D3.5 Workflow manager for generic HPC workloads

Version 1.0

Document Information

Contract Number	823844
Project Website	https://cheese-coe.eu/
Contractual Deadline	30/11/2021 (M37)
Dissemination Level	PU
Nature	OTHER
Author	Christoph Niethammer, Alexey Cheptsov
Contributors	
Reviewers	Giorgio Amati, Piero Lanucara

The ChEESE project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No 823844

Change Log

Version	Description of Change
V0.1	Initial draft for internal review
V0.2	Incorporated internal feedback
V1.0	Final version

Index

1. Introduction

The Centre of Excellence for Exascale in Solid Earth (ChEESE) targets at preparing flagship codes and a series of Pilot Demonstrators (PDs) for their use as services executed on upcoming Exascale computers. The steps necessary to achieve this are split up into several tasks in the project. So, Work Package 4 (WP4) focuses on setting up the Pilot Demonstrators and Work Package 5 (WP5) targets the implementation of services on their basis. These two activities are accompanied by supporting actions around software engineering (WP2) and modelling HPC workflows and tools (WP3).

One of the central aspects of WP3—which this deliverable D3.5 is part of— is to address challenges encountered by the Pilot Demonstrators (WP4) that are related to the system environment, e.g., managing execution workflows across and data transfer between multiple sites. WP3 therefore provides within Task T3.4 a Workflow Management System (WMS) that supports most, if not all, off these requirements.

This deliverable presents the first stable release of the ChEESE HPC Workflow Management System – *WMS-light*. Starting from the requirements specified by the ChEESE Pilot Demonstrators, and the initial Prototype (Deliverable D3.2), the WMS-light has evolved to a general tool that can execute any component-based workflow, also beyond the Geoscience domain. This document accompanies the software provided in this deliverable by giving background information about the WMS and its latest state, release information, as well as a quick introduction how to set up and run *WMS-light*.

2. WMS-light: Main concepts and implementation

2.1. Motivating requirements

Modern HPC applications rely largely on interdisciplinary approaches that require a tight functional integration between diverse software components, implementing the related facets of the *cross-domain applications*. These cross-domain applications consist of different highly specialized and optimized codes to simulate specific aspects of the overall question to be answered. Here results of one simulation code will be used as part of input data for the next code. Geoscience applications, as developed in ChEESE, are a great example of such integrated solutions, e.g., an earthquake simulation may provide input data for a following tsunami simulation.

The integration challenges are addressed by consolidating the related software projects into a common workflow – a component-based application with the automatically managed dependencies (synchronization control, data exchange, etc.) realization, such as simplified by Figure 1.

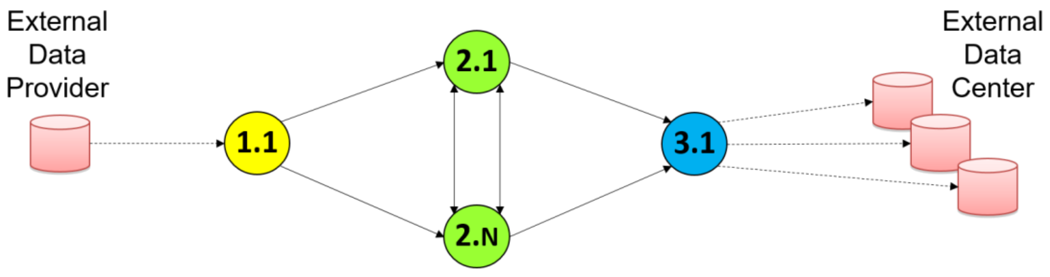


Figure 1: A bird-view of a workflow. Here one simulation (component 1.1) is run using one code which is fed by some external data as input. The output of this simulation is then used by several other codes as input to run N simulations (components 2.1 to 2.N). The output of these simulations then may be combined with another code (component 3.1) and the results are stored in some external data center.

The component-based architecture is of a big advantage when any external data dependencies have to be met (components "1.1" and "3.1" in Figure 1) or several parallel instances have to be executed (components "2.1"-"2.N" in Figure 1). In particular, the component-based architectures are especially beneficial for distributed deplyoments, where some parts of the workflow have to run on different systems, like illustrated in Figure 2.

