

**EGI-Engage**

Final release of the accounting and operational tools

D3.17

|  |  |
| --- | --- |
| **Date** | 28 July 2017 |
| **Activity** | WP3 |
| **Lead Partner** | [provide] |
| **Document Status** | DRAFT |
| **Document Link** | https://documents.egi.eu/document/3037 |

Abstract

**COPYRIGHT NOTICE**



This work by Parties of the EGI-Engage Consortium is licensed under a Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/). The EGI-Engage project is co-funded by the European Union Horizon 2020 programme under grant number 654142.

**DELIVERY SLIP**

|  |  |  |  |
| --- | --- | --- | --- |
|  | ***Name*** | ***Partner/Activity*** | ***Date*** |
| **From:** |  |  |  |
| **Moderated by:** |  |  |  |
| **Reviewed by** |  |  |  |
| **Approved by:** |  |  |  |

**DOCUMENT LOG**

|  |  |  |  |
| --- | --- | --- | --- |
| ***Issue*** | ***Date*** | ***Comment*** | ***Author/Partner*** |
| **v.1** |  |  |  |
| **...** |  |  |  |
| **...** |  |  |  |
| **v.n** |  |  |  |

**TERMINOLOGY**

A complete project glossary and acronyms are provided at the following pages:

* <https://wiki.egi.eu/wiki/Glossary>
* <https://wiki.egi.eu/wiki/Acronyms>

**Contents**

[1 Operations Portal 7](#_Toc486517639)

[1.1 Introduction 7](#_Toc486517640)

[1.2 Service architecture 7](#_Toc486517641)

[1.2.1 High-Level Service architecture 8](#_Toc486517642)

[1.2.2 Integration and dependencies 8](#_Toc486517643)

[1.3 Release notes 8](#_Toc486517644)

[1.3.1 Requirements covered in the release 8](#_Toc486517645)

[1.4 Feedback on satisfaction 8](#_Toc486517646)

[1.5 Plan for Exploitation and Dissemination 8](#_Toc486517647)

[1.6 Future plans 10](#_Toc486517648)

[2 ARGO 11](#_Toc486517649)

[2.1 Introduction 11](#_Toc486517650)

[2.2 Service architecture 11](#_Toc486517651)

[2.2.1 High-Level Service architecture 12](#_Toc486517652)

[2.2.2 Integration and dependencies 12](#_Toc486517653)

[2.3 Release notes 12](#_Toc486517654)

[2.3.1 Requirements covered in the release 12](#_Toc486517655)

[2.4 Feedback on satisfaction 12](#_Toc486517656)

[2.5 Plan for Exploitation and Dissemination 12](#_Toc486517657)

[Future plans 14](#_Toc486517658)

[3 Messaging Service 15](#_Toc486517659)

[3.1 Introduction 15](#_Toc486517660)

[3.2 Service architecture 15](#_Toc486517661)

[3.2.1 High-Level Service architecture 16](#_Toc486517662)

[3.2.2 Integration and dependencies 16](#_Toc486517663)

[3.3 Release notes 16](#_Toc486517664)

[3.3.1 Requirements covered in the release 16](#_Toc486517665)

[3.4 Feedback on satisfaction 16](#_Toc486517666)

[3.5 Plan for Exploitation and Dissemination 16](#_Toc486517667)

[3.6 Future plans 18](#_Toc486517668)

[4 GOCDB 19](#_Toc486517669)

[4.1 Introduction 19](#_Toc486517670)

[4.2 Service architecture 19](#_Toc486517671)

[4.2.1 High-Level Service architecture 20](#_Toc486517672)

[4.2.2 Integration and dependencies 20](#_Toc486517673)

[4.3 Release notes 20](#_Toc486517674)

[4.3.1 Requirements covered in the release 20](#_Toc486517675)

[4.4 Feedback on satisfaction 20](#_Toc486517676)

[4.5 Plan for Exploitation and Dissemination 20](#_Toc486517677)

[4.6 Future plans 22](#_Toc486517678)

[5 Security Monitoring 23](#_Toc486517679)

[5.1 Introduction 23](#_Toc486517680)

[5.2 Service architecture 23](#_Toc486517681)

[5.2.1 High-Level Service architecture 24](#_Toc486517682)

[5.2.2 Integration and dependencies 24](#_Toc486517683)

[5.3 Release notes 24](#_Toc486517684)

[5.3.1 Requirements covered in the release 24](#_Toc486517685)

[5.4 Feedback on satisfaction 24](#_Toc486517686)

[5.5 Plan for Exploitation and Dissemination 24](#_Toc486517687)

[5.6 Future plans 26](#_Toc486517688)

[6 Accounting Repository 27](#_Toc486517689)

[6.1 Introduction 27](#_Toc486517690)

[6.2 Service architecture 27](#_Toc486517691)

[6.2.1 High-Level Service architecture 28](#_Toc486517692)

[6.2.2 Integration and dependencies 28](#_Toc486517693)

[6.3 Release notes 28](#_Toc486517694)

[6.3.1 Requirements covered in the release 28](#_Toc486517695)

[6.4 Feedback on satisfaction 28](#_Toc486517696)

[6.5 Plan for Exploitation and Dissemination 28](#_Toc486517697)

[6.6 Future plans 30](#_Toc486517698)

[7 Accounting Portal 31](#_Toc486517699)

[7.1 Introduction 31](#_Toc486517700)

[7.2 Service architecture 31](#_Toc486517701)

[7.2.1 High-Level Service architecture 32](#_Toc486517702)

[7.2.2 Integration and dependencies 32](#_Toc486517703)

[7.3 Release notes 32](#_Toc486517704)

[7.3.1 Requirements covered in the release 32](#_Toc486517705)

[7.4 Feedback on satisfaction 32](#_Toc486517706)

[7.5 Plan for Exploitation and Dissemination 32](#_Toc486517707)

[7.6 Future plans 34](#_Toc486517708)

[Appendix I. ARGO development process 35](#_Toc486517709)

[Appendix II. GOCDB development process 36](#_Toc486517710)

**Executive summary**

# Operations Portal

## Introduction

|  |  |
| --- | --- |
| **Tool name** | Operations Portal |
| **Tool url** | <http://operations-portal.egi.eu> |
| **Tool wiki page** | <https://wiki.egi.eu/wiki/Operations_Portal> |
| **Description** | The Operations Portal provides VO management functions and other capabilities, which support the EGI daily operations. It is a central portal for the operations community that offers a bundle of different capabilities, such as the broadcast tool, VO management facilities, a security dashboard and an operations dashboard that is used to display information about failing monitoring probes and to open tickets to the affected Resource Centres. The dashboard also supports the central grid oversight activities. It is fully interfaced with the EGI Helpdesk and the monitoring system through messaging. It is a critical component as it is used by all EGI Operations Centres to provide support to the respective Resource Centres. The Operations Portal provides tools supporting the daily running of operations of the entire infrastructure: grid oversight, security operations, VO management, broadcast , VO metrics.VAPOR: the Vo Administration and operations PORtal, is a generic tool to assist community managers and support teams in performing their daily activities. The application provides resources status indicators, statistical reports, data management tools. It gathers the resources information from the BDII and displays them in a ordered way, replacing the features previously offered by GSTAT. The amount of resources and the resources themselves are shown in different views that group information per Operations Centres, Countries and VOs. |
| **Value proposition** | New features offered by the Operations Portals allow its customers to better monitor and browse the infrastructure and, then, adapting their workflows according to the exact status of the computing and storage resources (e.g. moving some computation from one provider to another since the latter is working better). |
| **Customer of the tool** | EGI; NGI; RI; Resource Provider; Research Communities |
| **User of the service** | Site admins; Operations Managers; VO Manager; VO users;  |
| **User Documentation**  | <https://forge.in2p3.fr/projects/opsportaluser/wiki/Main_Features_of_the_dashboard><http://operations-portal.egi.eu/vapor/globalHelp> |
| **Technical Documentation**  | <https://forge.in2p3.fr/projects/opsportaluser/wiki/Main_Features_of_the_dashboard>  |
| **Product team** | IN2P3/CNRS |
| **License** | Apache 2.0 |
| **Source code** | <https://gitlab.in2p3.fr/groups/opsportal>  |

## Service architecture

### High-Level Service architecture

The Operations Portal has been designed as an integration platform, allowing for strong interaction among existing tools with similar scope but also filling up gaps wherever functionality has been lacking. The displayed information is retrieved from several distributed static and dynamic sources – databases, Grid Information System, Web Services, etc. – and gathered within the portal.

