), and the stage-out of the output data is performed via LOBCDER.

Table 34: GAR Portal – PROCESS final release

Software catalogue	https://software.process-project.eu/software/papi
Source code repo	https://github.com/process-project/UC5-API
Documentation	https://github.com/process-project/UC5-API/blob/master/README.md
DOI	10.5281/zenodo.4147574
Version	1.0.1
License	MIT

5 Additional Software Releases

In this section we provide information on additional software releases which are part of the final PROCESS release. The software listed here is more generic than the use case specific tools and is not part of the PROCESS service stack. We therefore list this software separately.

5.1 BRANE

Brane (programmable orchestration of applications and networking) provides a programmatic approach to constructing research infrastructures that is intuitive and easy to use, yet is expressive enough to capture and control the entire, distributed, technical stack.

Brane provides a notebook-based environment which can be used to define pipelines and access the PROCESS infrastructure. Its programming model is based on the separation of concerns principle. For each level of the technical stack, and associated roles, different tooling and abstractions are provided. As a result, top-level applications can be written in a domain-specific language by domain scientists, while underlying routines are implemented and optimised by the relevant experts, e.g. software and system engineers.

Table 35: BRANE – PROCESS final release

Software catalogue	https://software.process-project.eu/software/brane
Source code repo	https://github.com/onnovalkering/brane
Documentation	https://onnovalkering.github.io/brane/
DOI	10.5281/zenodo.3890929
Version	1.0.0
License	MIT

5.2 Xenon-flow

Xenon-flow is a service designed to run CWL workflows on remote compute clusters. It provides both a REST API and web frontend which can be used to submit workflows, and a configurable backend which uses *Xenon* for job submission to remote clusters and *cwl-runner* to execute the workflows. Both cwl-runner and the tools invoked by cwl-runner are expected to be pre-installed on the compute cluster or within container images, which are executed. For data staging into and out of the compute cluster, a simple webdav service is provided. Xenon-flow was used in UC#2 as an initial backend to run processing pipelines. It was later replaced by the integration with the REST API of the IEE and the data services offered by LOBCDER.

Table 36: Xenon-flow – PROCESS final release

Software catalogue	https://software.process-project.eu/software/xenon-flow
Source code repo	https://github.com/xenon-middleware/xenon-flow
Documentation	https://github.com/xenon-middleware/xenon-flow/blob/master/README.md
DOI	10.5281/zenodo.4154720
Version	0.4-process
License	Apache 2.0

5.3 Xenon

Many applications use remote storage and compute resources. To do so, they need to include code to interact with the scheduling systems and file transfer protocols used on those remote machines. Unfortunately, many different scheduler systems and file transfer protocols exist, often with completely different programming interfaces. This makes it difficult for applications to switch to a different system or support multiple remote systems simultaneously.

Xenon solves this problem by providing a single programming interface to many different types of remote resources. As a result, changing from one scheduler to another, or from one file transfer protocol to another, becomes a matter of changing just a few lines of code. This is obviously much cheaper in time and money than developing, debugging, and maintaining new code that implements the same logic you had before, but for a different scheduler or a different file transfer protocol.

Xenon was applied in PROCESS as dependency of *xenon-flow* (described above) used in UC#2. This resulted in several small fixes and extensions in Xenon. Note that we directly refer to the entry for Xenon in the software directory of the Netherlands eScience Center

Table 37: Xenon – PROCESS final release

Software catalogue	https://www.research-software.nl/software/xenon
Source code repo	https://github.com/xenon-middleware/xenon
Documentation	https://github.com/xenon-middleware/xenon/blob/master/README.md
DOI	10.5281/zenodo.3719296
Version	3.1.0
License	Apache 2.0

5.4 Research Software Directory

The Research Software Directory is a content management system that is tailored to software. The system is designed to be flexible enough to allow different data sources, database schemas, and so on. By default, the Research Software Directory is set up to collect data from GitHub, Zenodo, Zotero, as well as Medium blogs.

For each software package, a *product page* can be created on the Research Software Directory if the software is deemed useful to others. While the content shown on the product page can be completely customised, by default it includes a *Mentions* section, which can be used to characterise the context in which the software exists. The context may include links to scientific papers, but is certainly broader than that: for example, there may be links to web applications that demonstrate the use of the software, there may be links to videos on YouTube, tutorials on readthedocs.io or Jupyter notebooks, or there may be links to blog posts; really, anything that helps visitors decide if the software could be useful for them.

The Research Software Directory improves findability of software packages, partly because it provides metadata that helps search engines understand what the software is about, but more importantly because of the human centred text snippets that must be provided for each software package.

We have applied the research software directory to provide an overview of the software produced in the PROCESS project. Note that we refer to the entry for the Research Software Directory in the software directory of the Netherlands eScience Center.

Table 38: Research Software Directory – PROCESS final release

Software catalogue	https://www.research-software.nl/software/research-software-directory
Source code repo	https://github.com/process-project/research-software-directory
Documentation	https://github.com/process-project/research-software-directory/blob/master/README.md
DOI	10.5281/zenodo.3876805
Version	2.0.2
License	Apache 2.0

6 Conclusion

In this deliverable we provide an overview of the final release of the PROCESS software. For each of the services developed or modified in the PROCESS project (IEE, RIMROCK, LOBCDER and DataNet), we have created a release which adheres to the FAIR software guidelines, and provided the relevant information on this release in this document. For Cloudify, an externally developed service, we provide the relevant information on the version used, plus the necessary configuration files and service and application blueprints.

In addition, we have also released orchestration files to create a standalone instance of a subset of the PROCESS services for testing and demonstration purposes. This standalone *mini-process* can be used as a starting point by new users of the PROCESS services, as it does not require a complex configuration and setup before use.

We have also released the use case benchmark software and workflows (where possible), and any additional software we have developed or adopted during this project. As for the services, the relevant details are presented in this document, while a more complete (and up-to-date) overview is provided in our online software directory.