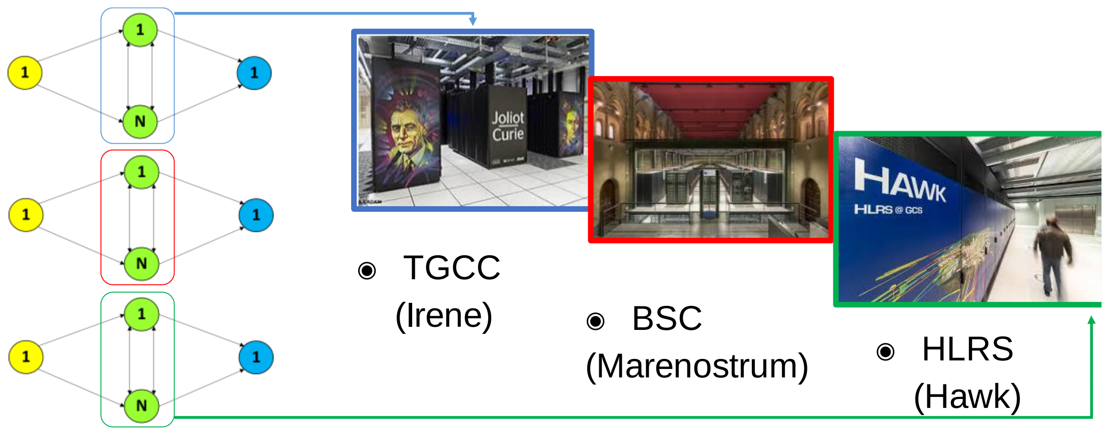


Figure 2: Realization of application studies with heterogeneous workflow deployments.

Unlike in tightly-coupled applications—which normally run on a single (and well-known to the developers) IT-environment—the components of *distributed workflows* have to be executed on heterogeneous infrastructures that span over diverse resources and may even be located at different HPC cites (as depicted in Figure 3). Each resource might have a certain specific way of execution and status monitoring of software. This makes automation of such distributed workflow executions normally a big challenge for component-based applications and introduces the major motivation for having a management system that offers basic support for such consolidated workflows – a *workflow management system* (WMS).

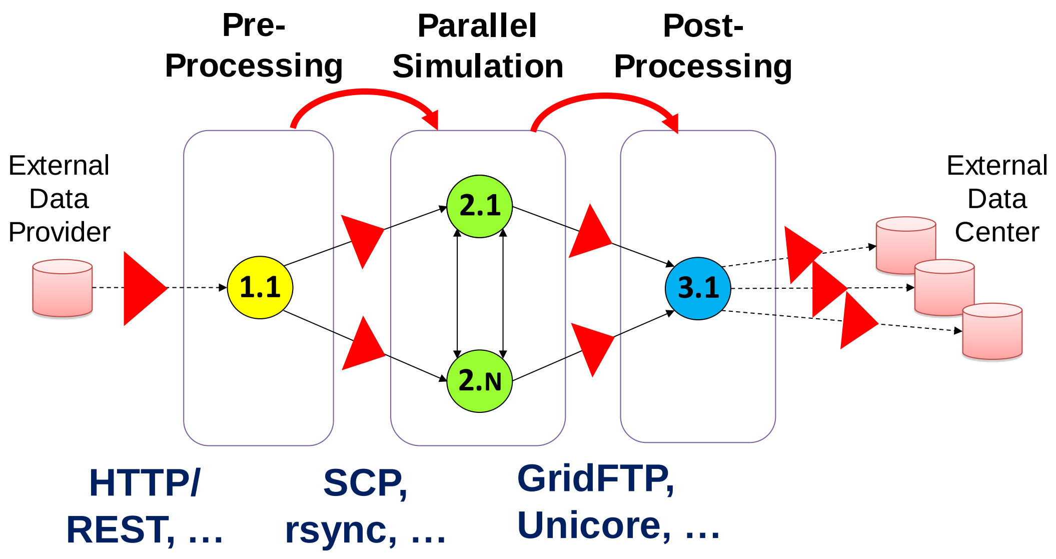


Figure 3: Control- and data-flow realization in distributed workflows. The arrows symbolize the control flow (above) and data exchange dependencies (below) between the component groups deployed on distributed infrastructure.

Along with the control actions, the WMS also has to implement the **data dependencies** between the components as well as the proper data transfer mechanisms like the standard *scp* or *rsync*, the more HPC-specific ones like *GridFTP* etc., but also the specialized protoclos for the communication with the data infrastructure (like *EUDAT [1]*, shown as "external data provided" in Figure 3). The current version of *WMS-light* is implementing the most wide-spread remote connection and communication protocols, i.e., ssh/scp.

In this way, the worklows may run in **heterogeneous deployment** configurations on different infrastructure platforms without any implication on their design. This might be useful for conducting several application studies, each running on a currently available infrastructure platform. The WMS will aim to automatically resolve the issues related to the diverse application execution aspects like access policies, submission mechanisms (such as *PBS/Torque*, *SLURM*, etc.), status monitoring options, potential firewall problems etc.

Some workflows impose a requirement of **dynamic component scaling**, meaning that the number of parallel running entities of a specific component is not known a-priory and will be derived from the dynamic features during the runtime of the workflow components (see Figure 4).