The architecture of the portal is composed of three modules:

* A database – to store information related to the users or the VO;
* A web module – graphical user interface – which is currently integrated into the Symfony framework;
* A Data Aggregation and Unification Service named Lavoisier.

Lavoisier is the component used to store, consolidate and “feed” data into the web application.

The global information from the primary and heterogeneous data sources (e.g. BDII, GOCDB, NAGIOS, GGUS, ARGO, etc.) is retrieved by means of the use of the different plug-ins. The collected information is structured and organized within configuration files in Lavoisier and, finally, made available to the web application without the need for any further computations. This modular architecture is conceived to add easily new data source in this model and use the cached information if a primary source is unavailable. The data sources are refreshed only as needed and only when an action has been triggered. In addition, it is very easy to add a new data source in this model, as depicted in Fig. 1 and Fig. 2. Nevertheless, two critical dependencies are remaining: GGUS[[1]](#footnote-1) and RTIR[[2]](#footnote-2) (red arrows on the left on next figure).

These dependencies are due to the communication via web services between the Operations Portal and GGUS/RTIR for the creation or the update of tickets.

In case of disruptions of the GGUS or RT services, a part of the features of the Operations Portal will be affected: the creation and the update of tickets into the dashboards. For the rest of data sources, the cache mechanism of Lavoisier permits us to ensure the integrity of the application in case of failures of third parties providers.



Figure 1. Operations Portal architecture

For the VAPOR application, we use the same architecture with a dedicated instance of Lavoisier. Information is aggregated from several top BDII objects and from a monitoring tool based on Jsaga (JobMonitor) and local scripts in python and shell developed specifically to ease the VO support.

VAPOR is fully integrated in the Operations Portal and is presented to the users as an additional feature available.



Figure 2. VAPOR architecture

### Integration and dependencies

Operations Portal dependencies have been already described in the previous section. They are not changed in this release.

## Release notes

### Operations Portal 4.0

This version is a major evolution of the background technologies of the portal.

The aim was to upgrade the different technologies used around the portal and ensure a better maintainability and an improvement of the performances. Here are the main changes for this version:

1. Frameworks & JS Libraries
* Migration to Symfony 3;
* Upgrade of bootstrap library;
* Adoption of the Datatables Js libraries to optimize the presentation of the tables (VO Management, Metrics);
* Use of Google Chart (VO Management, Metrics).
1. Ergonomics
* Addition of links to ARGO and VAPOR applications;
* Changes into global menu presentation (and optimization depending on screen size).
1. Module and project modifications
* Reorganisation of the project infrastructure;
* Removal of obsolete files and features;
* Merge of the VO Management Tool and VO ID cards (all-in-one page);
* Removal of Availabilities/reliabilities module (replaced by ARGO).
1. Downtime Module (new module)

The historical downtime subscription system has been removed and replaced within a dedicated module offering the following features:

* A subscription page (emails , rss , ical);
* Timelines charts and tables;
* Search tool;
* Data exportable in different formats (CSV, JSON).
1. Continuous Integration
* A procedure about good practices for the development procedure is in place: <https://forge.in2p3.fr/projects/opsportaluser/wiki/Development_Procedure>
* An integration platform has been set-up with PHPUnit , GitlabCI , docker and SonarQBE: <https://forge.in2p3.fr/projects/opsportaluser/wiki/Continuous_Integration>

### Operations Portal 4.1

This version was focused on:

* Several improvements on the VO ID cards;
* Improvement of the documentation of the main features;
* The fixes of different bugs due to the important changes of the previous version.

### Operations Portal 4.2

This version is foreseen for August and is focused on:

* Integration of complementary metrics for the VO: accounting data and AppDB changes;
* Improvements on the VO ID Card;
* The support of the new EGI AAI based on the CheckIn service (IdP/SP Proxy).
* A backend for the monitoring
	+ Exploration of logs (apache , symfony, access)
	+ Status of the Lavoisier servers and views
	+ Status of some tables of the DB
	+ The use of ARGO messaging system to collect Nagios notifications

### VAPOR 2.0

The initial prototype (described in D3.4[[3]](#footnote-3)) has been put in production after a test phase of one month.

### VAPOR 2.1

The main features of this release were:

* Integration of GSTAT features;
	+ a map of the resources:

<http://operations-portal.egi.eu/vapor/resources/GL2Map>

* + a table of the resources:

<http://operations-portal.egi.eu/vapor/resources/GL2ResSummary>

* + a Top BDII browser:

<http://operations-portal.egi.eu/vapor/resources/GL2ResBdiiBrowser>

* New menu;
* Bug fixing;
* Integration of feedback given by users;
* Ergonomics improvements.

### VAPOR 2.2

This release has been delivered in February 2017.

For this release, the Operations Portal team has worked closely with the EGI Operations to consolidate the different queries to the Top BDII and the different extracted figures. The results are the following:

* A summary of the CPU and storage capacities by countries, sites or Operations Centres;
* A geographical map with the distribution of sites with a VO filter;
* Some additions in the faulty publications: bad HEPSPEC, mismatches between the different benchmarks, negative values for jobs.

This release has been also focused on the documentation of the different features and the access to the API.

### VAPOR 2.3

This release is currently in the test phase and will be delivered in August 2017.

Once again this release is the results of multiple exchanges with EGI Operations team to enhance the current features. We have worked on different improvements :

* Upgrade of the different javascript libraries to improve the performances.
* identify the duplicated values published by the sites.
* A map has been added with a global view of all the sites.
* A summary of the figures is now available for each site.
* The global storage capacity computation has been improved.
* One new metric has been added in agreement with EGI Operations team : “the Computation power”

## Feedback on satisfaction

Prioritization and testing has been done by dedicated Operations Portal Advisory and Testing Board (OPAnTG)[[4]](#footnote-4) coordinated by EGI Operations team. Furthermore, the Operations Portal team has worked on the automation of tests. Unit and acceptance tests are now done through Docker piloted by GitLab Continuous Integration server.

If tests are failing, new features are not propagated to the test infrastructure. This allows performing a first bug filter before manually tests are executed. Complementary to these tests, the team also adopted a SonarQBE instance to inspect the quality of code.

The architecture of the Operations Portal automatic test suite is described below.

As a result, a minor number of bugs have been identified by the testing team in the most recent releases.



Figure 3. Operations Portal - Automatic test suite.

## Plan for Exploitation and Dissemination

|  |  |
| --- | --- |
| *Name of the result* | Operation Portal |
| *DEFINITION*  |
| *Category of result* | Software & service innovation |
| *Description of the result* | Software enhancement: integrate the VO Administration and operations PORtal (VAPOR) into the Operations Portal and enhance the monitor infrastructure resources including the most relevant features currently offered by GSTAT.  |
| *EXPLOITATION* |
| *Target group(s)* | Users, NGIs, Resource centres, RIs |
| *Needs* | Monitor / browse / Evaluate the resources for VO, sites, Operations Centres |
| *How the target groups will use the result?* | * Exploit the new features in the daily operations of the EGI infrastructure
* Exploit the advanced metrics to better promote the EGI infrastructure
 |
| *Benefits* | * Ease the daily administration of the resources
* Have an overview of the resources and their status
* Be more efficient in the daily job submission
 |
| *How will you protect the results?* | Apache 2 License |
| *Actions for exploitation* | The result is accessible through the web site and the code is hosted on a gitlab. |
| *URL to project result* | <http://operations-portal/vapor><https://gitlab.in2p3.fr/opsportal/>  |
| *Success criteria* | The deployment in production and the use by end users. |
| *DISSEMINATION* |
| *Key messages* | Browse and evaluate your resources |
| *Channels* | EGI Broadcast tool, EGI Meetings |
| *Actions for dissemination* | EGI conferences, publications, participation to workshops organised by potential users. |
| *Cost* |  |
| *Evaluation* | The number of requests and the feedback given by users |

## Future plans

* VAPOR
	+ Enhance the historical scripts, especially the ‘JobMonitor’ Tool;
	+ Consolidation / coherency of the data:
		- Data issued from site publications with incoherencies:
		- Detect and propose corrections:
	+ Extend the current features with user feedback.
* Operations Portal
	+ integration of complementary metrics for the VO;
	+ Add more genericity in the VO Id cards;
	+ Extend the current features with user feedback;
	+ Adapt the current tools to the new communities;
	+ Define a new module for the SLA/OLA management including:
		- workflows to automatic service activations;
		- on-demand generation of reports on resource usage.

