D-JRA2.1.3: VERCE Architecture and Tools for Data-Intensive Applications

30/9/2014

Project acronym:	VERCE
Project n°:	283543
Funding Scheme:	Combination of CP & CSA
Call Identifier:	FP7-INFRASTRUCTURES-2011-2
WP:	WP9/JRA2, VERCE Architecture and Tools for Data Analysis and Data Modelling Applications
Filename:	D-JRA2.1.3.pdf
Authors[1]*:*	Malcolm Atkinson (UEDIN), Rosa Filgueira (UEDIN), Iraklis Klampanos (UEDIN), Jonas Matser (KNMI), Amy Krause (EPCC), Alessandro Spinuso (KNMI)
Location:	http://www.verce.eu/Repository/Deliverables/RP4
Type of document:	Deliverable
Dissemination level:	Public
Status:	Final
Due date of delivery:	30/9/2014 (Revised 20/11/2015)
Reviewers:	
Keywords:	JRA2, data-intensive, compute-intensive, distributed-systems, architecture, e-Infrastructure, tools.

History	*Author*	*Date*	*Comments*
1	I. Klampanos (UEDIN)	14/09/2014	Initial draft.
2	I. Klampanos (UEDIN)	19/09/2014	Populated core sections.
3	A. Spinuso (KNMI)	19/09/2014	Added content relating to the VERCE Gateway.
4	J. Matser (KNMI)	24/09/2014	Added content for portlets.
5	I. Klampanos (UEDIN)	24/09/2014	Architectural overview - MS41.
6	M. Atkinson (UEDIN)	26/09/2014	Modifications and suggestions.
7	R. Filgueira (UEDIN)	26/09/2014	Dissemination actions.
7	M.P. Atkinson (UEDIN)	20/11/2015	Inspected to identify any necessary revisions.

[1] Alphabetical order

Copyright notice

Contents

List of Figures

List of Tables

Executive Summary

JRA2 is responsible for leading the development of a high-level architecture for enabling and enhancing research in seismology. The JRA2 activities are intended to prototype new distributed, high-level integration services and tools, intelligently and intuitively exploiting European computational resources and therefore enable new methods for seismology research. These prototypes may then be adopted and developed into production components by other workpackages.

JRA2's activities aim to

- identify critical components and services for the VERCE platform
- identify, appropriately adapt and integrate existing seismology data resources and analysis tools
- derive a toolset for the effective and efficient development and parametrisation of scientific data-intensive workflows
- aid the design and prototyping of the VERCE scientific gateway by consulting with the scientific community, advising on potential paths to integration, etc.

The activities and actions undertaken within JRA2 are guided by its role as the VERCE architect. Purely technical goals aside, just as a building's architect must repeatedly interact with clients to tease out exactly what they need, what they can afford and ultimately to gain approval for particular decisions, so too must JRA2 meet with the Earth scientists, through NA2 and JRA1, as well as with the project's and work packages' leaders, to pursue understanding, influence thinking and gain commitment to critical decisions. In each six-month cycle, the JRA2 work package will develop, prototype, communicate with partners and report on a set of increments to the architecture, designed to take VERCE closer to its goal.

RP Objectives

Deliverables:

D9.1.3 D-JRA2.1.3: Annual revision of the VERCE architecture: catalogue of prototyped or upgraded services and tools and code package available to SA2 and to research developers (former D-JRA2.2.1) catalogue of new or upgraded portlets in the scientific gateway tuned to the requirements of researchers.

Milestone: MS41 (M-JRA2.4) Delivery of the VERCE architecture

The VERCE architecture has been delivered incrementally during previous as well as the current reporting period, through work agreed within and directed by JRA2. Figure depicts the overall VERCE architecture as well as the basic elements of each layer. Issues of detail were resolved by those engaged in co-design to meet seismologists' requirements precisely consulting the JRA2 team, who were frequently part of the active task forces and prototyping (co-development) efforts.

Work Progress

- Introduced `dispel4py`, a Python framework for the specification and deployment of fine-grained scientific workflows. `dispel4py` replaces Dispel and OGSA-DAI and is closer to the scientists' and architectural requirements. It supports enacting abstract workflows on arbitrary MPI computational platforms, Apache Storm clusters as well as on shared-memory multiprocessors.

