H2020-EINFRA-2017

EINFRA-21-2017 - Platform-driven e-infrastructure innovation

DARE [777413] "Delivering Agile Research Excellence on European e-Infrastructures"

D3.7 - Integrated Monitoring and Management Tools I

Project Reference No	777413 — DARE — H2020-EINFRA-2017 / EINFRA-21-2017
Deliverable	D-3.7 Integrated Monitoring and Management Tools M12
Work package	WP3: Large-scale Lineage and Process Management
Tasks involved	T3.2, T3.3
Type	DEM: Demonstrator, pilot, prototype
Dissemination Level	PU = Public
Due Date	31/12/2018
Submission Date	31/12/2018
Status	Final – v5
Editor(s)	Alessandro Spinuso (KNMI)
Contributor(s)	Iraklis Klampanos (NCSRD)
Reviewer(s)	Ioannis Foufoulas (ATHENA)
Document description	Overview of the first release of the Monitoring and Management tools and services.

Document Revision History

Version	Date	Modifications Introduced	
		Modification Reason	Modified by
1	20/11/2018	Monitoring Tools in the first DARE release.	Alessandro Spinuso
2	28/11/2018	Dispel4py PEs registry	Iraklis Klampanos
3	28/11/2018	Added sections 1.1 and Conclusions	Alessandro Spinuso
4	29/12/2018	Sections re-organisation	Alessandro Spinuso
5	31/12/2018	Internal review	Ioannis Foufoulas
6	31/12/2018	Finalisation	Iraklis Klampanos

Table of Contents

List of Terms and Abbreviations

Abbreviation	Definition
MVV	Monitoring and Validation Visualiser
BDV	Bulk Dependencies Visualiser
PROV	W3C Standard for Provenance Representation
S-PROV	PROV extension for lineage representation of of streaming operators.

1 Introduction

This is a short report on the Monitoring and Management tools included in the first deployment of the DARE platform. The lineage exploration tools have been improved with additional search capabilities that take into account the conceptual contextualisation of data and processes. The underlying service layer is based on a REST API. This has been described in an internal deliverable (ID-3.2) and it is currently going under further improvements. The API will be thoroughly illustrated in the Data Lineage Service deliverable at M18. The document reports also the preliminary implementation of a workflow components' registry, which is integrated with the dispel4py library.

1.1 Approach and relationship with other Work Packages and Deliverables

In alignment with the architecture's principles expressed in D2.1, the lineage queries have been extended with the support of high level *Concepts,* as additional search parameter*. Concepts'* belong to a certain domain and application namespace and further describe methods and data. Their definition will be managed by the DARE API and Catalogues.