# ARGO

## Introduction

|  |  |
| --- | --- |
| **Tool name** | ARGO |
| **Tool url** | <http://argo.egi.eu>  |
| **Tool wiki page** | <https://wiki.egi.eu/wiki/ARGO>  |
| **Description** | ARGO is a flexible and scalable framework for monitoring status, availability and reliability |
| **Value proposition** | Improved portal design that allows new and easier way to access and visualise data for the final users. Third parties can now gather monitoring data from the system through a complete API. A central deployment of the ARGO monitoring engine can serve a large infrastructure reducing the maintenance costs. |
| **Customer of the tool** | EGI; NGI; RI; Resource Provider; Research Communities |
| **User of the service** | Site admins; Operations Managers; large research group |
| **User Documentation**  | <http://argoeu.github.io>;<http://argo.egi.eu> |
| **Technical Documentation**  | <http://argoeu.github.io> |
| **Product team** | GRNET, SRCE, CNRS |
| **License** | Apache License Version 2.0 |
| **Source code** | <https://github.com/ARGOeu/>  |

## Service architecture

### High-Level Service architecture

ARGO is a flexible and scalable framework for monitoring status, availability and reliability of services provided by infrastructures with medium to high complexity. It can generate multiple reports using customer defined profiles (e.g. for SLA management, operations, etc.) and has built-in multi-tenant support in the core framework.

ARGO supports flexible deployment models and its modular design enables ARGO to be integrated with external systems (such as CMDBs, Service Catalogues, etc.). During the report generation, ARGO can take into account custom factors such as the importance of a specific service endpoint, scheduled or unscheduled downtimes, etc.



Figure 4. ARGO architecture.

For the Availability & Reliability monitoring, ARGO relies on a modular architecture comprised of the following components:

#### The ARGO Monitoring Engine

For status monitoring, ARGO relies on Nagios. All probes developed for ARGO follow the Nagios conventions and can run on any stock Nagios box. ARGO provides an optional set of add-ons for the stock Nagios that provide features such as auto-configuration from external information sources, publishing results to external Message Brokers, etc.

In order to use the new messaging service, the monitoring engine also supports the new AMS Publisher. The AMS publisher is a new component acting as bridge from Nagios to ARGO Messaging system. It is integral part of software stack running on ARGO monitoring instance and is responsible for forming and dispatching messages that are results of Nagios tests. Ready and running on the development infrastructure. It is running as a Unix daemon and it consists of two subsystems:

* Queueing mechanism;
* Publishing/dispatching part.

Messages are cached in a local queue with the help of OCSP Nagios calls and each queue is being monitored by the daemon. After configurable amount of accumulated messages, the publisher that is associated to queue sends them to ARGO Messaging system and drains the queue. argo-nagios-ams-publisher is written in multiprocessing manner enabling the support for multiple queue/publish pairs where for each, new worker process will be spawned.

#### The ARGO Connectors

Through the use of custom connectors, ARGO can connect to multiple external Configuration Management Databases and Service Catalogues. Connectors for the EGI and EUDAT e-Infrastructures are already available.

#### The ARGO Consumer

The ARGO Consumer is ingesting monitoring results in real-time from external Message Brokers. The consumer is responsible for the initial pre-filtering of the monitoring results and encodes them using AVRO serialization format[[5]](#footnote-5) before passing to the Compute Engine.

#### The ARGO Compute Engine

A powerful and scalable analytics engine built on top of Hadoop and HDFS[[6]](#footnote-6). The Compute Engine is responsible for the aggregation of the status results and the computation of availability and reliability of composite services using customer defined algorithms. The reorganization of the Compute Engine to support stream processing in real time is one of the key new factors. A new streaming layer has been introduced. Monitoring results flow through the AMS, to the streaming layer (in parallel to the HDFS). The streaming layer is used in order to push raw metric results to the metric result store and to compute status results and push them to the status store in real-time.

#### The ARGO Web API

The ARGO Web API provides the serving layer of ARGO. It is comprised of a high performance and scalable data store and a multi-tenant REST HTTP API, which is used for retrieving the Status, Availability and Reliability reports and the actual raw metric results.

#### The ARGO Web UI

The default web UI is based on the Lavoisier Data Aggregation Framework[[7]](#footnote-7).

### Integration and dependencies

ARGO can utilize external configuration sources through connectors in order to allow the automatic configuration of various ARGO components. The current version of ARGO includes connectors for the following sources:

* GOCDB: It is used as the source of EGI infrastructure topology information and information about declared downtimes.
* VAPOR: It is used as the source for custom factor values, which in the case of EGI it is the HEPSPEC[[8]](#footnote-8) values of the sites.

The dependency to these external tools is optional. ARGO can be used without having any of these connectors enabled, if there is at least a static configuration for the topology of the monitored infrastructure.

Finally, ARGO relies on the Message Broker network as the transport layer for publishing monitoring results from the Nagios Monitoring Engines to the ARGO Compute Engine.

## Release notes

### Requirements covered in the release

As already mentioned ARGO is not just single software, but a suite of software components, each one managed independently. During the third year of the project, there have been a number of releases of the ARGO components that covered the following requirements:

**ARGO Compute Engine & Web API**

* Streaming processing;
* Alerting mechanism;
* Separation of A/R and Metric stores:
* APIv2;
* Stability and performance improvements.

**ARGO Monitoring Engine**

* Migration of ops probes from opsmon to the central monitoring instances argo-mon/2.egi.eu and decommissioning of opsmon.egi.eu;
* Deployment of three new ARGO Monitoring Services:
	+ Testing instance (argo-mon-test) used for testing new ARGO Monitoring Service releases and deployment of new probes and updates of existing probes; instance is constantly monitoring subset of EGI infrastructure and list of sites and service endpoints is extended on demand;
	+ Uncertified instance (argo-mon-uncert) used for monitoring uncertified sites which fully relies on information provided by sites in GOCDB;
	+ Internal instance used for monitoring all internal ARGO components by using ARGO probes and NRPE;
* New probes and updates of existing probes:
	+ New probe for decommissioning of dCache 2.10 and dCache 2.13;
	+ New probes for OneData services;
	+ New probes for AAI CheckIn service;
	+ New probe for NGI Argus service;
	+ New probe for WebDAV service;
	+ Improved probes for FTS3, gsisshd and VOMS services;
	+ Improved probes for CREAM-CE;
	+ Analysis and deployment of new ARC-CE probes;
	+ Scripts provided for handling UNICORE probes configuration;
* Prototype version of ARGO Monitoring Service for biomed VO;
* AMS Publisher: is a new component acting as bridge from Nagios to ARGO Messaging system. It is integral part of software stack running on ARGO monitoring instance and is responsible for forming and dispatching messages that are results of Nagios tests. Successfully running on the devel infrastructure for more than a month;
* Support for GOCDB as a single source of topology:
	+ Step 1: Randomly check service endpoint;
* Stability and performance improvements.

**ARGO EGI Consumer and Connectors**

* Use of ARGO nagios AMS-publisher;
	+ Ready on devel infrastructure;
* Use of the messaging API for Connectors component;
	+ Ready on devel infrastructure;
* Stability and performance improvements.

**ARGO EGI Web UI**

* New Uncertified report;
* New FedCloud Report;
* UI Enhancements;
	+ New pdf report;
	+ Updates to ELIXIR report;
	+ Updates to admin list;
	+ Updates to links to reports.

**ARGO POEM**

* Finalize support for probe management;
* Initial steps for the connection to the EGI IdP/SP Proxy;
* Stability and performance improvements.

