

**EGI-Engage**

Second release of the accounting and operational tools

D3.10

|  |  |
| --- | --- |
| **Date** | 30 January 2017 |
| **Activity** | [provide] |
| **Lead Partner** | [provide] |
| **Document Status** | DRAFT |
| **Document Link** | https://documents.egi.eu/document/XXX |

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](#_Toc473555865)

[1.1 Introduction 7](#_Toc473555866)

[1.2 Service architecture 8](#_Toc473555867)

[1.2.1 High-Level Service architecture 8](#_Toc473555868)

[1.2.2 Integration and dependencies 10](#_Toc473555869)

[1.3 Release notes 10](#_Toc473555870)

[1.3.1 Operations Portal 4.0 10](#_Toc473555871)

[1.3.2 Operations Portal 4.1 11](#_Toc473555872)

[1.3.3 VAPOR 2.0 12](#_Toc473555873)

[1.3.4 VAPOR 2.1 12](#_Toc473555874)

[1.3.5 VAPOR 2.2 12](#_Toc473555875)

[1.4 Feedback on satisfaction 13](#_Toc473555876)

[1.5 Plan for Exploitation and Dissemination 13](#_Toc473555877)

[1.6 Future plans 15](#_Toc473555878)

[2 ARGO 16](#_Toc473555879)

[2.1 Introduction 16](#_Toc473555880)

[2.2 Service architecture 16](#_Toc473555881)

[2.2.1 High-Level Service architecture 16](#_Toc473555882)

[2.2.2 Integration and dependencies 18](#_Toc473555883)

[2.3 Release notes 19](#_Toc473555884)

[2.3.1 Requirements covered in the release 19](#_Toc473555885)

[2.4 Feedback on satisfaction 22](#_Toc473555886)

[2.5 Plan for Exploitation and Dissemination 22](#_Toc473555887)

[2.6 Future plans 24](#_Toc473555888)

[3 Messaging 26](#_Toc473555889)

[3.1 Introduction 26](#_Toc473555890)

[3.2 Service architecture 26](#_Toc473555891)

[3.2.1 High-Level Service architecture 26](#_Toc473555892)

[3.2.2 Integration and dependencies 28](#_Toc473555893)

[3.3 Release notes 28](#_Toc473555894)

[3.3.1 Requirements covered in the release 28](#_Toc473555895)

[3.3.2 Changelog 29](#_Toc473555896)

[3.4 Feedback on satisfaction 29](#_Toc473555897)

[3.5 Plan for Exploitation and Dissemination 29](#_Toc473555898)

[3.6 Future plans 31](#_Toc473555899)

[4 GOCDB 32](#_Toc473555900)

[4.1 Introduction 32](#_Toc473555901)

[4.2 Service architecture 32](#_Toc473555902)

[4.2.1 High-Level Service architecture 33](#_Toc473555903)

[4.2.2 Integration and dependencies 33](#_Toc473555904)

[4.3 Release notes 33](#_Toc473555905)

[4.3.1 Requirements covered in the release 33](#_Toc473555906)

[4.4 Feedback on satisfaction 33](#_Toc473555907)

[4.5 Plan for Exploitation and Dissemination 33](#_Toc473555908)

[4.6 Future plans 35](#_Toc473555909)

[5 Security Monitoring 36](#_Toc473555910)

[5.1 Introduction 36](#_Toc473555911)

[5.2 Service architecture 36](#_Toc473555912)

[5.2.1 High-Level Service architecture 37](#_Toc473555913)

[5.2.2 Integration and dependencies 37](#_Toc473555914)

[5.3 Release notes 37](#_Toc473555915)

[5.3.1 Requirements covered in the release 37](#_Toc473555916)

[5.4 Feedback on satisfaction 37](#_Toc473555917)

[5.5 Plan for Exploitation and Dissemination 37](#_Toc473555918)

[5.6 Future plans 39](#_Toc473555919)

[6 Accounting Repository 40](#_Toc473555920)

[6.1 Introduction 40](#_Toc473555921)

[6.2 Service architecture 40](#_Toc473555922)

[6.2.1 High-Level Service architecture 41](#_Toc473555923)

[6.2.2 Integration and dependencies 41](#_Toc473555924)

[6.3 Release notes 41](#_Toc473555925)

[6.3.1 Requirements covered in the release 41](#_Toc473555926)

[6.4 Feedback on satisfaction 41](#_Toc473555927)

[6.5 Plan for Exploitation and Dissemination 41](#_Toc473555928)

[6.6 Future plans 43](#_Toc473555929)

[7 Accounting Portal 44](#_Toc473555930)

[7.1 Introduction 44](#_Toc473555931)

[7.2 Service architecture 44](#_Toc473555932)

[7.2.1 High-Level Service architecture 45](#_Toc473555933)

[7.2.2 Integration and dependencies 45](#_Toc473555934)

[7.3 Release notes 45](#_Toc473555935)

[7.3.1 Requirements covered in the release 45](#_Toc473555936)

[7.4 Feedback on satisfaction 45](#_Toc473555937)

[7.5 Plan for Exploitation and Dissemination 45](#_Toc473555938)

[7.6 Future plans 47](#_Toc473555939)

[Appendix I. Related work 48](#_Toc473555940)

**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. |
| **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** | 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*](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

*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.*

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. inally, the data sources are refreshed only as needed and only when an action has been triggered. Last but not least, it is very easy to add a new data source in this model as (depicted in Fig. 1 and Fig. 2) . Nevertheless 2 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 with GGUS or RT services a part of the features of the Operations Portal will be affected. In this specific case: 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.