- Improved the integration and operation of certificates, making authentication on the VERCE gateway easier.
  - The users can produce proxy certificates within the gateway itself.
  - The portal framework has been upgraded to the latest gUSE version supporting SHA2 encryption.
- Integration of the VERCE architecture with the SCAI cloud.
- Upgrade to SCI-BUS 3.6.6 and user certificates compatible with directives endorsed by EGI, PRACE and others.
- The VERCE Registry supports user registration and authentication. These mechanisms are currently independent of the authentication mechanisms implemented by the VERCE gateway.

Achievements

- Creation of the `dispel4py` library for creating streaming workflows.
- `dispel4py` demonstrated in real use-cases for seismic noise cross-correlation as well as for pre- and post-processing.
- Mapping of `dispel4py` to several computational platforms:
  - Local/development machines
  - Terracorrelator
  - Cloud
  - Shared-memory multi-processor architectures
- iRODS federation for storage of research data products across multiple sites
- Production-ready gUSE-powered VERCE gateway

In Progress and Next Steps

JRA2's plan for the next months are as follows:

- Extending VERCE by integrating with external data provision services, including services providing observational data.
- Finalise and package `dispel4py` for production use.
- Provide preconfigured library of seismological example pipelines.
- Registry-enabled component sharing.
- Incremental improvements on:
  - iRODS federation for observation and result data.
  - GridFTP bulk data transport.
  - Export provenance metadata to W3C PROV serialisation, enhancing interoperability.
  - Tools to exploit data provenance.
  - Use of iRODS services and MongoDB.

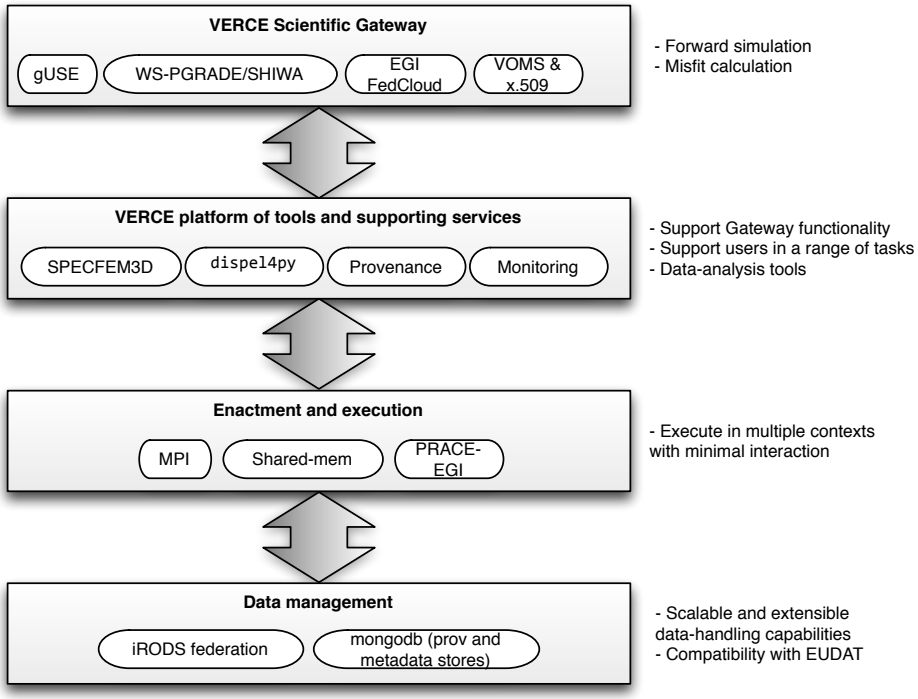


Figure 1 – Overview of the VERCE architecture.