#### Changelog

* **25/06/2017**
	+ ARGO Monitoring Plugin - AMS publisher [Version 0.2.0-1] <https://github.com/ARGOeu/argo-nagios-ams-publisher/releases/tag/v0.2.0-1>
	+ ARGO-Monitoring Engine [Version 0.4.0-1] <https://github.com/ARGOeu/argo-ncg/releases/tag/v0.4.0-1>
* **20/06/2017**
	+ ARGO-Poem [Version 1.0.5-1] <https://github.com/ARGOeu/poem/releases/tag/v1.0.5-1>
* **24/05/2017**
	+ ARGO-Connectros [Version 1.5.9-1] <https://github.com/ARGOeu/argo-egi-connectors/releases/tag/v1.5.9>
	+ ARGO-Monitoring Engine [Version 0.3.4-1] <https://github.com/ARGOeu/argo-ncg/releases/tag/0.3.4-1>
* **06/05/2017**
	+ ARGO-Poem [Version 1.0.4-1] <https://github.com/ARGOeu/poem/releases/tag/v1.0.4-1>
	+ ARGO Web UI [Version 1.3.6-2] <https://github.com/ARGOeu/argo-egi-web/releases/tag/V1.3.6-2>
* **04/05/2017**
	+ ARGO-Monitoring Engine [Version 0.3.3-1] <https://github.com/ARGOeu/argo-ncg/releases/tag/0.3.3-1>
* **03/04/2017**
	+ ARGO-Monitoring Engine [Version 0.3.2-1] <https://github.com/ARGOeu/argo-ncg/releases/tag/0.3.2-1>
* **03/04/2017**
	+ ARGO-Connectros [Version 1.5.8-1] <https://github.com/ARGOeu/argo-egi-connectors/releases/tag/v1.5.8>
* **17/03/2017**
	+ ARGO-Connectros [Version 1.5.6-1] <https://github.com/ARGOeu/argo-egi-connectors/releases/tag/v1.5.6>
* **06/03/2017**
	+ ARGO-Connectros [Version 1.5.4-1] <https://github.com/ARGOeu/argo-egi-connectors/releases/tag/V1.5.4-1>
* **03/03/2017**
	+ ARGO-Monitoring Engine [Version 0.3.1-1] <https://github.com/ARGOeu/argo-ncg/releases/tag/0.3.1-1>
* **16/02/2017**
	+ ARGO-Monitoring Engine [Version 0.3.0-1] <https://github.com/ARGOeu/argo-ncg/releases/tag/0.3.0-1>
* **09/02/2017**
	+ ARGO Web UI [Version 1.3.6-1] <https://github.com/ARGOeu/argo-egi-web/releases/tag/v1.3.6-1>
* **30/01/2017**
	+ ARGO Web UI [Version 1.3.5-1] <https://github.com/ARGOeu/argo-egi-web/releases/tag/v1.3.5-1>
* **17/01/2017**
	+ ARGO Compute Engine [Version 1.6.9-1] <https://github.com/ARGOeu/argo-compute-engine/releases/tag/v1.6.9-1>
* **10/01/2017**
	+ ARGO-Poem [Version 1.0.3-1] <https://github.com/ARGOeu/poem/releases/tag/v1.0.3-1>

## Feedback on satisfaction

The ARGO product team uses a development process based around GitHub, which includes procedures that guarantee a high quality of software releases. For details of the ARGO development process, see Appendix I.

## Plan for Exploitation and Dissemination

*This section should provide a plan for exploitation and dissemination (PEDR) of the project results documented in this deliverable. If a plan was already provided in an earlier deliverable, then this plan should provide an update. The content will be used to update the catalogue of project results (*[*http://go.egi.eu/egi-engage-results*](http://go.egi.eu/egi-engage-results)*) and to develop an overall PEDR for the whole project.* ***You can create as many tables as the number of results being described.***

|  |  |
| --- | --- |
| *Name of the result* | ARGO |
| *DEFINITION*  |
| *Category of result* | Software & service innovation |
| *Description of the result* | Software enhancement: improve the portal designing new and easier way to access and visualise data for the final users and exposing a complete API allowing third parties to gather accounting data from the system.Stability and performance improvements of the central ARGO Monitoring Service. NGI instances were decommissioned or kept for NGI’s internal purposes. In addition, specific monitoring instances like opsmon were decommissioned and all probes were integrated into central ARGO Monitoring Service. A/R calculations are performed solely by using results from the central ARGO Monitoring Service. Uncertified instances are also monitored via the centralized ARGO Monitoring Service. Two additional centralized ARGO Monitoring Services were deployed for testing and verification of new probes and for monitoring internal ARGO components.Centralized ARGO Monitoring Service poses a risk if only one instance is deployed. In case of failure of that instance, the whole infrastructure will not be monitored. Therefore, a high availability setup is used.The reorganization of the Compute Engine to support stream processing in real time is one of the key new factors. A new streaming layer has been introduced. Monitoring results flow through the AMS to the streaming layer (in parallel to the HDFS). The streaming layer is used in order to push raw metric results to the metric result store and to compute status results and push them to the status store in real-time. The streaming and batch job for the status results is running in the devel infrastructure producing the same results as the production infrastructure. At the same time, the new AMS publisher has been introduced. It is a new component acting as bridge from Nagios to the new ARGO Messaging system. It is successfully running on the devel infrastructure for a while producing the same results as on production. It is integral part of the software stack running on ARGO monitoring instance and is responsible for forming and dispatching messages that are results of Nagios tests. Thanks to the new real-time Streaming processing layer, we are now able to introduce new functionality to the ARGO Monitoring Service that goes beyond infrastructure monitoring, as, for example, the alerting. We are working on a new component on top of the streaming engine. This component will analyse the monitoring results and send notification based on a set of rules. The minimum set of rules support should mimic the Nagios behaviour. |
| *EXPLOITATION* |
| *Target group(s)* | RIs, Service providers, Users, NGIs, Resource centres |
| *Needs* | * Used for the Availability and Reliability monitoring
* Provide complete API allowing third parties to gather data from the system.
* Used as a source of alerts for resource centres administrators through the Operations Portal Dashboard
* Used for middleware versions monitoring and upgrade campaigns
 |
| *How the target groups will use the result?* | The ARGO Availability and Reliability Monitoring Framework is used by the ARGO Monitoring Service that is operated by EGI for the monitoring of the availability and reliability of the EGI infrastructure. The ARGO Monitoring Service can be provided also to research communities and other infrastructures as a service in order to monitor the status, availability and reliability of their services. |
| *Benefits* | The developments during this period, allowed EGI to replace the older implementation of the SAM Nagios Monitoring Engine, which required one monitoring engine per NGI, with a new implementation using the ARGO Monitoring Engine, which provided a monitoring engine that could deliver monitoring probe scheduling and execution as a service for all the NGI and communities. Central ARGO requires less maintenance effort and enables faster and streamlined deployment of new tests or update of existing tests. This leads to improvements in the performance, robustness and reliability of the ARGO Monitoring Service.Furthermore, real-time computations allows the ability to take immediate action for urgent issues. The goal is to obtain the insight required to act prudently at the right time - which increasingly means immediately. |
| *How will you protect the results?* | The ARGO Monitoring Framework is released under the Apache 2.0 license. |
| *Actions for exploitation* | The new version of the ARGO Monitoring Framework has already been adopted by the production ARGO Monitoring Service. In order to further exploit the results, we should promote the service also to research communities and other infrastructures that can benefit of its features. |
| *URL to project result* | <http://argo.egi.eu/> <https://github.com/ARGOeu/>  |
| *Success criteria* | The deployment of the results to the production EGI infrastructure. The usage of the service to monitor third party services. |
| *DISSEMINATION* |
| *Key messages* | Offer a guaranteed quality of services. |
| *Channels* | EGI Broadcast tool, EGI Meetings. |
| *Actions for dissemination* | EGI conferences, publications, participation to workshops organised by potential users |
| *Cost* |  |
| *Evaluation* | The number of requests for information is the main way to evaluate the impact of the dissemination actions. |

## Future plans

**ARGO Compute Engine**

* Streaming processing;
* Alerting mechanism;
* Separation of A/R and Metric stores;
* Stability and performance improvements.

**ARGO Monitoring Engine**

* Finalize support for GOCDB as a single support of topology;
* Integration with probe management feature in POEM;
* Use of the messaging API on production;
* Fedcloud probes updates;
* Stability and performance improvements.

**ARGO Web UI**

* UI Enhancements.

**ARGO EGI Consumers and Connectors**

* Decommission of Consumer and use ARGO nagios AMS-publisher instead;
* Finalize the use of the messaging API for Connectors component on production;
* Stability and performance improvements.

**ARGO POEM**

* Finalize the probe management feature;
* Connect to the EGI IdP/SP Proxy;
* Stability and performance improvements.