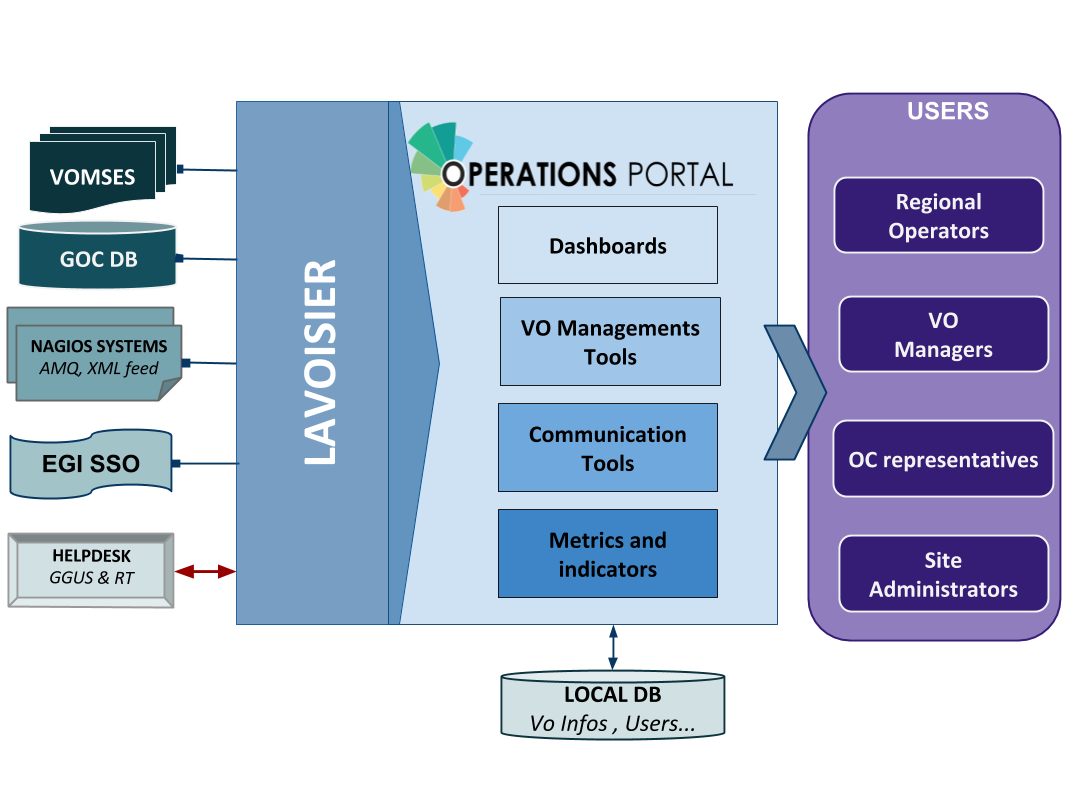


Figure 1. Operations Portal architecture

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.

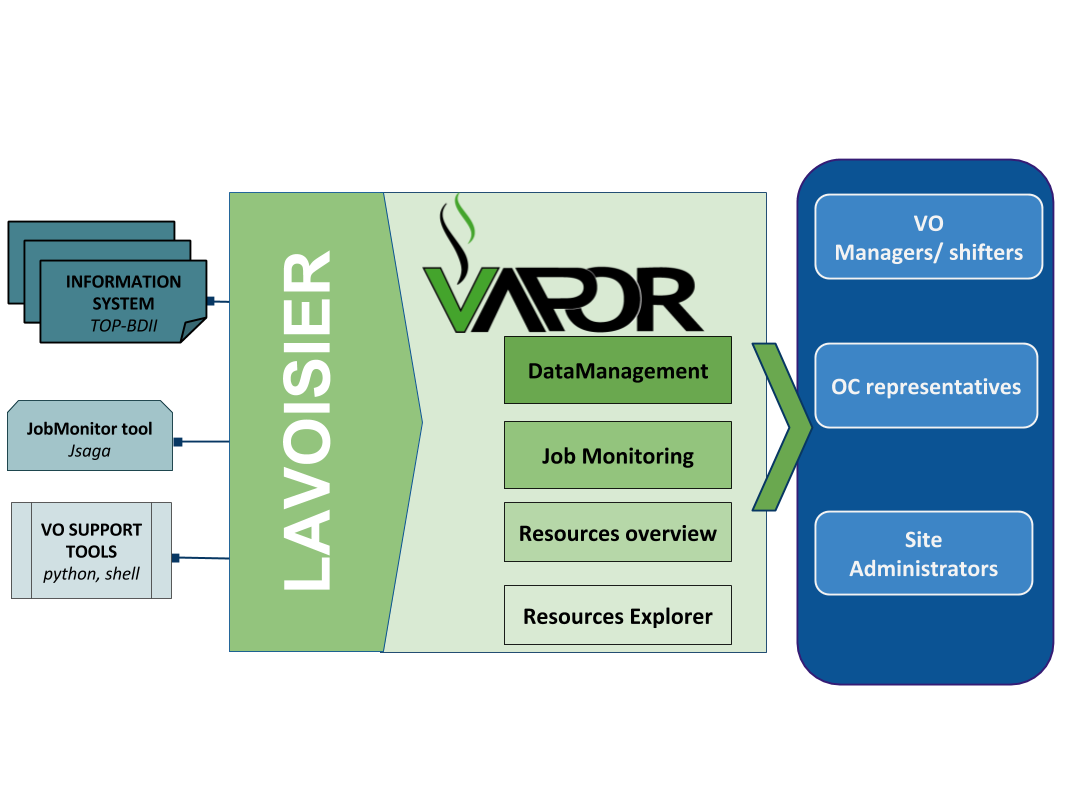


Figure 2. VAPOR architecture

### 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.*

Operations Portal dependencies have been 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 improve the performances . Here are the main changes for this version :

a) Frameworks & JS Libraries :

* migration to Symfony 3.
* upgrade of bootstrap library
* use of Datatables Js libraries to optimize the presentation of the tables (Vo Management, Metrics)
* Use of Google Chart (Vo Management, Metrics)

b) Ergonomics

* Addition of links to ARGO and VAPOR applications
* Changes into global menu presentation (and optimization depending on screen size)

c) Module and project modifications

* Reorganisation of project infrastructure
* Removal of useless files and features
* Merge of Vo Management Tool and VO ID cards (all-in-one page )
* Removal of Availabilities/reliabilities module (replaced by ARGO)

d) New module : Downtime Module

The historical downtime subscription system has been removed and replaced within a dedicated module :

* with a subscription page (emails , rss , ical)
* with timelines charts and tables
* with a search tool
* different formats (CSV, Json)

e) Continuous Integration

* a procedure about good practices for the development procedure  [is in place : https://forge.in2p3.fr/projects/opsportaluser/wiki/Development\_Procedure](https://forge.in2p3.fr/projects/opsportaluser/wiki/Development_Procedure)
* an integration plateform 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 is focused on :

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

### VAPOR 2.0

The initial prototype (described in [D3.4](https://documents.egi.eu/public/RetrieveFile?docid=2660&version=5&filename=EGI-Engage%20D3.4%20FINAL.pdf) ) has been put in production after a test phase of one month.