Overview of the VERCE architecture. gUSE is a liferay-powered framework VERCE employs, resulting from our collaboration with SCI-BUS (http://www.sci-bus.eu). WSPGRADE/SHIWA integration provides VERCE users with added value, resulting from VERCE's collaboration with ER-FLOW (http://www.erflow.eu). The support of VOMS and x.509 certificates was a clear security requirement, which has been met both through gateway/portal technologies as well as by additional tools offering increased usability, provided by LRZ. SPECFEM3D is a widely used tool for simulating seismic wave propagation (http://geodynamics.org/cig/software/specfem3d). Support for SPECFEM3D has been explicitly requested by the seismology research community. `dispel4py` (https://github.com/akrause2014/dispel4py) is a Python framework for describing abstract data-intensive workflows. `dispel4py` is used for pre- and post-processing, as well as for pure data-intensive computations on local machines or MPI-powered clusters. The VERCE Provenance service is used on the gateway and it interacts directly or indirectly with a number of architectural components, such as the corresponding data management elements seen under "Data Management". It follows the W3C-PROV standard (http://www.w3.org/TR/prov-overview). Monitoring of execution is a service offered on the gateway, interacting with the provenance solution. MPI, shared-memory and PRACE-EGI are architectures the VERCE solution is able to make use of, thus adding to sustainability and standardisation. iRODS (http://irods.org) is a rules-based data-management system employed by VERCE in order to address requirements to do with scalability as well as with local policies at different VERCE sites. MongoDB (http://www.mongodb.org) is a document database used by the provenance and monitoring subsystems.

1 dispel4py

`dispel4py`[2] is a Python library, developed by VERCE, to describe abstract workflows for distributed data-intensive applications. These workflows are compositions of processing elements representing knowledge discovery activities (such as batch database querying, noise filtering and data aggregation) through which significant volumes of data can be streamed in order to manufacture a useful knowledge artefact. Such processing elements may themselves be defined by compositions of other, more fundamental computational elements, in essence having their own internal workflows. Users can construct workflows importing existing processing elements from a registry, or can define their own, recording them in a registry for later use by themselves or others. Abstract dataflows described in `dispel4py` can be executed in numerous environments, for example using a Storm cluster or as an MPI job. Thus `dispel4py` allows seismologists to construct workflows without particular knowledge of the specific context in which they are to be executed, granting them greater generic applicability.

The development of `dispel4py` resulted directly from the requirements of the seismology research community, especially due to the reliance of the community on Python and ObsPy as well as to the operational and management independence of the computing sites involved in scientific computation. `dispel4py` ensures maximum compatibility with standards, such as MPI, as well as flexibility for modelling and describing complex computations in a way independent of underlying technical, implementation-specific, or middleware details.

`dispel4py` replaces the use of Dispel, Dispel Gateways and OGSA-DAI, reported previously in the VERCE context.

2 Provenance

Provenance systems are used by modern workflow engines for collecting metadata about the data transformations which occur during runtime. If combined with effective visualisation and monitoring interfaces, these provenance recordings can speed up the validation process of an experiment, suggesting interactive or automated interventions with immediate effects on the lifecycle of a workflow run. For instance, in the field of computational seismology, if we consider research applications performing long lasting cross correlation analysis and high resolution simulations, the immediate notification of logical errors and the rapid access to intermediate results, can produce reactions which foster a more efficient progress of the research.

These applications are often executed in secured and sophisticated HPC and HTC infrastructures, highlighting the need for a comprehensive framework that facilitates the extraction of fine grained provenance and the development of provenance aware components, leveraging the scalability characteristics of the adopted workflow engines, whose enactment can be mapped to different technologies (MPI, Storm clusters, etc).

The current implementation of the system, apart from provenance generation mechanisms, provides a browser-based user interface and a web API built on top of a NoSQL storage technology. These components have been exposed to the users and improvements are applied incrementally, depending from the feedbacks collected. The current implementation, which complies with the W3C-PROV standard, allows for rapid and flexible access to lineage traces, it supports users with the visualisation of graphical products and offers combined operations for accessing and downloading data by direct or resolvable links towards dedicated data archives.

[2] https://github.com/akrause2014/dispel4py

3 Registry

The VERCE Registry, previously designed to work with Dispel, is now updated to complement `dispel4py`. In addition, it now supports user registration and authentication. As the Registry is currently seen as a 3rd party, external system, its authentication relies on a username/password mechanism and it is independent from that of the VERCE Gateway. The integration of the Registry with the VERCE Gateway/portal, in terms of both authentication and interaction, will be decided based on the VERCE 2014-2015 Roadmap. The VERCE Registry is designed so that it stores and serves descriptions and specifications of processing elements and other related components, as well as that it maintains versions and identities of these components.