# Messaging Service

## Introduction

|  |  |
| --- | --- |
| **Tool name** | ARGO Messaging Service |
| **Tool url** | <http://argoeu.github.io> |
| **Tool wiki page** | <https://wiki.egi.eu/wiki/Message_brokers>  |
| **Description**  | The Messaging service enables reliable asynchronous messaging for the EGI infrastructure.  |
| **Value proposition** | e-Infrastructures and research communities are building distributed services and workflows in order to satisfy their operational and research requirements. Synchronization between services, gathering of telemetry, monitoring and accounting data any secure messages exchange is a core requirement in any type of distributed services. The Messaging Service provides an easy to use and reliable transport layer for the secure exchange of messages between services such as accounting data, monitoring data, event notifications, etc. |
| **Customer of the tool** | EGI; NGI; RI; Resource Provider; Research Communities |
| **User of the service** | Site admins; Operations Managers; Large research group |
| **User Documentation**  | <http://argoeu.github.io>; |
| **Technical Documentation**  | <http://argoeu.github.io> |
| **Product team** | GRNET, SRCE |
| **License** | Apache License Version 2.0 |
| **Source code** | <https://github.com/ARGOeu/>  |

## Service architecture

### High-Level Service architecture

The Messaging service enables reliable asynchronous messaging for the EGI infrastructure. The current implementation of the Messaging service relies on a Message Broker Network of ActiveMQ services and uses the STOMP protocol for the publication and consumption of messages.



Figure 5. Messaging service architecture.

During the project, we have developed a new version of the Messaging service that is going to replace the STOMP interface with an HTTP one, which will make the implementation of new clients easier and more robust. The new ARGO Messaging Service is a real-time messaging service that allows you to send and receive messages between independent applications.

The ARGO Messaging Service is a Publish/Subscribe Service, which implements the Google PubSub protocol. It provides an HTTP API that enables users/systems to implement a message-oriented service using the Publish/Subscribe Model over plain HTTP. Publishers are users/systems that can send messages to named-channels called Topics. Subscribers are users/systems that create Subscriptions to specific topics and receive messages.

It supports both push and pull message delivery. In push delivery, the Messaging Service initiates requests to your subscriber application to deliver messages. In pull delivery, your subscription application initiates requests to the server to retrieve messages.



Figure 6. The new ARGO messaging service.

#### AMS Metrics

The AMS Pub/Sub API exports usage metrics that can be monitored programmatically. The list of available metrics is the following:

* Memory usage per AMS instance: percentage value that displays the Memory usage of AMS service in the specific node;
* CPU usage per AMS instance: percentage value that displays the CPU usage of AMS service in the specific node;
* Messages published per topic/project/user:
	+ Counter that displays the number of messages published to the specific topic (per project and per user);
* Messages delivered per topic/subscription/project/user:
	+ Counter that displays the number of messages delivered to the specific subscription (per project, per user and per topic);
* Bytes in/out per topic/subscription/project/user:
	+ Counter that displays the total size of data (in bytes) published to the specific topic;
	+ Counter that displays the total size of data (in bytes) consumed from the specific subscription;
* Topics per project/user:
	+ Counter that displays the number of topics belonging to the specific project;
	+ Counter that displays the number of topics belonging to the specific user;
* Subscriptions per project/topic/user:
	+ Counter that displays the number of subscriptions belonging to the specific project:
	+ Counter that displays the number of subscriptions that a user has access to the specific project:
	+ Counter that displays the number of subscriptions belonging to the specific topic;
	+ Counter that displays the number of subscriptions belonging to the specific user.

#### Operational Metrics[[9]](#footnote-9)

The Operational Metrics mainly include metrics related to the CPU or memory usage of the AMS nodes. The list of operational metrics is the following:

* Memory usage per AMS instance;
* CPU usage per AMS instance;
* Messages published per topic/project/user;
* Messages delivered per topic/subscription/project/user;
* Bytes in/out per topic/subscription/project/user;
* Topics per project/user;
* Subscriptions per project/topic/user.

#### Accounting

The list of accounting metrics is the following:

* Number of messages published per topic/project/user;
* Number of messages delivered per topic/subscription/project/user;
* Number of topics per project/user;
* Number of subscriptions per project/topic/user.

#### AMS - Library[[10]](#footnote-10)

A simple python library for interacting with the ARGO Messaging Service.

The Messaging Service is implemented as a Publish/Subscribe service. Instead of focusing on a single Messaging API specification for handling the logic of publishing/subscribing to the broker network, the API focuses on creating nodes of Publishers and Subscribers as a Service.

In the Publish/Subscribe paradigm, Publishers are users/systems that can send messages to named-channels called Topics. Subscribers are users/systems that create Subscriptions to specific topics and receive messages.

You may find more information in the ARGO Messaging Service documentation[[11]](#footnote-11).

### Integration and dependencies

The following EGI Core Services rely on the EGI Messaging Service:

* ARGO Availability and Reliability Monitoring Service
* Accounting system
* Operations Portal

All these services are using the EGI Message Broker network today. The ARGO Monitoring Service is already implementing a connector for the new Messaging Service. Accounting and Operations portal are expected to also complete the implementation of their own interfaces to the new Messaging Service, within the timeframe of the EGI-Engage project.

The Messaging Service does not have any dependencies to other services now.

## Release notes

### Requirements covered in the release

* APIv1 test implementation;
* APIv1 final implementation;
* APIv1 final specification;
* Support APEL to use Messaging Service;
* Support appDB to use Messaging Service;
* Support Operational Portal to use Messaging Service;
* Message Service Accounting: Metrics for Messaging Service;
* Operational statistics;
* Usage Statistics;
* Stability and performance improvements.

###  Changelog

* **28/06/2017**
	+ **AMS Library [Version 0.3.0-1]** <https://github.com/ARGOeu/argo-ams-library/releases/tag/v0.3.0-1>
* **08/06/2017**
	+ **AMS Library [Version 0.2.0-1]** <https://github.com/ARGOeu/argo-ams-library/releases/tag/v0.2.0-1>
* **25/10/2016**
	+ **ARGO - Messaging Service [v1.0.0-1]** [**https://github.com/ARGOeu/argo-messaging/releases/tag/v1.0.0-1**](https://github.com/ARGOeu/argo-messaging/releases/tag/v1.0.0-1)

## Feedback on satisfaction

The ARGO product team uses a development process based around GitHub, which includes procedures that guarantee a high quality of software releases. For details of the ARGO development process, see Appendix I.

## Plan for Exploitation and Dissemination

|  |  |
| --- | --- |
| *Name of the result* | ARGO Messaging Service |
| DEFINITION  |
| *Category of result* | Software & service innovation |
| *Description of the result* | In the new version of the Messaging Service, the STOMP interface has been replaced with an HTTP interface, which makes the implementation of new clients easier and the implementation more robust. This new ARGO Messaging Service is a real-time messaging service that allows services to asynchronously send and receive messages using the Publish/Subscribe model. |
| EXPLOITATION |
| *Target group(s)* | RIs, Service providers, Users, NGIs, Resource centres, EGI Accounting service and the Operations Portal |
| *Needs* | e-Infrastructures and research communities are building distributed services and workflows in order to satisfy their operational and research requirements. Synchronization between services, gathering of telemetry, monitoring and accounting data any secure messages exchange is a core requirement in any type of distributed services. The Messaging Service provides an easy to use and reliable transport layer for the secure exchange of messages between services such as accounting data, monitoring data, event notifications, etc.  |
| *How the target groups will use the result?* | Infrastructure architects that need to design distributed architectures that require a robust and easy to use messaging backbone, which can scale to billions of messages. |
| *Benefits* | The ARGO Messaging service offers the following features: * Simple HTTP API for client access;
* An easy to use python library;
* Operations & usage metrics ;
* Transparent scalability & high availability;
* Access controls implemented at the API layer;
* Multi tenant support;
* Performance robustness.
 |
| *How will you protect the results?* | The ARGO Messaging service is released under the Apache 2.0 license. |
| *Actions for exploitation* | * Promote the service to other research communities and infrastructures that can benefit of its features.
* Provide the necessary documentation (all, for a publisher, or for a subscriber)
* Create test accounts per target group to publish messages to topics, or to consume messages as subscribers from a topic.
 |
| *URL to project result* | <http://argo.egi.eu/> <https://github.com/ARGOeu/>  |
| *Success criteria* | * The ARGO Messaging Service should be operated as a production EGI service.
* All the EGI tools services should have migrated from the old Messaging Broker service to the new ARGO Messaging service.
 |
| DISSEMINATION |
| *Key messages* | Interconnect your distributed services in a ease and efficient manner. |
| *Channels* | * Dissemination through the EGI conferences
* Article featured in the EGI newsletter
 |
| *Actions for dissemination* | EGI conferences, publications, participation to workshops organised by potential users |
| *Cost* |  |
| *Evaluation* | The number of requests for information, and/or accounts (either test or production) is the main way to evaluate the impact of the dissemination actions. |