### VAPOR 2.1

The main features of this release are :

* 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 TopBdii browser :<http://operations-portal.egi.eu/vapor/resources/GL2ResBdiiBrowser>
* New menu
* Corrections of different bugs
* Integration of feedback given by users
* Ergonomics improvements

### VAPOR 2.2

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

For this release we have worked closely with EGI Operations to consolidate the different queries to the Top Bdii and the different extracted figures . The results are :

* a summary of the CPU and storage capacities by countries , sites or Operations Centers
* 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

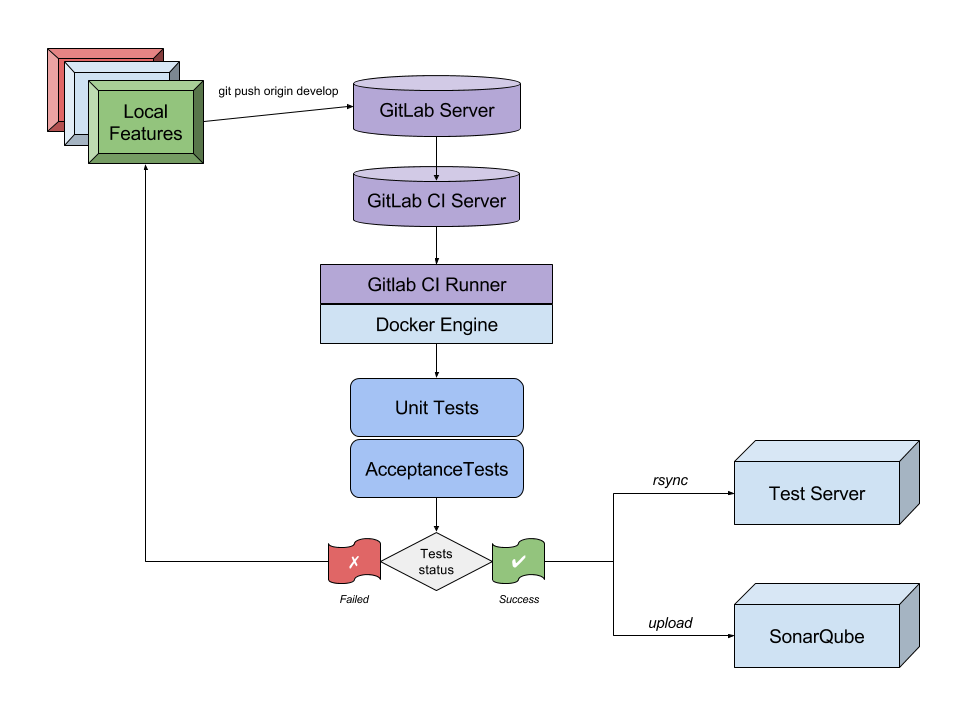
This release has been also focused on the documentation of the different features and the access to the API .

## Feedback on satisfaction

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

Prioritization and testing is done by dedicated [Operations Portal Advisory and Testing Board (OPAnTG)](https://wiki.egi.eu/wiki/OTAG#Operations_Portal_Advisory_and_Testing_Board) coordinated by EGI Operations team . To complete this test phase we have worked on the automation of tests . Unit and acceptance tests are done through docker piloted by GitLab CI server .

If tests are failing they are not propagated to the test infrastructure . Complementary to these tests we use a SonarQBE instance to inspect the quality of code .



## 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* | 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 centers, RIs |
| *Needs* | Monitor / browse / Evaluate the resources for VO, sites, Operations Centers |
| *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 the status of these resources * 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* | Not planned in EGI-Engage |
| *Evaluation* | The number of requests and the feedback given by users |

## Future plans

The effort for EGI-Engage is now limited to the management of the project . Consequently the development will be limited for the last phase of the project. Nevertheless we will continue the current work thanks to the Core Activities fundings

* for VAPOR
  + v2.2 release in production
  + implementation of improvements asked by users
* for the Operations Portal
  + integration of complementary metrics for the VO : accounting data and AppDB changes
  + improvements on the Vo ID Card
  + connection to the EGI IdP/SP Proxy

# ARGO

## Introduction

|  |  |
| --- | --- |
| **Tool name** | ARGO |
| **Tool url** | http://argo.egi.eu |
| **Tool wiki page** | *Link to EGI wiki with description of the product*  *For JRA1 from https://wiki.egi.eu/wiki/Tools* |
| **Description** | ARGO is a flexible and scalable framework for monitoring status, availability and reliability |
| **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** | *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

*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.*

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 integrate with external systems (such as CMDBs, Service Catalogs 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



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

[The ARGO Monitoring Engine](https://github.com/ARGOeu/argo-nagios-egi)

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 addons for the stock Nagios that provide features such as auto-configuration from external information sources, publishing results to external Message Brokers etc

In this last year**, a central ARGO monitoring engine with a high availability setup was deployed** . NGI instances were decommissioned or kept for NGI’s internal purposes. In addition, monitoring instances for middleware versions (midmon) and fedcloud services (cloudmon) were decommissioned and all probes were integrated into central ARGO monitoring engine. A/R calculations are performed solely by using results from the central ARGO monitoring engine.

[The ARGO Connectors](https://github.com/ARGOeu/argo-egi-connectors)

Through the use of custom connectors, ARGO can connect to multiple external Configuration Management Databases and Service Catalogs. Already there are connectors for the EGI and EUDAT e-Infrastructures.

[The ARGO Consumer](https://github.com/ARGOeu/argo-egi-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 before passing to the Compute Engine.

[The ARGO Compute Engine](https://github.com/ARGOeu/argo-compute-engine)

A powerful and scalable analytics engine built on top of Hadoop and HDFS. 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 ARGO Web API](https://github.com/ARGOeu/argo-web-api)

The ARGO Web API provides the Serving Layer of ARGO. It is comprised of a high performance and scalable datastore 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](https://github.com/ARGOeu/argo-egi-web)

The default web UI is based on the [Lavoisier Data Aggregation Framework](http://software.in2p3.fr/lavoisier/).

### 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.*

ARGO can utilize external configuration sources through connectors in order to allow for 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 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 values of the sites.