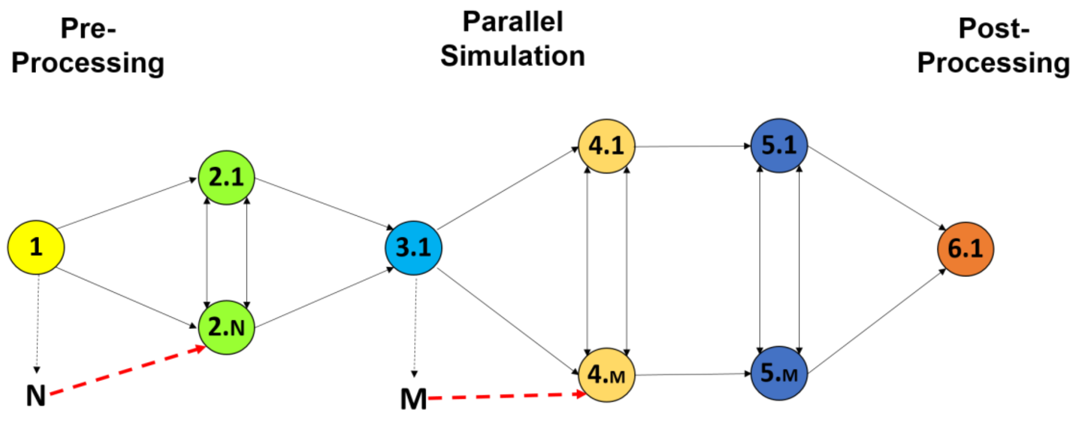


Figure 4: Dynamic workflow component scaling at runtime: Component 1 determines how often Component 2 has to be run (N times) and Component 3 determines the number of executions for the chained Components 4 and 5 (M times), whereas N and M are defined at runtime.

2.2. Distinctive features of WMS-light

The development of the ChEESE WMS solution was largely motivated by two factors that are drastically limiting the abilities of the widespread workflow management solutions (like *Taverna [2]*, *Copernicus [3]*, *Pegasus [4]*, etc.) to support ChEESE applications (see the detailed description in the previous deliverable D3.2[5]):

- Complex programming models that require considerable changes in the way codes are developed. Often code or components have to be compliant with a specific, sophisticated API imposed by the WMS (e.g., the API of *Copernicus-WMS*).
- Need of installing special extension in the system software stack of the targeted infrastructure platform (e.g., quite heavy-weight *Condor* middleware of the *Pegasus-WMS*).
- Necessity of using higher-level design tools to create workflow specification like in *Taverna-WMS*.

In contrast to those solutions, *WMS-light* is built on a light-weight approach that allows to reduce any needed tailoring of the application and/or infrastructure to the absolute minimum while still meeting the major workflow execution requirements (Section 2.1). This is achieved by providing a very-rich-logic functionality at the workflow engine side. *WMS-light* offers a non-intrusive programming model that is capable of meeting the requirement of the most batch-based HPC workflow models.

To the distinctive feature of WMS-light can be referred:

- No changes in the original software are needed to make it workflow-enabled (achieved by the non-declarative execution model, see details in Section 2.4).

- No changes in the system software of the target system (unlike, e.g., in *Pegasus*) are required (thanks to a light-weight service-packs that are provided for the user-space).
- Executable workflow units (i.e., components) can be any user-defined software, such as binaries, bash scripts, other local workflow systems, etc. (in contrast to *Copernicus* or *Taverna*).
- The workflow specification is kept simple and intuitive for the users/developers (based on the JSON format).
- Rich monitoring functionality and interfaces are provided.

2.3. Adaptation for generic HPC workflows and final design

Since the initial prototype (see deliverable D3.2), the *WMS-light* architecture has seen some updates, aiming to provide a better support of the identified pilot use cases but also to address more generic workflow functionalities than the ones necessary in the geoscience community, thus extending the potential application area of *WMS-light*. Here, the initial design from the prototype proved already to be a good basis.

In particular, the following actions have been completed:

- The infrastructure-related functions, such as job submission, status monitoring, etc. were implemented in a set of scripts that can easily be ported to any HPC infrastructure – the *service-pack*. Service packs were implemented for the major PRACE centres, such as CINECA, BSC, TGCC, etc.
- Support of elastic, horizontally-scalable workflow components with dynamically (at runtime) identified number of parallel instances was added.
- Extended monitoring (with more metrics such as start and end time, duration, etc.) with different levels of granularity (workflow- and component-level) was provided.
- Security was addressed in a special component that allows encryption of the sensitive user data.