## Future plans

* Move to production
* Stability and performance improvements

# GOCDB

## Introduction

|  |  |
| --- | --- |
| **Tool name** | GOCDB |
| **Tool url** | <https://goc.egi.eu>  |
| **Tool wiki page** | <https://wiki.egi.eu/wiki/GOCDB> |
| **Description** | GOCDB is a central registry to record information about the topology of an e-Infrastructure. This includes entities such as resource centers (sites), services, service-endpoints and their downtimes, contact information and roles of users responsible for operations at different levels. The service enforces a number of business rules and defines different grouping mechanisms including object-tagging for the purposes of fine-grained resource filtering. |
| **Value proposition** | The Extensions to the write API will greatly increase the ability for external tools to interact in a programmatic way with GOCDB. This will make GOCDB more viable for the future and reduce the need for other information systems.  |
| **Customer of the tool** | EGI Operations and WLCG  |
| **User of the service** | Site/service admins, NGI managers and Security teams. |
| **User Documentation**  | <https://wiki.egi.eu/wiki/GOCDB> |
| **Technical Documentation**  | <https://wiki.egi.eu/wiki/GOCDB> |
| **Product team** | STFC |
| **License** | Apache 2  |
| **Source code** | <https://github.com/GOCDB/gocdb> |

## Service architecture

### High-Level Service architecture

GOCDB is a central information repository providing a web portal interface for CRUD operations, and a REST API for data queries.

It is a definitive information source where data is directly populated and managed in the system. Because GOCDB is a primary data-input source, the portal applies a range of business rules and data-validations to control input. It applies a comprehensive Role-based authorization model that enables different actions over different target resources. The Role model allows communities to manage their own resources where users with existing roles can approve or reject new role-requests.

It is intentionally designed to have no dependencies on other operational tools (other than the EGI CheckIn service described below). For example, it does not query other systems to populate its core data model. The underling Oracle DB is hosted by the STFC DB Services Team with nightly tape backups. An additional failover instance is hosted at a second STFC site (Daresbury Labs). The failover instance is synchronized hourly against the production data.

The previous release, introduced a new dependency on the EGI CheckIn service in order to provide federated access to GOCDB for users without client certificates. However, for users with certificates there continues to be no dependencies on other operational tools. Other than the extensions to the capability to the write API, this release brings no major alterations to the architecture.

### Integration and dependencies

GOCDB newly depends on the EGI CheckIn service to provide federated authentication and access without client certificates. When accessed using a client certificate, GOCDB continues to depend on no other tool.

## Release notes

### Requirements covered in the release

By August the Write API will have been extended to meet requirements of WLCG[[12]](#footnote-12). This will allow programmatic:

* Creation, update, and deletion of service endpoints
* Update of details of services

There will be no changes to the way authentication and authorisation for the write API since the previous release. Building upon the previous release, these updates allow changes to key entities within GOCDB programmatically. This represents a significant change in the way in which GOCDB works, allowing for much greater automated interaction with the information managed by GOCDB. This will help secure GOCDBs future in an evolving information space.

 A number of smaller bugs[[13]](#footnote-13) will also have been addressed.

## Feedback on satisfaction

Before every production release, GOCDB development is frozen and a period of testing is announced that lasts for approximately two weeks to one month using the GOCDB test instance[[14]](#footnote-14). This testing phase is widely disseminated using the relevant mail lists, and all operational tools and users are invited to perform tests against this instance. Recent GOCDB releases successfully passed this stage.

The GOCDB development process is described in Appendix II.

## Plan for Exploitation and Dissemination

|  |  |
| --- | --- |
| *Name of the result* | GOCDB |
| *DEFINITION*  |
| *Category of result* | Software & service innovation |
| *Description of the result* | * Extension of the write API.
 |
| *EXPLOITATION* |
| *Target group(s)* | WLCG tool developers, ARGO service, Resource/service provider admins and NGI managers  |
| *Needs* | The extension to the Write API will enable communities (e.g. WLCG) to further automate their interactions with the GOCDDB and move away from other information sources.  |
| *How the target groups will use the result?* | The results are integrated into the production instance of GOCDB, on which much of the target group’s infrastructure relies.  |
| *Benefits* | The result will improve the efficiency of target group’s use of the GOCDB service, as well as ensure its continuing fitness to serve them. |
| *How will you protect the results?* | Apache 2 licence |
| *Actions for exploitation* | The code needs to be integrated into the production instance of the GOCDB in order to provide the described functionality. The full source code will be available for use (under the Apache 2 licence) at <https://github.com/GOCDB/gocdb> |
| *URL to project result* | [https://github.com/GOCDB/gocdb/releases/tag/5.8](https://github.com/GOCDB/gocdb/releases/tag/5.7)[[15]](#footnote-15) <https://goc.egi.eu/>  |
| *Success criteria* | Regular use of the write API extension by at least one tool. |
| *DISSEMINATION* |
| *Key messages* | The Write API has now been extended to have greater functionality. |
| *Channels* | WP3 meetings, EGI OMB meetings, WLCG Information Systems Evolution Task Force |
| *Actions for dissemination* | Announcement emails to multiple EGI mailing lists and WLCG information system evolution mailing list.Description of new features to EGI Conference (May 2017: <https://indico.egi.eu/indico/event/3249/session/32/contribution/31>. |
| *Cost* |  |
| *Evaluation* | Uptake of use of new features. |

## Future plans

* Data freshness check[[16]](#footnote-16);
* Replacement of the GOCDB UI with a modern Web framework;
* Extending GOCDB in the info-service space supporting dynamic attributes;
* Improve change logging.

# Security Monitoring

## Introduction

|  |  |
| --- | --- |
| **Tool name** | Secant |
| **Tool url** | <https://github.com/CESNET/secant>  |
| **Tool wiki page** | <https://wiki.egi.eu/wiki/Tools>  |
| **Description** | Secant is a framework to detect security vulnerabilities in images of virtual machines. It tries to detect the most common security issues that often lead to incidents and prevent them from appearing in the context of EGI cloud facilities. |
| **Value proposition** | Security incidents may cause significant problems for users, service providers and infrastructure operators. Secant was designed to detect common weakness in virtual appliances so that these can be fixed before they threaten a production infrastructure. |
| **Customer of the tool** | Cloud providers, VA owners, EGI operations, the EGI CSIRT |
| **User of the service** | Administrators, operators, security staff |
| **User Documentation**  | <https://github.com/CESNET/secant>  |
| **Technical Documentation**  | <https://github.com/CESNET/secant>  |
| **Product team** | CESNET |
| **License** | Apache License Version 2.0 |
| **Source code** | <https://github.com/CESNET/secant>  |

## Service architecture

### High-Level Service architecture

Secant runs as a service that periodically checks for new images available in a repository and performs their security assessment. When a new image becomes available in the system, it is taken by Secant and checked for security vulnerabilities. In order to perform the security checks, Secant instantiates a virtual machine from the appliance that is being verified and performs two phases of security checks. During the first phase, Secant launches a series of external scans that tries to detect vulnerabilities exposed by the machine to the Internet. Following these tests, and if the machine supports that, Secant runs a series of internal probes on the virtual machine, which checks security properties of the installed software. Both internal and external probes are modular and new tests can be easily added when needed. After the probes are executed, Secant processes the results and generated the assessment.

### Integration and dependencies

Secant needs to integrate support of a cloud management framework, which enables to both manage virtual machines and maintain the list of images to assess. The current implementation supports OpenNebula for the management of virtual machine and uses the EGI CloudKeeper[[17]](#footnote-17) to maintain the list of images and templates in the cloud repository.