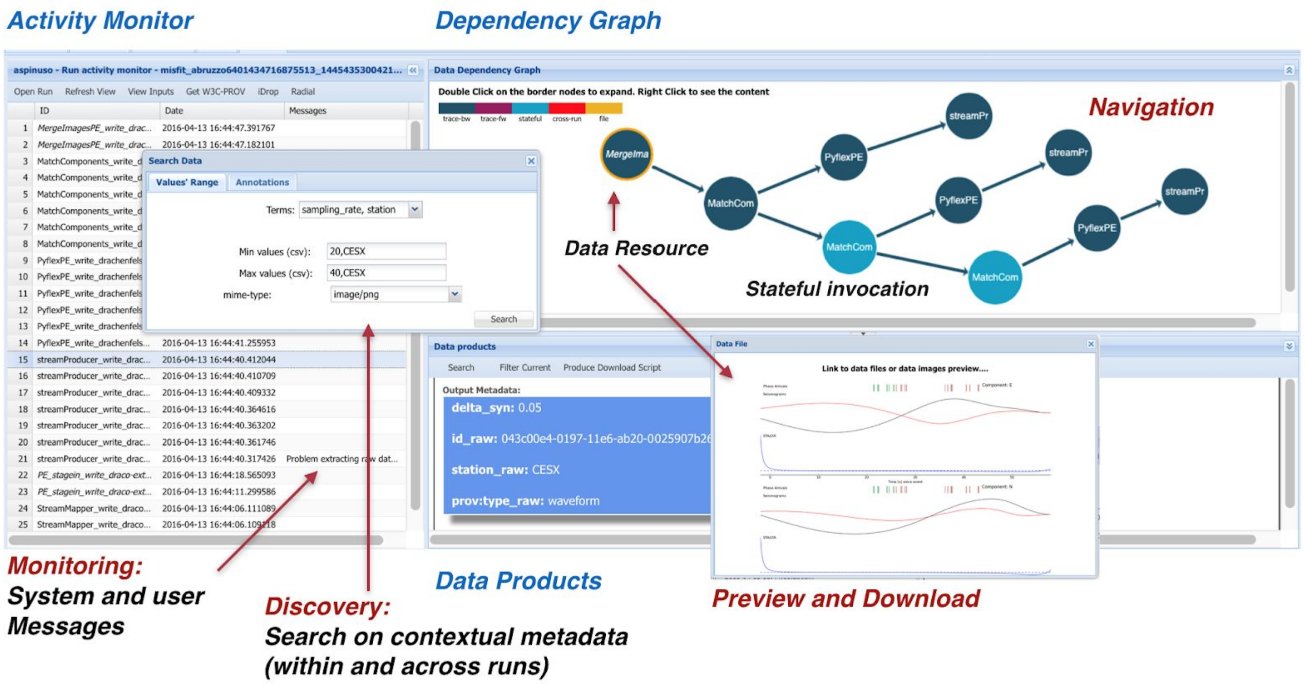
In cooperation with WP5, the containerisation of the *S-ProvFlow* system, offering the tools described in this report, has been improved to allow a sound deployment within the DARE test-bed. This will also facilitate the realisation of a continuous integration solution.

The population of the *S-ProvFlow* repository of workflows' execution lineage is performed through the *dispel4py.provenance* module. It offers the developers the possibility of implementing *Provenance Types* and using them within a *Provenance Configuration* for a particular execution of a workflow. This is described in D3.5 as part of the current DARE API.

The Processing Element Registry is developed in cooperation with WP2 and WP4 and it will be further extended to cover the architectural principles, according to agreed priorities.

2 MVV - Monitoring and Validation Visualiser

The *S-ProvFlow* system offers a visual tool (Monitoring and Validation Visualiser - MVV), that enables different sorts of operations through the interactive access and manipulation of the lineage information. These include monitoring of the progress of the execution, discovery of data and runs, filtering, data preview, download and staging. Below we cover each of these aspects separately. The visual components of the tool illustrated in the following sections are shown in Figure 1.



aspinuso - Run activity monitor - misfit_abruzzo6401434716875513_1445435300421...

	ID	Date	Messages
1	*MergeImagesPE_write_drac...*	2016-04-13 16:44:47.391767	
2	*MergeImagesPE_write_drac...*	2016-04-13 16:44:47.182101	
3	MatchComponents_write_d		
4	MatchComponents_write_d		
5	MatchComponents_write_d		
6	MatchComponents_write_d		
7	MatchComponents_write_d		
8	MatchComponents_write_d		
9	PyflexPE_write_drachenfels		
10	PyflexPE_write_drachenfels		
11	PyflexPE_write_drachenfels		
12	PyflexPE_write_drachenfels		
13	PyflexPE_write_drachenfels		
14	PyflexPE_write_drachenfels...	2016-04-13 16:44:41.255953	
15	streamProducer_write_drac...	2016-04-13 16:44:40.412044	
16	streamProducer_write_drac...	2016-04-13 16:44:40.410709	
17	streamProducer_write_drac...	2016-04-13 16:44:40.409332	
18	streamProducer_write_drac...	2016-04-13 16:44:40.364616	
19	streamProducer_write_drac...	2016-04-13 16:44:40.363202	
20	streamProducer_write_drac...	2016-04-13 16:44:40.361746	
21	streamProducer_write_drac...	2016-04-13 16:44:40.317426	Problem extracting raw dat...
22	*PE_stagein_write_draco-ext...*	2016-04-13 16:44:18.565093	
23	*PE_stagein_write_draco-ext...*	2016-04-13 16:44:11.299586	
24	StreamMapper_write_draco...	2016-04-13 16:44:06.111089	
25	StreamMapper_write_draco...	2016-04-13 16:44:06.109218	