The dependency to these external tools is optional. ARGO can be used without having any of these connectors enabled, provided that 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

*List requirements that have been implemented in the release*

As already mentioned ARGO is not a single software, but a suite of software components that each is managed independently. There have been 12 releases of the ARGO components. During this period the following requirements have been covered:

**ARGO Compute Engine & Web API**

* Support for multiple monitoring engines running in active-active setup
* APIv2
* Stability and performance improvements

**ARGO Monitoring Engine**

* Completion of the Centralised Monitoring Engine
* Migration of middleware versions (midmon) and fedcloud services (cloudmon) monitoring to the Centralised Monitoring Engine
* Initial support for GOCDB as a single source of topology
* New probes (OneData)
* Fedcloud probes update
* Stability and performance improvements

**ARGO EGI Consumer and Connectors**

* Use of CE ingestion api for EGI Consumer
* Update connectors to use the VAPOR service instead of the decommissioned GSTAT
* Stability and performance improvements

**ARGO EGI Web UI**

* UI Enhancements
* Integration of SAML

**ARGO POEM**

* Initial support for probe management
* Initial steps for the connection to the EGI IdP/SP Proxy
* Stability and performance improvements
  + - **Changelog**
* **24/12/2016**
  + **Argo- Web UI [V1.3.4-1]** [**https://github.com/ARGOeu/argo-egi-web/releases/tag/v1.3.4-1**](https://github.com/ARGOeu/argo-egi-web/releases/tag/v1.3.4-1)
  + **Argo- Web UI [V1.3.3-1]** [**https://github.com/ARGOeu/argo-egi-web/releases/tag/v1.3.3-1**](https://github.com/ARGOeu/argo-egi-web/releases/tag/v1.3.3-1)
  + **Argo- Web UI [V1.3.2-1]** [**https://github.com/ARGOeu/argo-egi-web/releases/tag/v1.3.2-1**](https://github.com/ARGOeu/argo-egi-web/releases/tag/v1.3.2-1)
* **20/12/2016**
  + **Argo- Web UI [V1.3.1-1]** [**https://github.com/ARGOeu/argo-egi-web/releases/tag/v1.3.1-1**](https://github.com/ARGOeu/argo-egi-web/releases/tag/v1.3.1-1)
* **12/12/2016**
  + **Argo - Web API [v1.7.1-1]** [**https://github.com/ARGOeu/argo-web-api/releases/tag/v1.7.1-1**](https://github.com/ARGOeu/argo-web-api/releases/tag/v1.7.1-1)
  + **Argo- Connectors [V1.5.1-2]** [**https://github.com/ARGOeu/argo-egi-connectors/releases/tag/v1.5.1-2**](https://github.com/ARGOeu/argo-egi-connectors/releases/tag/v1.5.1-2)
  + **Poem [V1.0.3-1]** [**https://github.com/ARGOeu/poem/releases/tag/v1.0.3-1**](https://github.com/ARGOeu/poem/releases/tag/v1.0.3-1)
  + **Poem [V1.0.2-1]** [**https://github.com/ARGOeu/poem/releases/tag/v1.0.2-1**](https://github.com/ARGOeu/poem/releases/tag/v1.0.2-1)
  + **ARGO - Compute Engine [v1.6.9-1]** [**https://github.com/ARGOeu/argo-compute-engine/releases/tag/untagged-00740fb1f34cc1f6be6e**](https://github.com/ARGOeu/argo-compute-engine/releases/tag/untagged-00740fb1f34cc1f6be6e)
* **24/10/2016**
  + ARGO - Web API [v1.6.5.-2] <https://github.com/ARGOeu/argo-web-api/releases/tag/v1.6.5-2>
* **12/10/2016**
  + ARGO - Web API [v1.6.5.-1] <https://github.com/ARGOeu/argo-web-api/releases/tag/v1.6.5-1>
  + Argo - Web UI [v1.2.2-1] <https://github.com/ARGOeu/argo-egi-web/releases/tag/v1.2.2>
* **08/10/2016**
  + ARGO - Consumer [ingestion-enabled] <https://github.com/ARGOeu/argo-egi-consumer/commits/ingestion-enabled>
* **27/09/2016**
  + Argo Consumer [v1.4.5-1] <https://github.com/ARGOeu/argo-egi-consumer/releases/tag/v1.4.5-1>
* **26/09/2016**
  + **Poem [V1.0.1-1]** [**https://github.com/ARGOeu/poem/releases/tag/v1.0.1-1**](https://github.com/ARGOeu/poem/releases/tag/v1.0.1-1)
* **24/09/2016**
  + Argo Web API [v1.6.4-1] <https://github.com/ARGOeu/argo-web-api/releases/tag/v1.6.4-1>
  + Argo Web UI [v1.2.1-1] <https://github.com/ARGOeu/argo-egi-web/releases/tag/v1.2.1-1>
  + Argo Consumer [v1.4.4-1] <https://github.com/ARGOeu/argo-egi-consumer/releases/tag/v1.4.4-1>
  + **Argo- Connectors [v1.5.0-1]** [**https://github.com/ARGOeu/argo-egi-connectors/releases/tag/v1.5.0-1**](https://github.com/ARGOeu/argo-egi-connectors/releases/tag/v1.5.0-1)
  + **Poem [V1.0.0-1]** [**https://github.com/ARGOeu/poem/releases/tag/v1.0.0-1**](https://github.com/ARGOeu/poem/releases/tag/v1.0.0-1)
* **23/03/2016**
  + Argo Compute Engine [v1.6.7-1] <https://github.com/ARGOeu/argo-compute-engine/releases/tag/1.6.7-20160323160546.09642d4.build55>
* **03/02/2016**
  + Argo Web API [v1.6.3-1] <https://github.com/ARGOeu/argo-web-api/releases/tag/v1.6.3-1>
  + Argo Compute Engine [v1.6.6-1] <https://github.com/ARGOeu/argo-compute-engine/releases/tag/1.6.6-20160203183442.590f388.build48>
* **11/01/2016**
  + **Poem [V0.11.1-1]** [**https://github.com/ARGOeu/poem/releases/tag/v0.11.1-1**](https://github.com/ARGOeu/poem/releases/tag/v0.11.1-1)