The final architecture of *WMS-light* can be seen in Figure 5. The core of WMS-light is developed in the *Java* language, thus allowing portability to a wide range of host systems. As an integrated data layer *Elasticsearch* is used to keep all the workflow-specific information, both static (prior to the execution) and dynamic (occurring at runtime) in one place. The distributed realization of *Elasticsearch* helps dealing for many-components executions, when the service data are generated in many parallel streams. A rich set of RESTful web-services build on top of the data layer enable flexible communication with it not only using the steering *WMS-light*'s Java components, but also from a rich set of the interfaces exposed to the users. The infrastructure-specific operations like job submission, execution status monitoring, etc. are accomplished by means of a set of scripts, also installed on the target infrastructure as a service pack.

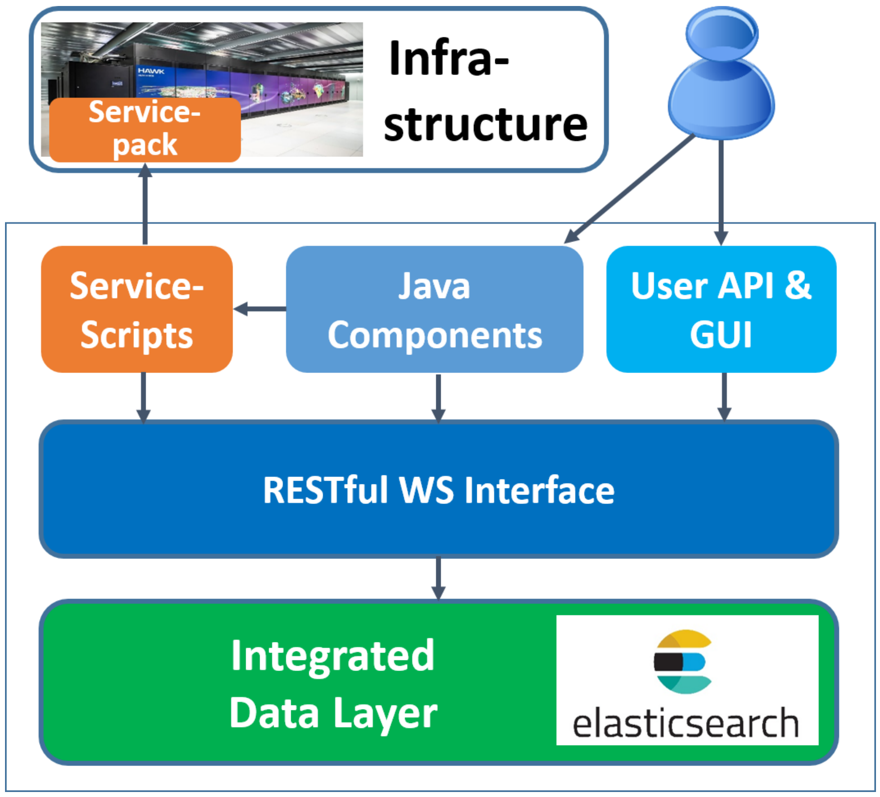


Figure 5: WMS-light architecture.

2.4. Major concepts and specifications

A typical WMS-light application can be illustrated by a workflow that was developed for the PD6 use case, see Figure 6. It consists of 7 components, two of which (Fall3D-*) are supposed to run on an external HPC system. All components are wrapped by *bash* scripts that prepare and run the component-specific code (largely implemented in *python*).

This workflow life-cycle is depicted in Figure 7 and detailed in deliverable D3.2. Here we provide some basic examples of the specifications aiming to better understand the underlying concepts.