Search Data

Values' Range | Annotations

Terms: sampling_rate, station
Min values (csv): 20,CESX
Max values (csv): 40,CESX
mime-type: image/png

Double Click on the border nodes to expand. Right Click to see the content

Data products

Output Metadata:
delta_syn: 0.05
id_raw: 043c00e4-0197-11e6-ab20-0025907b26
station_raw: CESX
prov:type_raw: waveform

Link to data files or data images preview....

Figure 1: Combined access to a workflow execution's lineage. The **Activity Monitor** shows the list of active processes performing the workflow activities. It displays feedback messages, quantity of data produced, last event's timestamp and whether any change occurred in the process implementation or parametrisation. The **Data Dependencies Graph** offers interactive access to data dependencies. Yellow circles indicate that the provenance entity links to a concrete data resource such a file or an image. Arrows represent the *wasDerivedFrom* relationship (in PROV term) between data. Metadata are visible in the **Data Products** panel. The interface also allows the selective export of the lineage to PROV compliant formats, according to the S-PROV ontology (an extension of PROV for data-intensive provenance), available at: https://gitlab.com/project-dare/s-ProvFlow/blob/master/resources/s-prov-o.owl

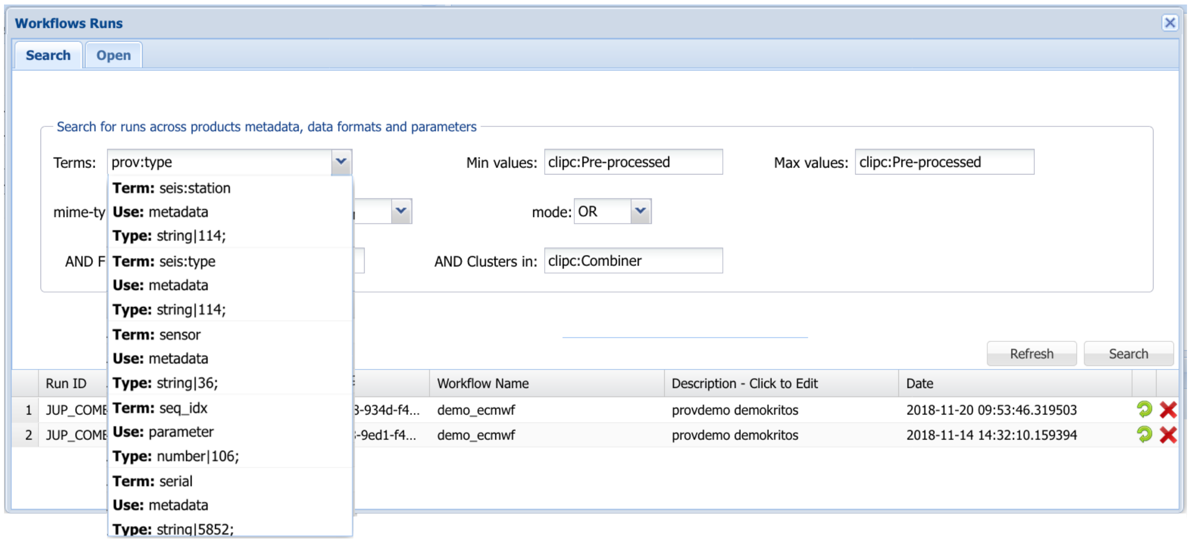
2.1 Monitoring

The **Activity Monitor**, displays the activity of the workflow after it has been mapped to the target resource. The view can be updated at runtime and its content is dynamically fetched from the underlying Lineage API, according to the user's position within the scrollable area. This allows users to easily browse through the items.

The list shows the timestamp of the most recent invocation of a workflow's process instance, the count of the data produced, occurrence of runtime messages, such as errors, warnings or special textual annotations coming from the live computation. Runtime changes affecting the instances are highlighted reporting their total count. Clicking on one of the list's items loads the information of the generated data in the **Data Products** panel, for visualisation and further operations, as we will describe in the sections below.