## 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* | Monitoring updates |
| *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 expose a complete API allowing third parties to gather accounting data from the system.  Deployment of a central ARGO monitoring engine. NGI instances were decommissioned or kept for NGI’s internal purposes. In addition, monitoring instances for middleware versions (midmon) and fedcloud services (cloudmon) were decommissioned and all probes were integrated into central ARGO monitoring engine. A/R calculations are performed solely by using results from the central ARGO monitoring engine. Centralized ARGO monitoring engine poses a risk if only one instance is deployed. In case of failure of that instance whole grid will not be monitored. Therefore, a high availability setup is used.  With the introduction of a Centralized Monitoring Engine, accompanied with an HA active-active setup, Compute Engine needs to be able to accept metric data from two centralized sources at the same time. Thus there were two major design goals for the Compute Engine implemented. Compute A/R & status results by accepting data from multiple monitoring engines and exclude data for specific time periods for problematic monitoring engines. |
| *EXPLOITATION* | |
| *Target group(s)* | RIs, service providers, Users, NGIs, Resource centers |
| *Needs* | * Used for the Availability and Reliability monitoring * Provide a complete API allowing third parties to gather data from the system. * Used as a source of alerts for Operations Portal Dashboard * User 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, that 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 lead to improvements in the performance, robustness and reliability of the ARGO Monitoring Service. |
| *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 allow promote the service also to research communities and other infrastructures that can benefit for such type of service. |
| *URL to project result* | http://argo.egi.eu/  https://github.com/ARGOeu/ |
| *Success criteria* | The deployment of the results to the production implementation of the ARGO Monitoring Service. |
| *DISSEMINATION* | |
| *Key messages* | Promotion to key research infrastructures, e-infrastructures and scientific communities |
| *Channels* | Other than participation to EGI conferences, we do not plan any dissemination activities. This should be coordinated by EGI centrally. |
| *Actions for dissemination* | Dissemination actions will be performed by EGI |
| *Cost* | Not planned in EGI-Engage |
| *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
* Fedcloud probes update
* Use of the messaging API
* Stability and performance improvements

ARGO Web UI

* UI Enhancements
* Connect to the EGI IdP/SP Proxy

ARGO EGI Consumers and Connectors

* Decommission of Consumer and use ARGO nagios AMS-publisher instead
* Use of the messaging API for Connectors component
* Stability and performance improvements

ARGO POEM

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

# Messaging

## Introduction

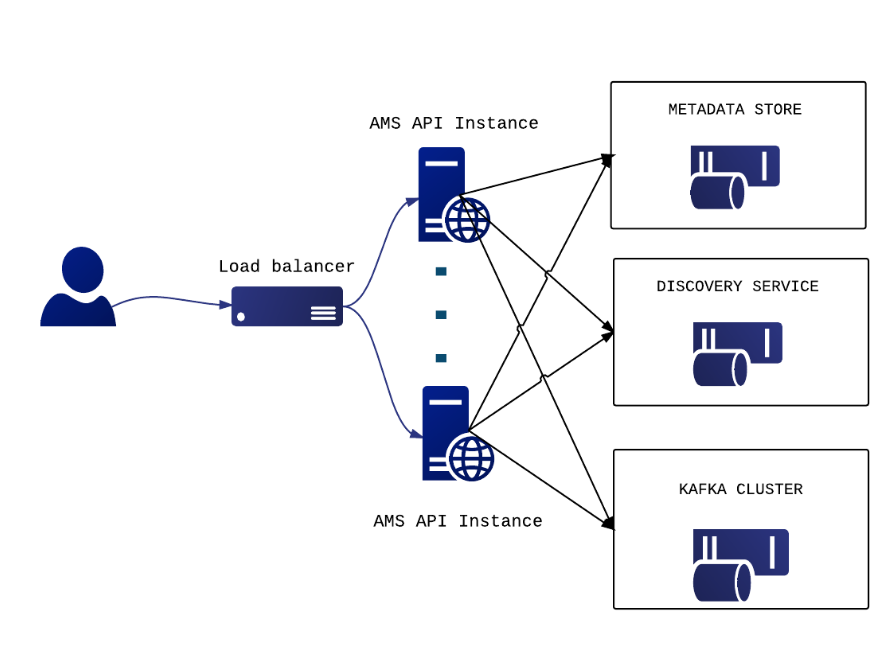
|  |  |
| --- | --- |
| **Tool name** | ARGO Messaging Service |
| **Tool url** | <http://argoeu.github.io> |
| **Tool wiki page** | Link to EGI wiki with description of the product  For JRA1 from <https://wiki.egi.eu/wiki/Message_brokers> |
| **Description** | The Messaging service enables reliable asynchronous messaging for the EGI infrastructure. |
| **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

*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

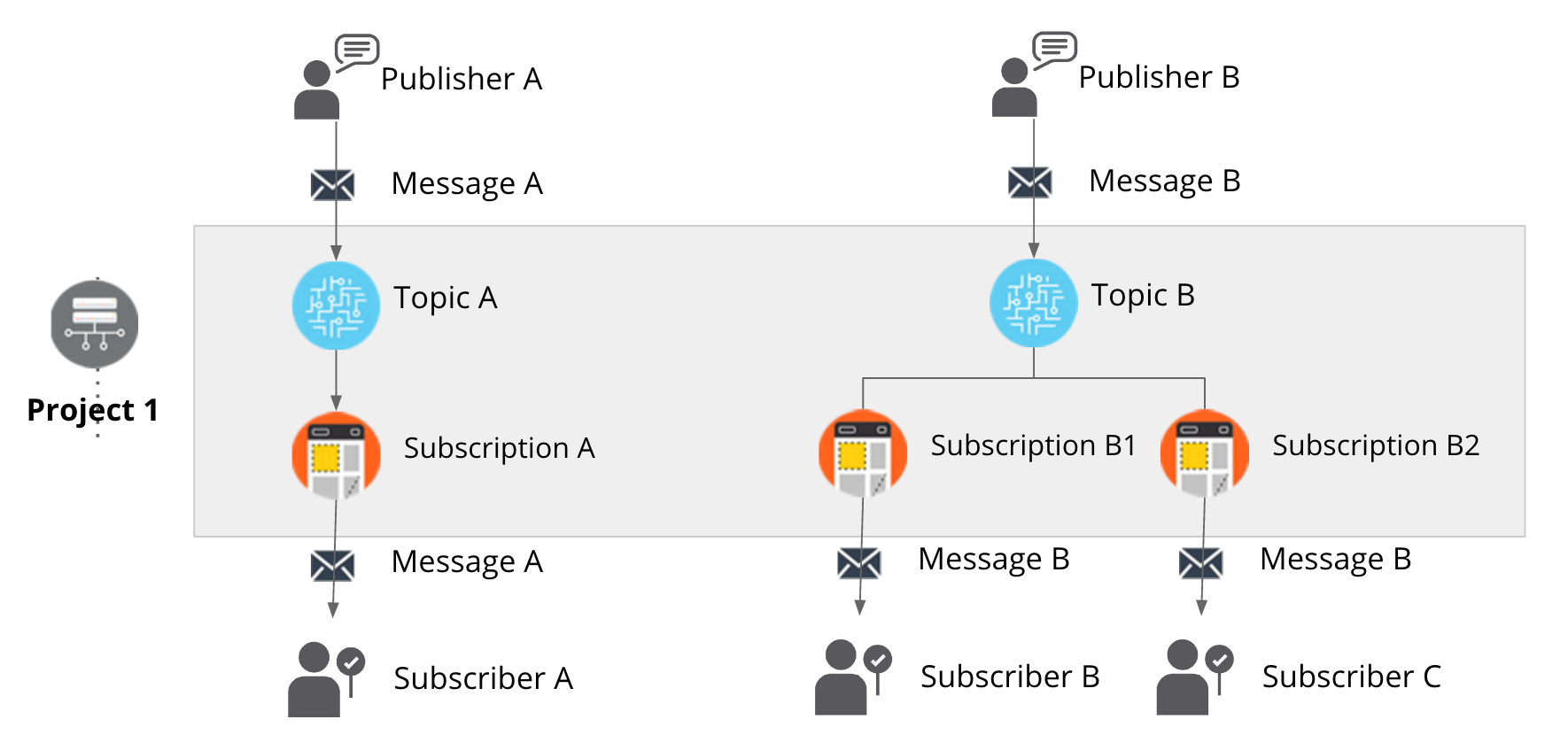
The Messaging service enables reliable asynchronous messaging for the EGI infrastructure. The current implementation of the Messaging service relies on a Message Broker Network of ActiveMQ services and uses the STOMP protocol for the publication and consumption of messages.