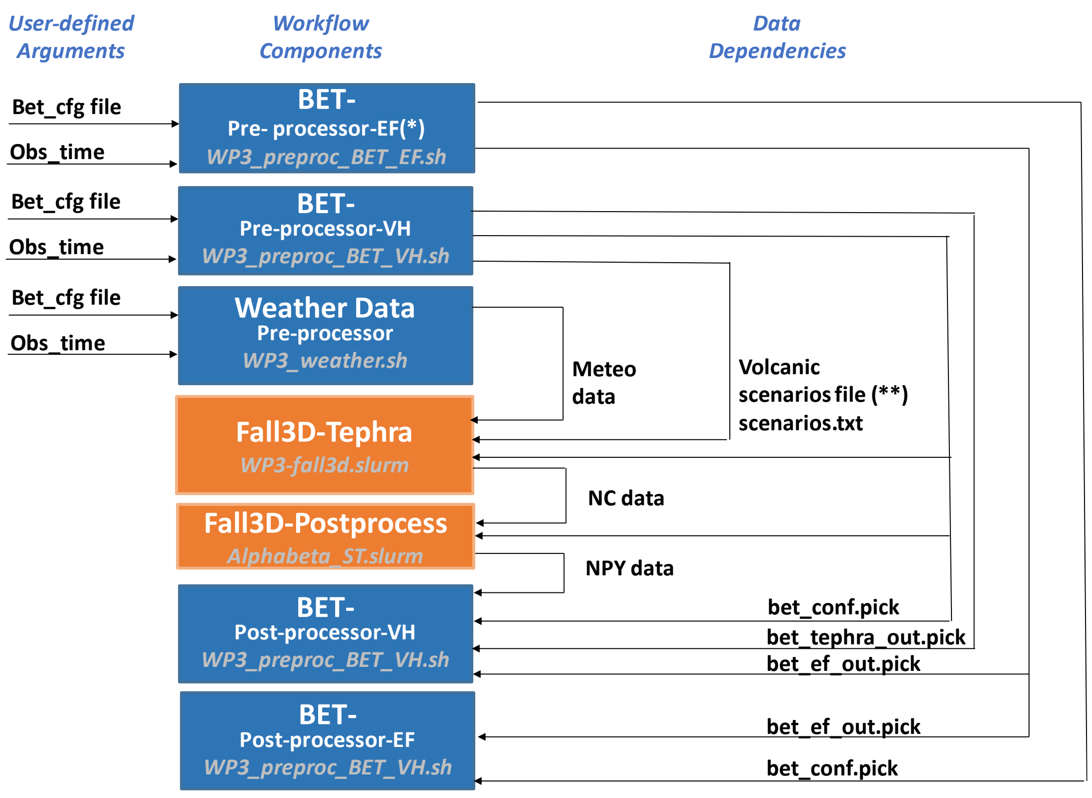


Figure 6: WMS-light workflow example: PD8.

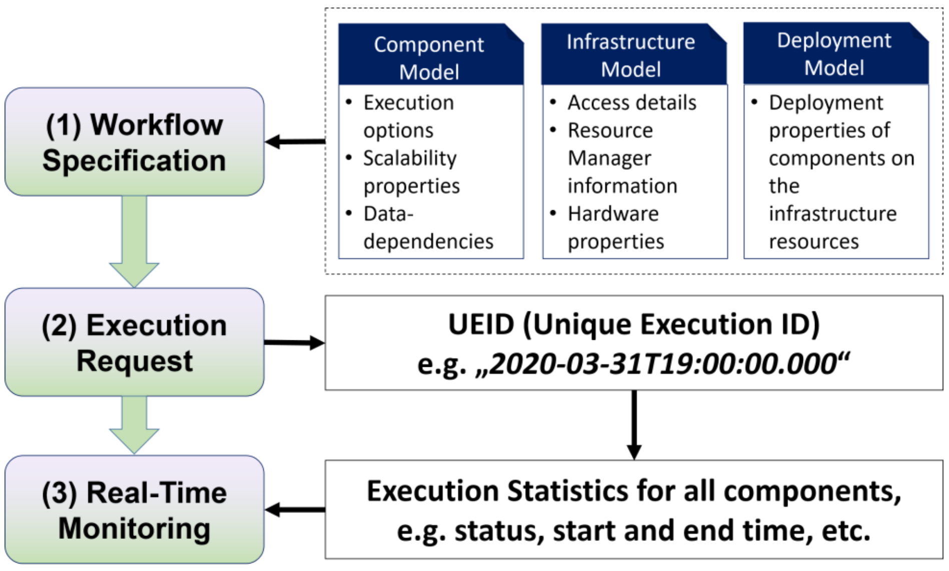
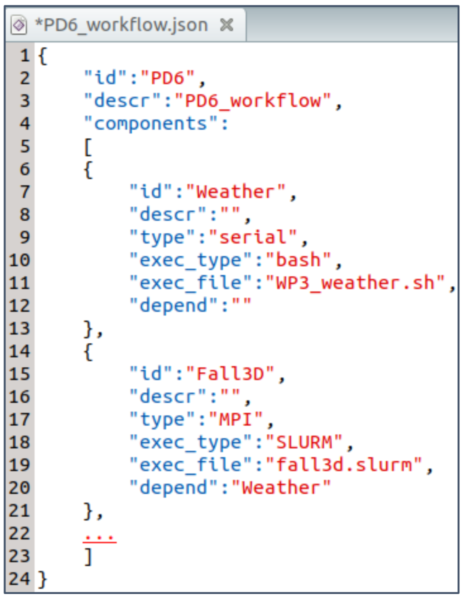

(1) Workflow Specification
(2) Execution Request
(3) Real-Time Monitoring
Component Model
• Execution options
• Scalability properties
• Data-dependencies
Infrastructure Model
• Access details
• Resource Manager information
• Hardware properties
Deployment Model
• Deployment properties of components on the infrastructure resources
UEID (Unique Execution ID)
e.g. „2020-03-31T19:00:00.000“
Execution Statistics for all components, e.g. status, start and end time, etc.