2.2 Discovery of Experiments and Data

Users can search for workflow executions and data elements adopting terms which refer to standard vocabularies, as well as experimental terms introduced by specific application's and researchers' requirements. Searching for workflows' executions, Figure 2, allows them to configure the MVV tool for the exploration of a specific workflow run. Once the run is selected and the MVV prepared to access its provenance, users can search for data in the **Data Products** panel or apply filters on the data already listed. Here, each data product is described by its metadata and the information about its generating process. The selection of the terms for the searching is assisted through hints (see Figure 2). These are suggested among the terms introduced by the user or research-developer when preparing the workflow for provenance extraction (see D3.5).


Workflows Runs

Search | Open

Search for runs across products metadata, data formats and parameters

Terms: prov:type

Min values: clipc:Pre-processed

Max values: clipc:Pre-processed

mime-ty

mode: OR

AND F

AND Clusters in: clipc:Combiner

Term: seis:station
Use: metadata
Type: string|114;
Term: seis:type
Use: metadata
Type: string|114;
Term: sensor
Use: metadata
Type: string|36;
Term: seq_idx
Use: parameter
Type: number|106;
Term: serial
Use: metadata
Type: string|5852;

Refresh | Search

	Run ID		Workflow Name	Description - Click to Edit	Date
1	JUP_COME	3-934d-f4...	demo_ecmwf	provdemo demokritos	2018-11-20 09:53:46.319503
2	JUP_COME	-9ed1-f4...	demo_ecmwf	provdemo demokritos	2018-11-14 14:32:10.159394

Figure 2: Workflows' Runs search panel. This panel is used to discover runs of interest based on parameters and metadata values-ranges and on *Concepts* defined in the application domain namespace. As shown in the example, *Concepts* are expressed as values of the *prov:type,* for the data produced (*clipc:Pre-processed*). and to characterise the belonging of workflow components to conceptual Clusters (*clipc:Combiner*). In the dropdown list, we show that metadata query terms can be either qualified (*seis:station*, *seis:magnitude*) or experimental (*seq_idx*). The terms are further described by their Use (parameters or metadata), their primitive type and the total count of their occurrence.

2.3 Data Dependencies Navigation

The Data elements returned by a search or a filter are examined within the **Data Products** panel. Each data element can be analysed in detail, including the possibility to start the interactive exploration of the data derivations, which is then performed within the **Data Dependencies Graph** panel. The navigation is controlled by clicking on each data node. Users can configure the depth of each step to rapidly expand the whole graph, which can span across multiple workflows execution, showing evidence of the reuse of data produced in different *sessions*.

2.4 Preview and Staging

The **Data Products** panel allows users to gain detailed information on the workflow data and to invoke operations on the data and its provenance. For stream-based data-intensive analysis, which is the main focus of this tool, the data does not always correspond to a concrete file or resource. Especially in the intermediate phases of a computation, data is volatile in most of the cases, therefore only described by its domain and processing metadata. However, when the data is materialised, the tool facilitates transfer operations across infrastructures, by generating staging scripts based on the data products location. This can be applied to all data listed in the **Data Products** panel. Once the script is produced its execution can be performed remotely (currently in GridFTP) by referencing the active user certificate. This solution requires the underlying infrastructures to support delegation of data transfer services, according to the granted authorisation credentials. Whether interoperable data-management systems offered by the EOSC and EUDAT CDI, suh has B2STAGE or B2DROP, also allow delegation, these could be easily integrated in our tool.

3 BDV - Bulk Dependencies Visualiser

The interactive features presented by the MVV are designed for in-depth analysis of large workflow runs. To obtain comprehensive views for a single workflow execution or involving many runs and users, we offer a different tool, the Bulk Dependencies Visualiser (BDV). It implements an approach to visual analytics of the information captured by the S-PROV model that exploits radial diagrams, combined with the Edge Bundles technique. The Bulk Dependencies Visualisers offers comprehensive views of the provenance repository at multiple levels of granularity and for different kinds of expertise and roles. It offers facilities to tune and organise the views.