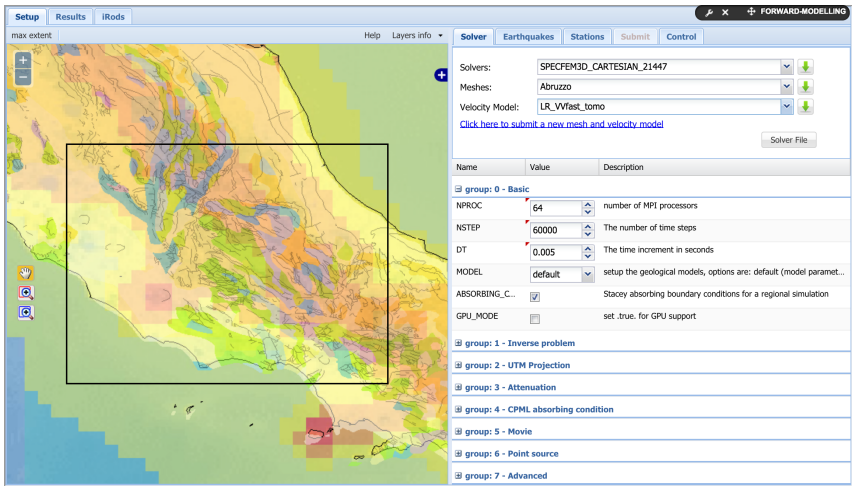
4 Gateway Components

The science gateway reached its first beta release during the previous reporting period and it is currently being updated to the latest release of the gUSE framework to support the SHA2 encryption algorithm adopted by the newly issued grid-certificates. The upgrade of the gateway is supported by SA1 and will be hosted in a cloud system maintained at SCAI. This new cloud environment will undergo the EGI FedCloud certification process. This upgrade is crucial to improve security and robustness of the gateway and will also give us more flexibility to test different layout of the core services. For instance, the new system tested the decoupling of the front-end component providing the gateway's portal environment and the back-end services, responsible for the interaction with the DCI. This is done by exploiting more VMs assigned to specific tasks and components of the gUSE framework. Moreover, we are currently working on the concrete integration of other resources adopting different grid middleware, more specifically the main target of this investigation will be the integration of the UNICORE PRACE resources provided by CINECA.

4.1 Portlets

The Forward Modeling portlet has been updated and extended with a number of new features. Much of the third party software used has been upgraded to a newer version (ExtJS5, GeoExt, ASM).

ExtJS5 is a significant improvement over version 4, that we used previously, allowing improvements to the SPECFEM confuration interface, as suggested by seismologists, that were problematic before. This concerns such things as using different input widgets for the different setting types, enforcing value ranges and grouping settings into useful categories.



The upgrade to ExtJS5 required updates for compatibility, which gave a good impulse to do some refactoring and refinement of the Forward Modeling portlet software. This included improving cross-browser compatibility and performance, using the ExtJS SDK compiler. GeoExt needed to be updated along with ExtJS for compatibility reasons.

ASM was updated to a newer version (the latest available, 3.4.10 + some additional changes that were not yet in an official release) that resolves some issues we had previously discussed with the SCI-BUS project. This nicely coincided with the upgrade of gUse to version 3.6.6.

4.2 WS-PGRADE workflows and SHIWA

Current workflows used for the Forward modelling have been exported to the SHIWA Platfrom. This is one of the outcomes of the MoU established with the ER-Flow project, which required the connection of the VERCE Science Gateway with an external workflow repository (The SHIWA workflow repository). This integration effort allowed us to export the implemented workflow and to make them available to other communities or science gateways, fostering interoperability of shareable methodologies across multiple gateways and, in the best scenarios, even across multiple scientific communities. For instance, the current WS-PGRADE workflows implemented in VERCE are mostly aiming at overcoming security constraints introduced by the adoption of professional HPC clusters and the derived complexity of interacting with our data infrastructure from within such clusters. We delegate the science case to a single component of the workflow which could be in principle composed of any relevant application or even workflow. This approach can be considered quite general and we believe might be reused in similar scenarios.