In order to facilitate integration with existing infrastructure services, support for a messaging has been introduced recently. Secant uses the ARGO messaging to consume information about available images and to deliver assessment reports once assessment has been finished.

## Release notes

### Requirements covered in the release

Following the principles of continuous delivery, Secant does not have fixed releases. The outcomes of recent development are always available from the pilot installation deployed at CESNET. The features introduced recently involve integration of the EGI Messaging, support of CloudKeeper, and integration work aiming at utilization of Secant for the Application Database and EGI endorsement of virtual appliances.

## Feedback on satisfaction

Secant runs in a piloting environment established at CESNET and its MetaCloud site. The development follows expectations of the EGI CSIRT team and the service was presented to the team several times. Since the integration works are on-going, assessment tests can only be executed manually. So far, several dozens of virtual appliances underwent testing by Secant and findings were incorporated by the developers.

## Plan for Exploitation and Dissemination

|  |  |
| --- | --- |
| *Name of the result* | Secant |
| *DEFINITION*  |
| *Category of result* | Software & service innovation |
| *Description of the result* | Secant is a framework to detect security vulnerabilities in images of virtual machines. It tries to detect the most common security issues that often lead to incidents and prevent them from appearing in the context of EGI cloud facilities. |
| *EXPLOITATION* |
| *Target group(s)* | Users, RIs, Resource centres, NGIs, security teams, VA endorsers. |
| *Needs* | Prevent from security incidents that misuse common vulnerabilities exposed by servers connected to the Internet. |
| *How the target groups will use the result?* | The tools will facilitate the endorsement process and will help the endorsers detect common weaknesses. The tools will also be available to users preparing their images or installations on the top of running virtual machines. |
| *Benefits* | Achieving a common security bottom line of virtual machines in clouds, based on shared knowledge and tooling. |
| *How will you protect the results?* | The tool is released under a standard open-source license. |
| *Actions for exploitation* | Secant will be freely available and its utilization documented. |
| *URL to project result* | <https://github.com/CESNET/secant>  |
| *Success criteria* | Availability of the tool for performing assessments. |
| *DISSEMINATION* |
| *Key messages* | Secant help identify common security vulnerabilities in virtual appliances. |
| *Channels* | EGI Conferences, meetings with cloud experts. |
| *Actions for dissemination* | Integration with AppDB will facilitate the introduction of the assessment in the endorsement process. |
| *Cost* |  |
| *Evaluation* | Utilization of Secant in endorsement process. |

## Future plans

After Secant has been fully integration with AppDB, it will be necessary to overview the endorsement process to support the assessment. We will need to take into account emerging technologies (like containers) to examine their impact on the assessment process.

# Accounting Repository

## Introduction

|  |  |
| --- | --- |
| **Tool name** | *Please provide clear, short name of tool* |
| **Tool url** | *Please provide url if applicable* |
| **Tool wiki page** | *Link to EGI wiki with description of the product**For JRA1 from https://wiki.egi.eu/wiki/Tools* |
| **Description** | *Please provide a high-level description of what the service does and functionality included.* |
| **Value proposition** | *Describe  how  the  new  or changed  service  alleviates specific user pains and/or supports its intended customer(s) to exploit new opportunities* |
| **Customer of the tool** | *Please describe customer of the tool. Customer commissions the service provider to receive the service, doing so on behalf of a number of users – specify the organisation type/category of the service e.g. NGI; RI; Resource Provider* |
| **User of the service** | *Please specify the user type/category of the service e.g. large research groups; individual researcher; site admins* |
| **User Documentation**  | *Please provide url* |
| **Technical Documentation**  | *Please provide url* |
| **Product team** | *Please provide institutions taking part in development* |
| **License** | *Please provide license*  |
| **Source code** | *Please provide url* |

## Service architecture

*The service architecture provides an overview of the key (logical) service components and their dependencies to help better understand the structure and logical as well as technical setup of the service.*

### High-Level Service architecture

*These sections describe how the service is built. If already described in technical documentation please provide link.*

*Highlight and shortly describe any change on the service architecture introduced by this release.*

### Integration and dependencies

*Insert a description and/or visualisation (figure) of the dependencies to other tools.*

*If already described in technical documentation please provide link.*

*Highlight and shortly describe any change on the dependencies to other tools introduced by this release.*

## Release notes

### Requirements covered in the release

*List requirements that have been implemented in the release*

## Feedback on satisfaction

*Who was involved in testing and what the outcome of the review was*

## Plan for Exploitation and Dissemination

*This section should provide a plan for exploitation and dissemination (PEDR) of the project results documented in this deliverable. If a plan was already provided in an earlier deliverable, then this plan should provide an update. The content will be used to update the catalogue of project results (*[*http://go.egi.eu/egi-engage-results*](http://go.egi.eu/egi-engage-results)*) and to develop an overall PEDR for the whole project.* ***You can create as many tables as the number of results being described.***

|  |  |
| --- | --- |
| *Name of the result* | *Short name for the result (results generated under the project could be any tangible or intangible output, more particularly data, knowledge or information whatever its form or nature, whether it can be protected or not.)* |
| *DEFINITION*  |
| *Category of result* | * *Technical input to standards: Technical specifications or extensions to standards adopted within the project*
* *Policy & Procedure developments: Technical procedures directed at users, service and infrastructure providers (for example to govern access and allocation to resources), policy reports and recommendations, and strategic analysis*
* *Software & service innovation: Software developments: (e.g.: workflows, Virtual Machines, applications), new software services deployed for the direct benefit of researchers (e.g.: web portals, gateways), e-Infrastructure Commons such as accounting, AAI, and the Federated Cloud platform and the Open Data platform, demonstrators and prototypes.*
* *Business model innovation: Business and sustainability-related outputs (the EGI Service Marketplace concept, the contribution to the Innovation space for the big data value chain, sustainability plans, pay-for-use models)*
* *Know-how: Includes all results from fact-finding activities (e.g. surveys, requirement gathering), but also the results from internal exercises (e.g. security challenges) and outputs that can be used for knowledge transfer as training materials.*
 |
| *Description of the result* | *Description of the result*  |
| *EXPLOITATION* |
| *Target group(s)* | *Describe who will use those results. Es: RIs, international research collaborations and the long-tail of science, industry/SMEs, service providers, Funding agencies and decision/policy makers, Standardisation bodies"* |
| *Needs* | *What are the needs of the target groups that the results aims to fulfil?* |
| *How the target groups will use the result?* | *How the project result will be used? How are you going to achieve the best benefits from the project outcomes? How can you make sure the results they owned are used:** *in further research activities other than those covered by the project concerned*
* *in developing, creating and marketing a product or process*
* *in creating and providing a service*
* *in standardisation activities*

*Note: The exploitation does not need necessarily to be done by participants, who may prefer to ensure its use by another entity. Such indirect exploitation can be performed by licensing the results or assigning them to third parties, in accordance with the requirements established in the grant agreement "* |
| *Benefits* | *What are the expected benefits of the result when this will be used by the target groups?* |
| *How will you protect the results?* | *Protection of results is indeed essential in Horizon 2020, since an effective exploitation depends on it. Thus, participants must assess the possibility of protecting their results once these are generated. Please, describe what IP protection approach will you put in place for this result. This can range from simple attribution via open source license to full copyright for commercially exploitable results. (For more information you can read “How to manage IP in Horizon 2020: project implementation and conclusion”* [*https://www.iprhelpdesk.eu/sites/default/files/newsdocuments/FS\_IP\_Management\_h2020\_implementation\_0.pdf*](https://www.iprhelpdesk.eu/sites/default/files/newsdocuments/FS_IP_Management_h2020_implementation_0.pdf) |
| *Actions for exploitation* | *Please, describe the concrete actions that need to be executed to make the result reusable by the target group (e.g., for a software, this can include software packaging for distribution, documentation for the installation, etc). Once executed, the target groups should be able to use the results without barriers.* |
| *URL to project result* | *Link where the result will be made available* |
| *Success criteria* | *What are the success criteria in terms of adoption by the end of the project?* |
| *DISSEMINATION* |
| *Key messages* | *What messages will you tell to the target groups when informing about the results?* |
| *Channels* | *What channels will you use to deliver the messages to the target?(e.g. Scientific publications, EGI web site, EGI newsletter, participation in conferences or trade fairs)* |
| *Actions for dissemination* | *Describe the concrete set of actions that will be put in place to disseminate this project output. When this result is ready, how will you reach to target group to ensure uptake of the result? (You can list the preliminary list of events where you plan to promote the results or material that will be produced or any other concrete actions that will be put in place during the project)* |
| *Cost* | *What is the expected cost of dissemination actions?* |
| *Evaluation* | *How will you evaluate the impact of the dissemination actions?* |