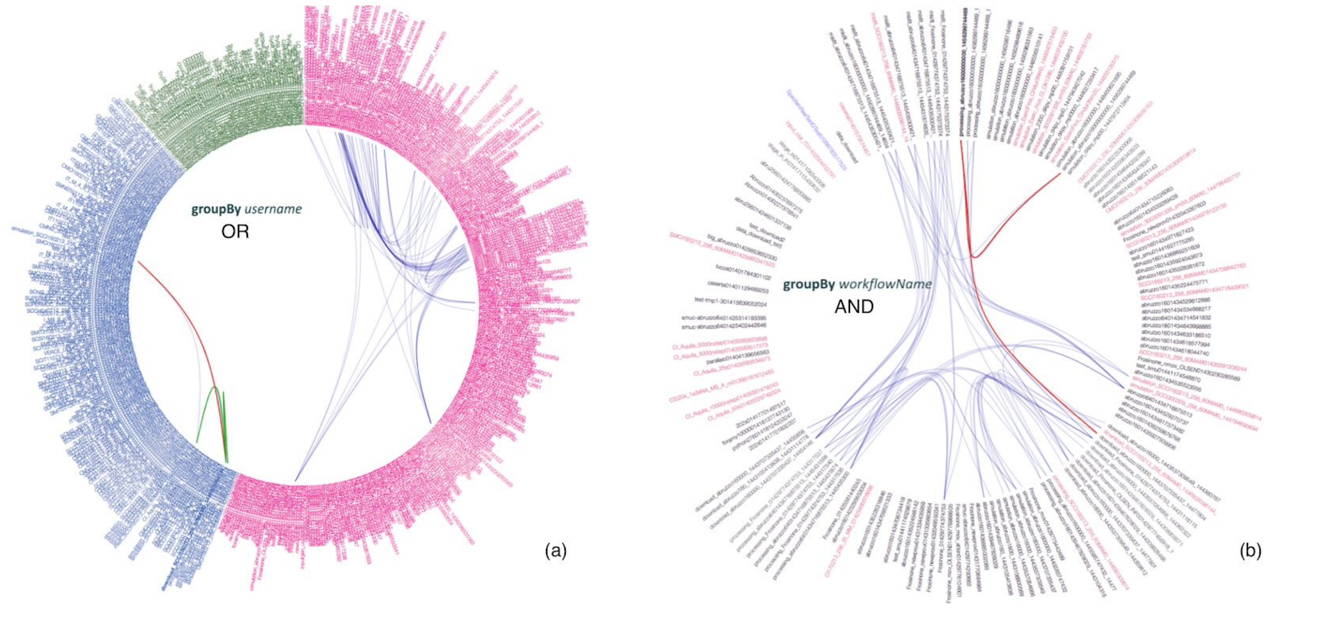
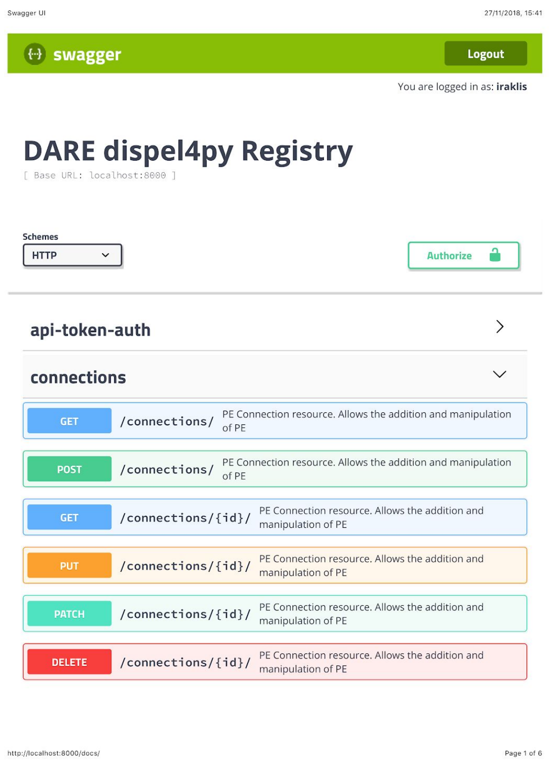


Figure 3: BDV - collaborative interactions among users and workflows. The diagrams are obtained by searching for runs that involved data that have metadata values within specific ranges and by applying different grouping rules: (a) by *username* and (b) by *workflowName*. Vertices represent workflows' run ids and edges indicate whether any data has been exchanged among runs (green and red edges represents respectively output and input). Vertices' colour are associated with the user that executed the workflow. The image (b) is obtained by requesting the logical conjunction (AND) of the metadata values-ranges describing the workflows' data.