5 Dissemination Actions

1. **Publications in international workshops and conferences**:
   - Rosa Filgueira, Malcolm Atkinson, Andrew Bell, Ian Main, Branwen Snelling. *VarPy: A Python library for volcanology and rock physics data analysis.* (Super Computer 14, PyHPC'14 worksop) [Submitted], New Orleans, U.S, November 16-21.
   - Rosa Filgueira, Iraklis Klampanos, Amrey Krause, Mario David, Alexander Moreno, Malcolm Atkinson. *dispel4py: A Python Framework for Data-Intensive Scientific Computing.* (Super Computer 14, DISCS'14 worksop) [Accepted], New Orleans, U.S, November 16-21.
   - Sandra Gesing, Malcolm Atkinson, Rosa Filgueira, Ian Taylor, Andrew Jones, Vlado Stankovski, Chee Sun Liew, Alessandro Spinuso, Gabor Terstyanszky and Peter Kacsuk. *Workflows in a Dashboard: A New Generation of Usability.* (Super Computer 14, WORKS'14 worksop), [Accepted], New Orleans, U.S, November 16-21.
   - Rosa Filgueira, Malcolm Atkinson, Andrew Bell, Ian Main, Steve Boon, Christopher Kilburn and Philip Meredith. *eScience gateway stimulating collaboration in rock physics and volcanology.* (IEEE escience 2014), [Accepted], Guaruja, Brazil, October 20-24.
   - Alessandro Spinuso and the VERCE team. *Data Intensive and HPC seismological applications across European computing infrastructures.* (ISWG14). [Accepted]. Dublin, Ireland, June 3-5.
   - Rosa Filgueira, Malcolm P. Atkinson, Yusuke Tanimura, Isao Kojima. *Applying Selectively Parallel I/O Compression to Parallel Storage Systems.* (Euro-Par 2014), Porto, Portugal, August 27-29.
   - Rosa Filgueira, Iraklis Klampanos, Yusuke Tanimura, Malcolm Atkinson. *FAST: flexible automated synchronization transfer.* (HPDC, DIDC'14 workshop), Vancouver, Canada, June 23-27.

- Alessandro Spinuso and the VERCE team. *HPC and Data Intensive Seismology.* (APARSEN-EGI), Amsterdam, Netherlands, March 4-6.
- Peter Kacsuk. *Science Gateways for Distributed Computing Infrastructures*[3]. Software Engineering, Springer, ISBN 978-3-319-11267-1.

2. **Participation in international events**:
   - European Geosciences Union General Assembly, 2014, Vienna, Austria, 1 May.
     - The VERCE Science Gateway: enabling user friendly seismic waves simulations across European HPC infrastructures.
     - VarPy: A python library for volcanology and rock physics data analysis.
3. **Contribution to develop and test `dispel4py` workflows in different platforms** :
   - Cross-correlation application by using three platforms: SuperMUC cluster[4], Terra-correlator machine[5], and Open Science Data Cloud sullivan cluster[6].
   - Forward modelling application by using SuperMUC cluster.
4. **Production of online material**:
   - Training material accessible online as videos[7]
   - Production of the `dispel4py` website[8]

[3] http://www.springer.com/computer/swe/book/978-3-319-11267-1
[4] http://www.lrz.de/services/compute/supermuc/systemdescription/
[5] https://www.wiki.ed.ac.uk/display/Terra/Terra-correlator+wiki
[6] https://www.opensciencedatacloud.org/
[7] http://verce.eu/Training/UseVERCE.php
[8] http://www2.epcc.ed.ac.uk/~amrey/VERCE/dispel4py/

6 Software Summary Table

Table 1: Summary table of recommended, developed and used software. The column *Development* refers to software being developed internally or externally to VERCE, while the column *Status* provides an assessment of the software's robustness. The column *State* refers to the assessment state, ✔ indicating that a given software package has been assessed, while ◗ indicating that it is still under assessment. When a package has software prerequisites, these prerequisites are also taken to be part of the VERCE architecture implicitly.