## Future plans

# Accounting Portal

## Introduction

|  |  |
| --- | --- |
| **Tool name** | *Please provide clear, short name of tool* |
| **Tool url** | *Please provide url if applicable* |
| **Tool wiki page** | *Link to EGI wiki with description of the product**For JRA1 from https://wiki.egi.eu/wiki/Tools* |
| **Description** | *Please provide a high-level description of what the service does and functionality included.* |
| **Value proposition** | *Describe  how  the  new  or changed  service  alleviates specific user pains and/or supports its intended customer(s) to exploit new opportunities* |
| **Customer of the tool** | *Please describe customer of the tool. Customer commissions the service provider to receive the service, doing so on behalf of a number of users – specify the organisation type/category of the service e.g. NGI; RI; Resource Provider* |
| **User of the service** | *Please specify the user type/category of the service e.g. large research groups; individual researcher; site admins* |
| **User Documentation**  | *Please provide url* |
| **Technical Documentation**  | *Please provide url* |
| **Product team** | *Please provide institutions taking part in development* |
| **License** | *Please provide license*  |
| **Source code** | *Please provide url* |

## Service architecture

*The service architecture provides an overview of the key (logical) service components and their dependencies to help better understand the structure and logical as well as technical setup of the service.*

### High-Level Service architecture

*These sections describe how the service is built. If already described in technical documentation please provide link.*

*Highlight and shortly describe any change on the service architecture introduced by this release.*

### Integration and dependencies

*Insert a description and/or visualisation (figure) of the dependencies to other tools.*

*If already described in technical documentation please provide link.*

*Highlight and shortly describe any change on the dependencies to other tools introduced by this release.*

## Release notes

### Requirements covered in the release

*List requirements that have been implemented in the release*

## Feedback on satisfaction

*Who was involved in testing and what the outcome of the review was*

## Plan for Exploitation and Dissemination

*This section should provide a plan for exploitation and dissemination (PEDR) of the project results documented in this deliverable. If a plan was already provided in an earlier deliverable, then this plan should provide an update. The content will be used to update the catalogue of project results (*[*http://go.egi.eu/egi-engage-results*](http://go.egi.eu/egi-engage-results)*) and to develop an overall PEDR for the whole project.* ***You can create as many tables as the number of results being described.***

|  |  |
| --- | --- |
| *Name of the result* | *Short name for the result (results generated under the project could be any tangible or intangible output, more particularly data, knowledge or information whatever its form or nature, whether it can be protected or not.)* |
| *DEFINITION*  |
| *Category of result* | * *Technical input to standards: Technical specifications or extensions to standards adopted within the project*
* *Policy & Procedure developments: Technical procedures directed at users, service and infrastructure providers (for example to govern access and allocation to resources), policy reports and recommendations, and strategic analysis*
* *Software & service innovation: Software developments: (e.g.: workflows, Virtual Machines, applications), new software services deployed for the direct benefit of researchers (e.g.: web portals, gateways), e-Infrastructure Commons such as accounting, AAI, and the Federated Cloud platform and the Open Data platform, demonstrators and prototypes.*
* *Business model innovation: Business and sustainability-related outputs (the EGI Service Marketplace concept, the contribution to the Innovation space for the big data value chain, sustainability plans, pay-for-use models)*
* *Know-how: Includes all results from fact-finding activities (e.g. surveys, requirement gathering), but also the results from internal exercises (e.g. security challenges) and outputs that can be used for knowledge transfer as training materials.*
 |
| *Description of the result* | *Description of the result*  |
| *EXPLOITATION* |
| *Target group(s)* | *Describe who will use those results. Es: RIs, international research collaborations and the long-tail of science, industry/SMEs, service providers, Funding agencies and decision/policy makers, Standardisation bodies"* |
| *Needs* | *What are the needs of the target groups that the results aims to fulfil?* |
| *How the target groups will use the result?* | *How the project result will be used? How are you going to achieve the best benefits from the project outcomes? How can you make sure the results they owned are used:** *in further research activities other than those covered by the project concerned*
* *in developing, creating and marketing a product or process*
* *in creating and providing a service*
* *in standardisation activities*

*Note: The exploitation does not need necessarily to be done by participants, who may prefer to ensure its use by another entity. Such indirect exploitation can be performed by licensing the results or assigning them to third parties, in accordance with the requirements established in the grant agreement "* |
| *Benefits* | *What are the expected benefits of the result when this will be used by the target groups?* |
| *How will you protect the results?* | *Protection of results is indeed essential in Horizon 2020, since an effective exploitation depends on it. Thus, participants must assess the possibility of protecting their results once these are generated. Please, describe what IP protection approach will you put in place for this result. This can range from simple attribution via open source license to full copyright for commercially exploitable results. (For more information you can read “How to manage IP in Horizon 2020: project implementation and conclusion”* [*https://www.iprhelpdesk.eu/sites/default/files/newsdocuments/FS\_IP\_Management\_h2020\_implementation\_0.pdf*](https://www.iprhelpdesk.eu/sites/default/files/newsdocuments/FS_IP_Management_h2020_implementation_0.pdf) |
| *Actions for exploitation* | *Please, describe the concrete actions that need to be executed to make the result reusable by the target group (e.g., for a software, this can include software packaging for distribution, documentation for the installation, etc). Once executed, the target groups should be able to use the results without barriers.* |
| *URL to project result* | *Link where the result will be made available* |
| *Success criteria* | *What are the success criteria in terms of adoption by the end of the project?* |
| *DISSEMINATION* |
| *Key messages* | *What messages will you tell to the target groups when informing about the results?* |
| *Channels* | *What channels will you use to deliver the messages to the target?(e.g. Scientific publications, EGI web site, EGI newsletter, participation in conferences or trade fairs)* |
| *Actions for dissemination* | *Describe the concrete set of actions that will be put in place to disseminate this project output. When this result is ready, how will you reach to target group to ensure uptake of the result? (You can list the preliminary list of events where you plan to promote the results or material that will be produced or any other concrete actions that will be put in place during the project)* |
| *Cost* | *What is the expected cost of dissemination actions?* |
| *Evaluation* | *How will you evaluate the impact of the dissemination actions?* |

## Future plans

1. ARGO development process
2. GOCDB development process
1. [www.ggus.eu](http://go.egi.eu/eng) [↑](#footnote-ref-1)
2. [https://wiki.egi.eu/wiki/EGI\_CSIRT:Main\_Page](http://go.egi.eu/eng) [↑](#footnote-ref-2)
3. <https://documents.egi.eu/document/2660> [↑](#footnote-ref-3)
4. <https://wiki.egi.eu/wiki/OTAG#Operations_Portal_Advisory_and_Testing_Board> [↑](#footnote-ref-4)
5. <https://avro.apache.org/docs/1.2.0> [↑](#footnote-ref-5)
6. <http://hadoop.apache.org/> [↑](#footnote-ref-6)
7. <http://software.in2p3.fr/lavoisier/> [↑](#footnote-ref-7)
8. <http://w3.hepix.org/benchmarks/doku.php> [↑](#footnote-ref-8)
9. [http://argoeu-devel.github.io/messaging/v1/api\_metrics/](http://argoeu-devel.github.io/messaging/v1/api_metrics/%20)  [↑](#footnote-ref-9)
10. https://github.com/ARGOeu/argo-ams-library [↑](#footnote-ref-10)
11. http://argoeu.github.io/messaging/v1/ [↑](#footnote-ref-11)
12. https://rt.egi.eu/rt/Ticket/Display.html?id=11020 [↑](#footnote-ref-12)
13. from the GitHub bug list: https://github.com/GOCDB/gocdb/issues [↑](#footnote-ref-13)
14. https://gocdb-test.esc.rl.ac.uk [↑](#footnote-ref-14)
15. Link will not be live until release in August [↑](#footnote-ref-15)
16. https://rt.egi.eu/rt/Ticket/Display.html?id=8240 [↑](#footnote-ref-16)
17. https://appdb.egi.eu/store/software/cloudkeeper [↑](#footnote-ref-17)