The new version of the Messaging service is going to replace the STOMP interface with an HTTP interface which will make the implementation of new clients easier and the implementation 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 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.



### Integration and dependencies

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

* ARGO Availability and Reliability Monitoring Service
* Accounting
* 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 at the moment.

## Release notes

### Requirements covered in the release

* APIv1 test implementation
* APIv1 final draft specification (ready for external party review)
* APIv1 final implementation
* APIv1 final specification
* API for data ingestion specification
* API for data ingestion implementation
* Stability and performance improvements

### Changelog

* **25/10/2016**
  + ARGO - Messaging Service [v1.0.0-1] <https://github.com/ARGOeu/argo-messaging/releases/tag/v1.0.0-1>

## Feedback on satisfaction

To be provided by EGI

## 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* | New Messaging Service - APIv1 final specification |
| DEFINITION | |
| *Category of result* | * Software & service innovation |
| *Description of the result* | In the new version of the Messaging service we have replaced the STOMP interface 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 centers, 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 developments during this period will allow the replacement of the Messaging Broker with this new Messaging Service.   * Messaging Service with an HTTP API for client access, for all * Transparent scalability & high availability * Access controls implemented at the API layer * Multi-tenant support * Improvements in the performance robustness of the Messaging Service. |
| *How will you protect the results?* | The ARGO Monitoring Framework is released under the Apache 2.0 license. |
| *Actions for exploitation* | * Create accounts per target group to publish messages to topics, or to consume messages as subscribers from a topic. * Provide the necessary documentation (all, for a publisher, or for a subscriber) * Let the target group test the Messaging Service. |
| *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 (This is depended on the available resources that the other product team have available in order to perform the migration within the EGI-Engage time frame). |
| DISSEMINATION | |
| *Key messages* | Promotion to key research infrastructures, e-infrastructures and scientific communities. |
| *Channels* | * Dissemination through the EGI conferences * Article featured in the EGI newsletter |
| *Actions for dissemination* | Results should be promoted by the EGI foundation |
| *Cost* | Not planned for EGI-Engage |
| *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

* APIv1 final specification
* Message Service Accounting: Metrics for Messaging Service
* Operational statistics
* Usage Statistics
* 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 new write API provides a script-accessible mechanism to manage custom properties. This allows clients to automate their property editing workflows which aims to reduce the admin overhead of manually managing custom properties.  The new cursor paging features allows all the data hosted in the DB to be paginated. This provides full API access to all historic data if needed.  Integration with the EGI CheckIn service provides Federated access to GOCDB for users who don’t own a client certificate or from browsers without personal certificates installed. |
| **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

*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.*

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 Check-In 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.

With the current release, a new dependency exists on the EGI Check In service to provide federated access to GOCDB for users without client certificates. In addition, a new write API has been introduced for managing custom properties on Sites/Services/Endpoints. This allows clients to automate their property editing workflows with the aim of reducing the admin overhead of manually managing custom properties. Other than this, there are no major alterations to the architecture.

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 Laboratory). The failover instance is synchronized hourly against the production data.

### 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.*

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

## Release notes

### Requirements covered in the release

*List requirements that have been implemented in the release*

* Prioritized Roadmap[[3]](#footnote-3)
* Full change log[[4]](#footnote-4) (includes smaller changes/bug fixes)

V5.7

* Addition of new monitoring attributes to service endpoints for ARGO monitoring. This has allowed ARGO to remove its dependency on the BDII, and GOCDB is now the only information system used for ARGO monitoring.
* A new write API for managing custom properties was implemented as requested by the WLCG Information Systems Task Force. This allows site admins to manage their own DN based ACLs per site required to post updates for a site/service. <https://rt.egi.eu/rt/Ticket/Display.html?id=11020>
* Cursor paging <https://rt.egi.eu/rt/Ticket/Display.html?id=10716>

Federated access <https://rt.egi.eu/rt/Ticket/Display.html?id=7493>

## Feedback on satisfaction

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

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 (<https://gocdb-test.esc.rl.ac.uk>). 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.