Name	Version	Description	Devel.	Status	State
`dispel4py`	**1**	Description	Internal	Stable	✔
Verce Reg'y	**0.3**	VERCE Registry of processing elements and related entities. http://sisprojs.ipgp.fr/repos/verce/All/JRA/JRA2/VerceRegistry	Internal	Stable	✔
Verce Prov	**0.3**	VERCE Provenance Store and API. http://sisprojs.ipgp.fr/repos/verce/All/SA/SA3/Scientifc-Gateway/provenance-api/	Internal	Stable	✔
gUSE	**3.6.6**	Science gateway framework that allows users to make use of grid and cloud infrastructures. gUSE is used as part of the collaboration between VERCE and SCI-BUS. http://guse.hu	External	Stable	✔
Java	**6+**	General-purpose programming language by Oracle. http://www.oracle.com/us/technologies/java/overview/index.html	External	Stable	✔
Python	**2.7**	General-purpose scripting language. http://www.python.org	External	Stable	✔
MongoDB	**2.4.10+**	Open-source document database written in C++. http://www.mongodb.org	External	Stable	✔
Tomcat	**5.5+**	Java servlet container. http://tomcat.apache.org	External	Stable	✔
ObsPy	**0.92**	Toolkit for seismological computations.	External	Stable	✔

Appendices

A Sample of the VERCE Registry API

VERCE Registry REST API

Italics denote optional parameters. Types are expressed in pseudo language.
By default parameters are provided as part of a JSON request. Parameters followed by * denote HTTP parameters.

Users and Groups

Login

Endpoint	/rest/login		
Method	POST		
	Name	Type	Example
Parameters	username*	String	“pat”
	password*	String	“patspasswd”
Returns	security token	String	

Create new user

Endpoint	/rest/user/create		
Method	POST		
	Name	Type	Example
Parameters	username	String	“pat”
	password	String	“patspasswd”
	email	Email	“pat@verce.eu”
	defaultGroup	String	“patgroup”
	groups	[String]	[“patgroup”]
	isAdmin	Boolean	false
Returns	user	json	

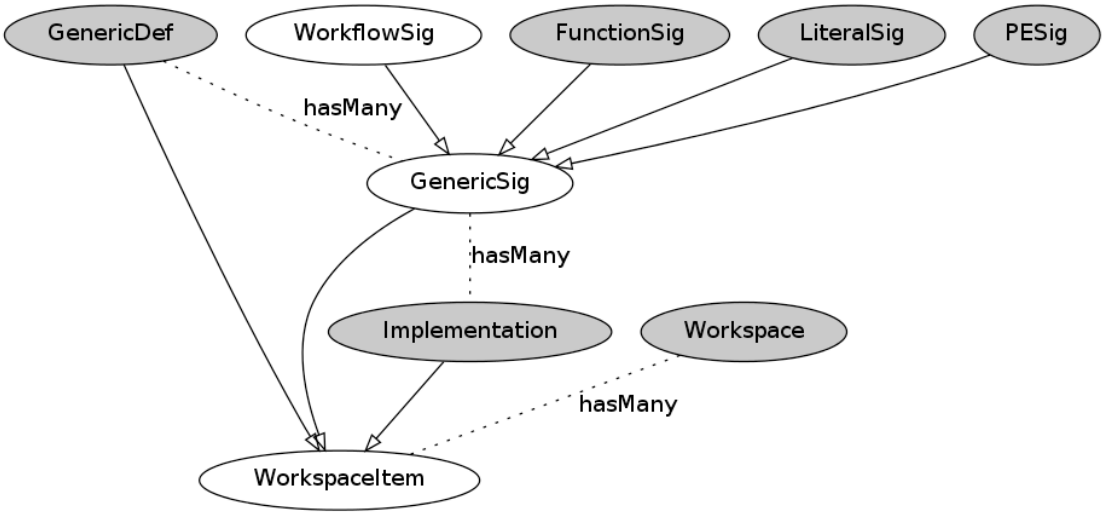
Modify user (add to groups, etc.)