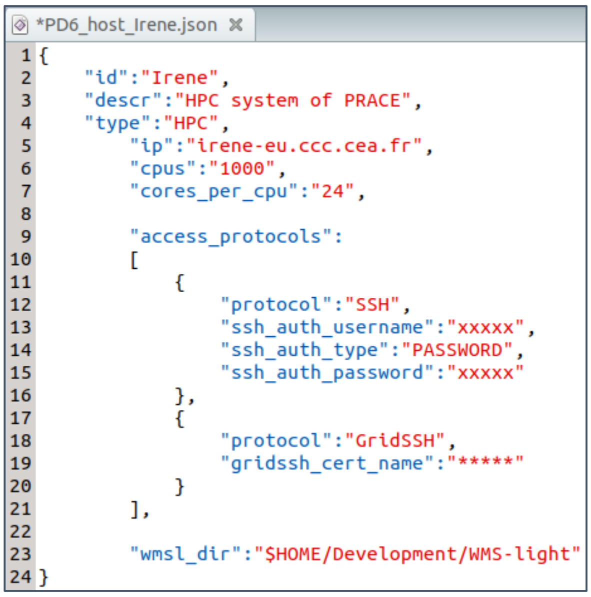
Figure 7: WMS-light workflow life-cycle.

The component model is a JSON-format-based description of the components and their dependencies, exemplified in Figure 8a for the PD6 application (Figure 6). The infrastructure model contains details of the infrastructure platforms, as shown in Figure 8b for the HPC system of TGCC (Irene). The deployment model specifies the deployment options of the workflow components on the infrastructure platforms (Figure 8c).



```
{
    "id":"PD6",
    "descr":"PD6_workflow",
    "components":
    [
    {
        "id":"Weather",
        "descr":"",
        "type":"serial",
        "exec_type":"bash",
        "exec_file":"WP3_weather.sh",
        "depend":""
    },
    {
        "id":"Fall3D",
        "descr":"",
        "type":"MPI",
        "exec_type":"SLURM",
        "exec_file":"fall3d.slurm",
        "depend":"Weather"
    },
    ...
    ]
}
```

(a)

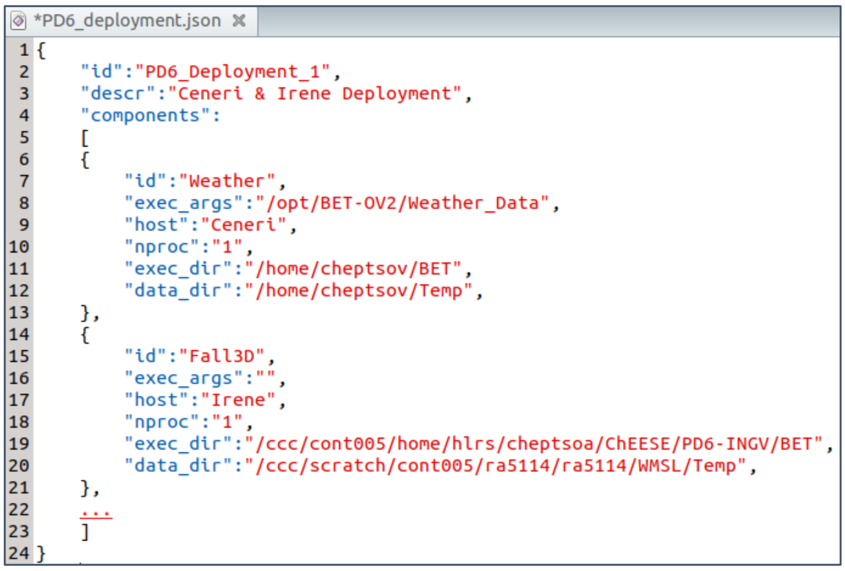


```
{
    "id":"Irene",
    "descr":"HPC system of PRACE",
    "type":"HPC",
        "ip":"irene-eu.ccc.cea.fr",
        "cpus":"1000",
        "cores_per_cpu":"24",

        "access_protocols":
        [
            {
                "protocol":"SSH",
                "ssh_auth_username":"xxxxx",
                "ssh_auth_type":"PASSWORD",
                "ssh_auth_password":"xxxxx"
            },
            {
                "protocol":"GridSSH",
                "gridssh_cert_name":"*****"
            }
        ],

        "wmsl_dir":"$HOME/Development/WMS-light"
}
```

(b)