## 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* | *GOCDB* |
| *DEFINITION* | |
| *Category of result* | Software & service innovation |
| *Description of the result* | * Extension to authentication mechanism to allow federated access to the GOCDB portal. * Addition of a new write API has for managing custom properties on Sites/Services/Endpoints. * Addition of new monitoring attributes to service endpoints. * Cursor based paging of API results |
| *EXPLOITATION* | |
| *Target group(s)* | WLCG tool developers, ARGO service, Site/service admins and NGI managers |
| *Needs* | The Write API will allow clients to automate their property editing workflows; reducing the admin overhead of manually managing custom properties. The addition of new monitoring attributes to service endpoints allows ARGO to remove its dependency on the BDII; making the GOCDB the only information system used for ARGO monitoring. The addition of federated access to GOCDB makes the tool more attractive to users in communities which don’t tend to use personal certification. |
| *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. This has been carried out. The full source code is 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/5.7*](https://github.com/GOCDB/gocdb/releases/tag/5.7)  [*https://goc.egi.eu/*](https://goc.egi.eu/) |
| *Success criteria* | Regular use of the write API by at least one tool. New service endpoint attributes being used. |
| *DISSEMINATION* | |
| *Key messages* | * *The write API is now available* * *GOCDB can be accessed using federated credentials* * *The required changes are in place for ARGO to switch to using GOCDB for information being provided by the BDII* |
| *Channels* | *WP3 meetings, EGI OMB meetings, WLCG Information Systems Evolution Task Force* |
| *Actions for dissemination* | WLCG Info. Sys. Evolution TF Dec. - <https://indico.cern.ch/event/575249/> EGI OMB November meeting - <https://indico.egi.eu/indico/event/2814/>  GridPP37 - <https://indico.cern.ch/event/556609/timetable/>  Announcement emails to multiple EGI mailing lists and WLCG information system evolution mailing list. |
| *Cost* |  |
| *Evaluation* | Uptake of use of new features |

## Future plans

* Hold a GOCDB OTAG (operation tools advisory group) meeting in order to review and re-prioritise the GOCDB development roadmap up until the end of the project.
* Review/clarify automatic freshness of data check (<https://rt.egi.eu/rt/Ticket/Display.html?id=8240>)
* Introduce configuration management to the GOCDB production server, enabling a quicker recovery back to a full service in the event of server failure and ensuring consistency in configuration

# 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 |
| **Source code** | https://github.com/CESNET/secant |

## 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.*

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

*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.*

There are two foreseen scenarios how Secant can be deployed, it can either work on the level of a cloud site to assess images used by the particular provider, or it can act as a tool supporting security assessment and endorsement on the level of EGI. In any cases Secant has to be integrated with a cloud management framework. The current implementation uses OpenNebula commands to manage virtual machines and their images.

## Release notes

### Requirements covered in the release

*List requirements that have been implemented in the release*

The release focuses on addressing bugs and issues detected in a pilot deployment and testing.

## Feedback on satisfaction

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

Secant is being tested at CESNET and its MetaCloud site. A movement to a more extensive testing phase was blocked by changes of the mechanism to distribute images to EGI clouds, which allowed the developers to only perform an evaluation in a closed environment. A few dozens of virtual appliances underwent testing done by Secant.

## 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.***

The goal of Secant is to improve the security of cloud images and provide a tool to facilitate the verification process, providing a procedure for the security endorsement.

|  |  |
| --- | --- |
| *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

We will continue to explore ways how the Secant tool can be integrated with the AppDB to support the endorsement process.

# Accounting Repository

## Introduction

The EGI Accounting Repository runs using software from the APEL project.

APEL is an accounting tool that collects resource usage data from sites participating in the EGI and WLCG infrastructures as well as from sites belonging to other Grid organisations that are collaborating with EGI, including OSG and NorduGrid.

The accounting information is gathered from different sensors into a central accounting repository where it is processed to generate statistical summaries that are available through the EGI/WLCG Accounting Portal. Statistics are available for view in different detail by Users, VO Managers, Site Administrators and anonymous users according to well-defined access rights.

Table 1 provides a summary of the tool covered in this release.

Table 1 – APEL tool summary

|  |  |
| --- | --- |
| **Tool name** | APEL |
| **Tool URL** | <http://apel.github.io/> |
| **Tool wiki page** | <https://wiki.egi.eu/wiki/Accounting_Repository> |
| **Description** | EGI Core Service – The Accounting Repository collects and stores user accounting records from various services offered by EGI. |
| **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** | EGI |
| **User of the service** | Site admins |
| **User Documentation** | <https://twiki.cern.ch/twiki/bin/view/EMI/EMI3APELClient> |
| **Technical Documentation** | <https://twiki.cern.ch/twiki/bin/view/EMI/EMI3APELClient> |
| **Product team** | STFC |
| **License** | Apache License, Version 2.0 |
| **Source code** | <https://github.com/apel/apel> |

This section provides a short introduction to the components provided by the APEL project as part of the EGI Accounting Repository. Then the high-level architecture of the tool and its components are described, along with the integrations and dependencies it has. Release notes and the results of testing for this release are then provided. Finally, plans for exploitation, dissemination, and future developments are shown.

## 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.*

Figure 1 shows how the APEL client, central APEL server and EGI Accounting Portals interact.



Figure 3 - APEL components and their interactions. Components in red are provided by the APEL project.

1. APEL clients can run an APEL parser to extract data from a batch system and place it in their client database, or they can use third-party tools to extract batch or cloud data. This data is then unloaded into a message format suitable for transmission.
2. APEL clients run a sending Secure Stomp Messenger[[5]](#footnote-5) (SSM) to send these messages containing records via the EGI Message Brokers the central APEL server. The messages can contain either Job Records or Summary records. This is configurable in the APEL client.
3. The central APEL server runs an instance of the SSM, which receives these messages and a “loader” processes the records in the messages and loads them into a MySQL database.
4. A “summariser” process runs to create summaries of any Job Records received and load them in a “SuperSummaries” table along with any Summary records. This summariser runs as a cron job approximately once a day.
5. A database “unloader” process unloads the summary records into the message format to be sent on by the sending SSM via the EGI Message Brokers to the EGI Accounting Portal.

There are no changes to the service architecture in 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.*

All communication between clients and servers is via the EGI Message Broker network using the APEL SSM package. The SSM software can be configured to send or receive messages. Where the messages are destined for is controlled by the queue, which is set in the SSM configuration.

The central APEL server uses the EGI service registry (GOCDB) to get a list of APEL endpoints so that only data from endpoints correctly defined in GOCDB are processed.

SSM can be configured to get a list of message brokers from the EGI information system (querying a BDII) or it can be pointed directly at a message broker.

There are no changes to the dependencies in this release.

## Release notes

These are the changes included in this release of the APEL software, version 1.6.0, since the previous Accounting Repository Release in EGI-Engage.