Endpoint	/rest/user/create		
Method	PUT		
	Name	Type	Example
Parameters	username	String	“pat”
	password	String	“patspasswd”
	email	Email	“pat@verce.eu”
	defaultGroup	String	“patgroup”
	groups	[String]	[“patgroup”]
	isAdmin	Boolean	false
Returns	user	json	

Create new user group

Endpoint	/rest/group/create		
Method	POST		
	Name	Type	Example
Parameters	name	String	“pat”
	description	String	“patspasswd”
Returns	group	json	

Workspace and Related VERCE Registry Entities



The main registry entities are depicted above. The arrows denote inheritance. The greyed out entities are the ones directly managed by the client. At this point it is worth noting that each signature workspace item (`GenericSig`) should correspond to a generic definition (`GenericDef`). This is to allow for multiple interpretations of a definition. The purpose of the signatures is to define parameters and descriptions, for instance to cater for different contexts or manually managed versions of an element. Finally, each signature can be implemented in more than one ways, via the `Implementation` entity.

Create a workspace

Endpoint	/rest/workspaces/create		
Method	POST		
	Name	Type	Example
Parameters	name	String	“pat’s workspace”
	group > name	String	“pat”
	owner > username	String	“pat”
	parent > name,username	list	“name”:“root”, “username”:“admin”
Returns	workspace	json	

Register a GenericDef

Endpoint	/rest/gendef		
Method	POST		
	Name	Type	Example
Parameters	defDescription	String	“some description”
	workspaceId	Long	1
	pckg	String	“my.package”
	name	String	“MyGenDef”
	group	String	“Edinburgh”
Returns	GenericDef	json	

Register a FunctionSig

Endpoint	/rest/gendef		
Method	POST		
	Name	Type	Example
Parameters	defDescription	String	“some description”
	workspaceId	Long	1
	pckg	String	“my.package”
	name	String	“MyGenDef”
	group	String	“Edinburgh”
	parameters	[(String, String)]	[(“int”, “param1”),…]
	returnType	String	int
Returns	FunctionSig	json	

Register a LiteralSig

Endpoint	/rest/literal		
Method	POST		
	Name	Type	Example
Parameters	defDescription	String	“some description”
	workspaceId	Long	1
	pckg	String	“my.package”
	name	String	“MyGenDef”
	group	String	“Edinburgh”
	value	Any	
	genericDefId	Long	1
Returns	LiteralSig	json	

Register a PESig

Endpoint	/rest/pe		
Method	POST		
	Name	Type	Example
Parameters	workspaceId	Long	1
	pckg	String	“my.package”
	name	String	“MyGenDef”
	group	String	“Edinburgh”
	kind	int (0: abstract, 1: primitive, 2: composite)	0
	returnType	String	int
	genericDefId	Long	1
	FunctionSig	json	
Returns	PESig		

Add an Implementation

Endpoint	/rest/implementation		
Method	POST		
	Name	Type	Example
Parameters	workspaceId	Long	1

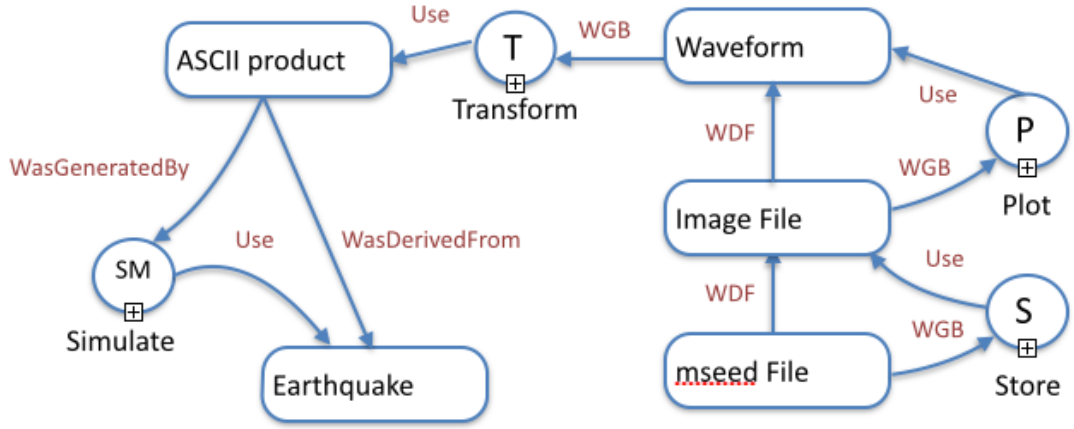
	pckg	String	“my.package”
	name	String	“MyGenDef”
	group	String	“Edinburgh”
	code	String	
	genericDefId	Long	1
Returns	Implementation		

B VERCE Provenance API

VERCE Provenance REST API

Italics denote optional parameters. Types are expressed in pseudo language.
By default parameters are provided as part of a JSON request. Parameters followed by * denote HTTP parameters.