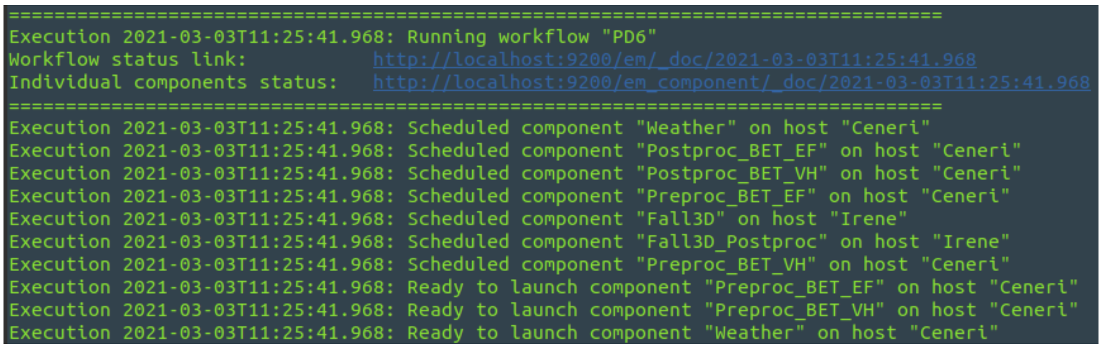
```
{
    "id":"PD6_Deployment_1",
    "descr":"Ceneri & Irene Deployment",
    "components":
    [
    {
        "id":"Weather",
        "exec_args":"/opt/BET-OV2/Weather_Data",
        "host":"Ceneri",
        "nproc":"1",
        "exec_dir":"/home/cheptsov/BET",
        "data_dir":"/home/cheptsov/Temp",
    },
    {
        "id":"Fall3D",
        "exec_args":"",
        "host":"Irene",
        "nproc":"1",
        "exec_dir":"/ccc/cont005/home/hlrs/cheptsoa/ChEESE/PD6-INGV/BET",
        "data_dir":"/ccc/scratch/cont005/ra5114/ra5114/WMSL/Temp",
    },
    ...
    ]
}
```

(c)

Figure 8: Example of basic workflow specifications taken from an implemented ChEESE-PD6 use case: a) Component Model, b) Infrastructure Model, c) Deployment Model

For storing security-crucial workflow specification data like account names and passwords, *WMS-light* provides a separate Security Component that implements a *PBKDF2WithHmacSHA1* based encryption of the sensitive user data.

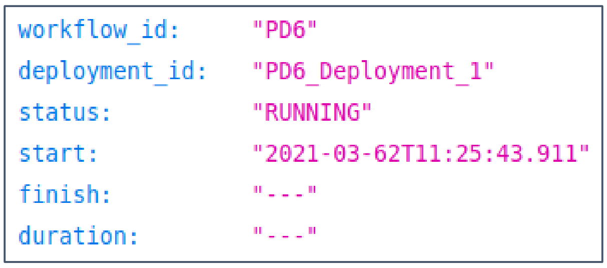
The workflow execution requests of the users are handled by the WMS-light runtime environment, managing a full set of the monitoring and steering functionalities during the workflow instance execution (see Figure 9).



```
================================================================================
Execution 2021-03-03T11:25:41.968: Running workflow "PD6"
Workflow status link:           http://localhost:9200/em/_doc/2021-03-03T11:25:41.968
Individual components status:   http://localhost:9200/em_component/_doc/2021-03-03T11:25:41.968
================================================================================
Execution 2021-03-03T11:25:41.968: Scheduled component "Weather" on host "Ceneri"
Execution 2021-03-03T11:25:41.968: Scheduled component "Postproc_BET_EF" on host "Ceneri"
Execution 2021-03-03T11:25:41.968: Scheduled component "Postproc_BET_VH" on host "Ceneri"
Execution 2021-03-03T11:25:41.968: Scheduled component "Preproc_BET_EF" on host "Ceneri"
Execution 2021-03-03T11:25:41.968: Scheduled component "Fall3D" on host "Irene"
Execution 2021-03-03T11:25:41.968: Scheduled component "Fall3D_Postproc" on host "Irene"
Execution 2021-03-03T11:25:41.968: Scheduled component "Preproc_BET_VH" on host "Ceneri"
Execution 2021-03-03T11:25:41.968: Ready to launch component "Preproc_BET_EF" on host "Ceneri"
Execution 2021-03-03T11:25:41.968: Ready to launch component "Preproc_BET_VH" on host "Ceneri"
Execution 2021-03-03T11:25:41.968: Ready to launch component "Weather" on host "Ceneri"
```

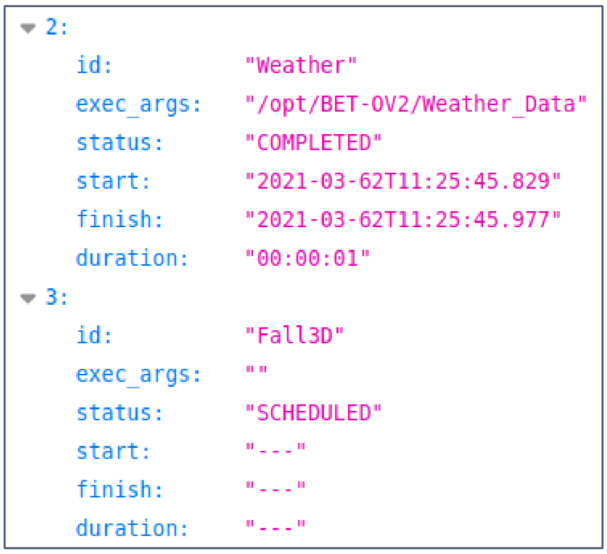
Figure 9: Running a new workflow instance with WMS-light.