We consider two classes of usage. In a first scenario the visual analytics approach can produce views related to a single run, while in a second scenario, see Figure 3, users can visually explore the interactions and data-reuse between users and workflows. Groping rules can organise the views in clusters according, for instance, to the computational resources involved or according to other entities and properties of the S-PROV model. The implementation of the BDV is realised using D3.js, a programmable visualisation toolkit. The functionalities and usability of this experimental tool will be refined in alignment to the DARE use cases. At this stage we have implemented the support of creating visual cluster by *Concept, according to the* Architecture's specifications.

4 Processing Elements Registry Interface

The dispel4py processing-elements (PE) registry[1] is a catalogue detailing the PEs (and workflows, due to the composability of simple into more complex PEs) available to the users. The registry is designed to store descriptions, typing information and implementation details, and it can be dynamically made use of by the dispel4py system. The catalogue is exposed to the users primarily via a REST interface, however, it also exposes an automatically generated graphical user interface which provides a unified testing and documentation solution. Users can familiarise themselves with the REST API using the GUI, while they can also use it to interrogate the live registry. Part of this GUI is appended at the end of this document. The API exposed by the PEs Registry is described in more detail in deliverable D3.5. in Figure 4, we show the *swagger-ui*[2] Interface for the API, automatically generated from its Swagger descriptor.


Swagger UI 27/11/2018, 15:41

swagger Logout

You are logged in as: **iraklis**

DARE dispel4py Registry

[Base URL: localhost:8000]

Schemes HTTP

Authorize

api-token-auth

connections

GET /connections/ PE Connection resource. Allows the addition and manipulation of PE

POST /connections/ PE Connection resource. Allows the addition and manipulation of PE

GET /connections/{id}/ PE Connection resource. Allows the addition and manipulation of PE

PUT /connections/{id}/ PE Connection resource. Allows the addition and manipulation of PE

PATCH /connections/{id}/ PE Connection resource. Allows the addition and manipulation of PE

DELETE /connections/{id}/ PE Connection resource. Allows the addition and manipulation of PE

http://localhost:8000/docs/ Page 1 of 6

Figure 4: The PE's registry Swagger Interface

[1] https://gitlab.com/project-dare/d4py-registry
[2] https://swagger.io/tools/swagger-ui/

5 Source Code and Deployment

The source code of the tools and services that have been introduced is available in the project's GitLab (S-ProvFlow[3], Processing Elements Registry[4]).

They are currently deployed in the DARE test-bed and managed by WP5. The MVV and BDV is available at http://snf-3480.ok-kno.grnetcloud.net/sprovflow-viewer/html/view.jsp

6 Conclusions

During this first year of project, the tools and services described in this deliverable have been refined with several improvements in functionalities, bug fixes and deployment model. Improvements include the implementation of an initial set of relevant design principles introduced by the DARE architecture, that particularly interests the conceptual and semantic classification of the research artifacts, which has to be captured in their provenance. The work will continue, in cooperation with WP2 and WP4, addressing additional provenance and lineage management requirements, especially with the incremental realisation of the DARE API and the mapping of dispel4py to additional enactment engines, such as Exareme[5]. This will introduce new provenance exploitation patterns and will impact on the current technical solution, which will be extended with the support of reusable provenance templates and with the realisation of communication and translation interfaces between the different DARE catalogues, towards a unified view.

[3] https://gitlab.com/project-dare/s-ProvFlow
[4] https://gitlab.com/project-dare/d4py-registry
[5] http://madgik.github.io/exareme/