Below we present a sample schema, showing the provenance relationships among the processes and the data involved in a seismic simulation, according to the W3C-PROV data model.



Workflow Runs

Insertion of lineage traces and workflow run description

Endpoint	/workflow/insert		
Method	POST		
	Name	Type	Example
Parameters	doc*	json	JSON Document
Returns	Insertion Confirmation	json	JSON Document

Edit workflow description

Endpoint	/workflow/edit/run_id		
Method	POST		
	Name	Type	Example
Parameters	doc*	json	JSON Document
Returns	Update Confirmation	json	

Delete workflow run

Endpoint	/workflow/delete/run_id		
Method	POST		
Returns	Deletion Confirmation	json	

Retrieve user's runs

Endpoint	/workflow/user/username		
Method	GET		
	Name	Type	Example
Parameters	start*	int	0
	limit*	int	1000
Returns	List of runs' metadata	json	

Export Run

Endpoint	/workflow/export/run_id		
Method	GET		
	Name	Type	Example
Parameters	start*	int	0
	limit*	int	1000
	format*	String	"w3c-prov"
Returns	List of run's provenance traces in one of the selected formats	json	

Workflow Run Activities and Entities

Retrieve run's activities metadata

Endpoint	/activities/run_id		
Method	GET		
	Name	Type	Example
Parameters	start*	int	0
	limit*	int	1000
Returns	List of run's activities metadata	json	

Retrieve data entities generated by an activity

Endpoint	/entities/generatedby		
Method	GET		
	Name	Type	Example
Parameters	ierationId	String	"decomposeMeshi19r0.."
	start*	int	0
	limit*	int	1000
Returns	List of activity's data entities metadata	json	

Retrieve data entities based on their metadata

Endpoint	/entities/values-range		
Method	GET		
	Name	Type	Example
Parameters	run_id* (optional)	String	“scai-nordit0141138”
	start*	int	0
	limit*	int	1000
	keys*	String (csv)	“magnitude,station”
	Maxvalues*	String (csv)	“3,AQU”
	Minvalues*	String (csv)	“5.6,AQU”
	mime-type* (optional)	String	“video/mpeg”
Returns	List of data entities entities metadata	json	

Retrieve data entities based on their ancestors metadata

Endpoint	/entities/hasAncestorWith		
Method	GET		
	Name	Type	Example
Parameters	dataId*	String	“i19r01a03-27189-..”
	start*	int	0
	limit*	int	1000
	keys*	String (csv)	“magnitude,station”
	maxvalues*	String (csv)	“3,AQU”
	minvalues*	String (csv)	“5.6,AQU”
Returns	List of data entities’ metadata	json	

Filter a set of entity ids based on the entities’ ancestors metadata

Endpoint	/entities/filterOnAncestorsValuesRange		
Method	GET		
	Name	Type	Example
Parameters	ids*	String (csv)	“i19r01a03-27189-..”
	start*	int	0
	limit*	int	1000
	keys*	String (csv)	“magnitude,station”
	maxvalues*	String (csv)	“3,AQU”
	minvalues*	String (csv)	“5.6,AQU”
Returns	List of filtered data entities’ ids	json	

Data Derivations

Navigates the data derivation graph starting from the current entity through all of the contributing entities, until a specified level or step (Backwards)

Endpoint	/wasDerivedFrom/entity_id		
Method	GET		
	Name	Type	Example
Parameters	level*	int	2
Returns	List of linked data dependencies	json	

Navigates the data derivation graph starting from the current entity through all of derived entities, until the specified level or step (Forward)

Endpoint	/derivedData/entity_id		
Method	GET		
	Name	Type	Example
Parameters	level*	int	2
Returns	List of linked data dependencies	json	