The control flow between the workflow components is enforced by tracking the components status (*SHEDULED*, *SUBMITTED*, *RUNNING*, *COMPLETED*) by means of the WMS-light's core java components and services, managed via the Data Layer and made available to the user interfaces via the RESTful service end-points, created for every unique execution ID (UEID) of the workflow execution. Such information is available at different granularity levels, such as the entire workflow (Figure 10a) or for individual components (Figure 10b).



```
workflow_id:      "PD6"
deployment_id:    "PD6_Deployment_1"
status:           "RUNNING"
start:            "2021-03-62T11:25:43.911"
finish:           "---"
duration:         "---"
```

(a)



```
▼ 2:
    id:           "Weather"
    exec_args:    "/opt/BET-OV2/Weather_Data"
    status:       "COMPLETED"
    start:        "2021-03-62T11:25:45.829"
    finish:       "2021-03-62T11:25:45.977"
    duration:     "00:00:01"
▼ 3:
    id:           "Fall3D"
    exec_args:    ""
    status:       "SCHEDULED"
    start:        "---"
    finish:       "---"
    duration:     "---"
```

(b)

Figure 10: Live monitoring information: a) Entire workflow, b) individual components.

3. Release information

3.1. New features since the previous release

Release 1.0 of WMS-light includes a considerably improved set of the basic components (such as Resource Manager, Execution Manager etc). and implements a set of features, the most important among those are:

- Dynamic scalability support of the parametrized applications ("parametric" value of the "type" field of the Component Model specification)
- Security manager for keeping the sensible data (like the passwords for the platforms in the Infrastructure Model specification)
- Better support of SLURM-based job management system.

3.2. Released packages content

The following table summarizes the release content details.

Component	Description
Source code release	Contains: - Run-time environment - Dependencies libraries (binaries) - Java code of all major components (Execution Manager, Resource Manager, etc.) - Security manager - Basic documentation Download: https://fs.hlrs.de/projects/cheese/releases/wms-light-1.0.0.tgz
HPC service pack	Contains: - Scripts to be installed in the user space of the targeted HPC platform Download: https://fs.hlrs.de/projects/cheese/releases/wms-light-servpack-1.0.0.tgz
Dockerfile	Contains: - Docker file with recipes for automated building Download: https://fs.hlrs.de/projects/cheese/releases/Dockerfile

3.3. Licensing information

All the WMS-light software components are released under the Apache 2.0 licence, which allows for a free use, redistribution and development of the whole codebase. The table below summarizes the software requirements of the libraries, used internally by WMS-light.

Library	Version	Description	Licence
Elasticsearch	7.6.1	Serves the Data Layer functionality	Elastic License
SSHJ	0.29.1	Serves ssh-based communication functionality	Apache 2.0.

4. Quick-start-guide with the Docker container

Using a Docker container is the easiest method to start to try the WMS-light software out.

The major prerequisites are:

- A Docker installation (requires sudo permisssions)
- About 2GB of the free disk space

In order to start with the Docker container, download the Dockerfile (see Section 3.1), e.g., with the Linux command:

```
wget https://fs.hlrs.de/projects/cheese/Dockerfile
```

to a dedicated folder and then launch the following command to build a Docker image:

```
sudo docker build --rm -t wmslight:1.0 .
```

A container instance can be then run with the following command:

```
sudo docker run --privileged  -ti -e container=docker  \
        -v /sys/fs/cgroup:/sys/fs/cgroup  wmslight:1.0
```

The non-docker "from scratch" installation guides are provided in the README file included in the release.

References

[1] EUDAT - EUROPEAN OPEN SCIENCE CLOUD FOR RESEARCH. Position paper: https://www.eudat.eu/sites/default/files/PositionPaperEOSCcard.pdf
[2] K. Wolstencroft et al. The Taverna workflow suite: designing and executing workflows of Web Services on the desktop, web or in the cloud. Nucleic Acids Res. 2013 Jul; 41(Web Server issue): W557–W561. Published online 2013 May 2.
[3] Copernicus Workflow Management System - Documentation. https://copernicus.readthedocs.io
[4] Ferreira da Silva et al. Pegasus - Community Resources for Enabling Research in Distributed Scientific Workflows. 10th IEEE International Conference on e-Science, Oct. 2014
[5] ChEESE Deliverable D3.2 - Worfklow manager prototype