* Added support for v0.4 of the cloud accounting schema
* Added support for GOCDB read API results paging
* Added support for mixed time duration formats found in Torque 5.1.3
* Added support for the new format of CPU counts found in Torque 5.1.0
* Disabled non-performant duplicate sites check in summariser
* Added scripts that support building packages for SL7 which are compatible with CentOS 7
* Fixed handling of group attributes in storage records
* Added setup script for installation on alternative operating systems
* Added more unit tests
* Minor bug fixes and tweaks

### Requirements covered in the release

*List requirements that have been implemented in the release*

WP3 task 3.1.1 – Cloud Accounting: Usage Record V0.4

## Feedback on satisfaction

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

The APEL project uses a development process based around GitHub, which includes a semi-automatic testing procedure used to assess the quality of software releases.

For details of the testing procedure used, see the APEL Development Process document[[6]](#footnote-6). Table 2 summarises the results of testing this release.

Table 2 - APEL 1.5.1 testing results

|  |  |  |
| --- | --- | --- |
|  | **Result** | **Link** |
| **Unit tests** | All unit tests passed | <https://travis-ci.org/apel/apel/builds/194861155> |
| **Coverage** | Coverage metric decreased by 6.7% due to previously ignored files being included – actual coverage increased slightly | <https://coveralls.io/builds/9818974> |

## 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 | Second EGI Accounting Repository release |
| DEFINITION | |
| Category of result | Software & service innovation |
| Description of the result | Update to the software that provides the EGI Accounting Repository including a number of small fixes and improvements as well as support for a new cloud accounting usage record schema. |
| EXPLOITATION | |
| Target group(s) | RIs, international research collaborations, service providers, Funding agencies and decision/policy makers |
| Needs | Usage accounting data that can aid in ensuring resources are used as expected. |
| How the target groups will use the result? | Service providers update client installations. Extra metrics collected in the repository will be presented in the Portal for use. |
| Benefits | Support for different version of batch system and packages now available for EL7 based systems. |
| How will you protect the results? | Open source license (Apache License, Version 2.0) |
| Actions for exploitation | Roll out update to production server infrastructure and package the software for use at the client end. Work with Accounting Portal to update views. |
| URL to project result | <https://github.com/apel/apel/releases/latest> |
| Success criteria | Smooth roll out and any issues resolved quickly |
| DISSEMINATION | |
| Key messages | New version of the accounting software available that support extra metrics for cloud accounting |
| Channels | EGI OMB, WP3 meetings |
| Actions for dissemination | Announce at an OMB and WP3 meeting |
| Cost | N/K |
| Evaluation | Installation of new release and feedback on new features |

## Future plans

The EGI Accounting Repository will be developed further under EGI Engage culminating in the final release of the Accounting Repository under EGI Engage.

# Accounting Portal

## Introduction

|  |  |
| --- | --- |
| **Tool name** | Accounting Portal |
| **Tool url** | https://accounting.egi.eu |
| **Tool wiki page** | https://wiki.egi.eu/wiki/Accounting\_Portal |
| **Description** | The Accounting Portal provides data accounting views for users, VO Managers, NGI operations and the general public. |
| **Value proposition** | The new UI enhancements allow users to see the accounting in new ways. |
| **Customer of the tool** | Infrastructure users, VO Managers, Operations Centres, Sites and the general public. |
| **User of the service** | Infrastructure users, VO Managers, Operations Centres, Sites and the general public. |
| **User Documentation** | https://documents.egi.eu/public/ShowDocument?docid=2789 |
| **Technical Documentation** | https://documents.egi.eu/public/ShowDocument?docid=2545 |
| **Product team** | CESGA, CSIC |
| **License** | Apache |
| **Source code** | https://github.com/cesga-egi/accounting |

## 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.*

The Accounting Portal is a web application based on Apache, and MySQL, which has as its primary function to provide users with customized accounting reports, containing tables and graphs, as web pages. It also offers RESTful web services to allow external entities to gather accounting data.

The basic architecture of the Portal consists on:

1. A backend, which aggregates both data and metadata in a MySQL database, using the APEL SSM messaging system[[7]](#footnote-7) to interact with the Accounting Repository and several scripts, which periodically gather the data, and metadata described below.
2. A Model represented by database schemas both external and internal which define database tables for several types of accounting (grid, cloud, storage, multicore, user statistics etc.) and metadata (topology, geographical data, site status, nodes, VO users and admins, site admins etc.) and a series of parametrized queries,
3. A set of views that expose the data to the user. These views contain a form to set the parameters and metric of the report, a number of tables showing the data parametrized by two selectable dimensions and filtered by several parameters, a line graph showing the table data, and pie charts showing the percentage distribution on each dimension. It is planned that this part of the portal will evolve with interactive graphs, responsive in real time, reactive and only exposing advanced controls on user demand.
4. A graphical representation of these components is depicted on Fig. 1.

Figure 4. Accounting portal architecture

### 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.*

There are dependencies on other tools and components that provide metadata that is used in the portal, this metadata includes:

1. **Geographical Metadata**: Country and NGI affiliation of sites. Generally, this follows current borders, but there are important exceptions. This is gathered from GOCDB using its XML-based API.
2. **Topological Metadata**: Sites are presented in trees, there are Country and NGI trees that correspond to geographical classifications, but there are also trees based on topological classifications like Tier1 and Tier2 sites, OSG sites and uncategorised sites. Inside Tier2 sites, the federation they belong to is also important and can trigger special code in some cases. Gathered from several sources, including OSG and WLCG databases.
3. **Role Metadata**: VO members and managers, and the site admins records. This metadata controls the access to restricted views. Information is gathered from GOCDB and individual VOMS servers constructing a list of individual VOMSes and querying them with the VOMS API.
4. **Country affiliation data**: Each user record contains a user identifier that has his/her user name, institution and sometimes country. Scripts in the backend map each user with a country based on the institution which issues their certificate. This data is used in anonymized statistics per country on: how much resources from other countries are used by given country and the distribution of its resources used by other countries.
5. **VO Data**: To make possible VO selection in the user interface, the portal stores lists of VOs. They are also used to filter incorrect VO names, provide access to VO managers, and arrange accounting by VO discipline (such as “High Energy Physics”, “Biomedicine”, “Earth Sciences”, etc.). Information is gathered from the Operations Portal using its XML based APIs.
6. **Site status metadata**: Sites must be filtered to exclude those that are not in production (due to being closed or being in test mode). There must be also metadata to aggregate the accounting history of sites whose name has been changed. There are requirements to extend this functionality to NGIs. Information is gathered from GOCDB using its XML tables and internal tables compiled as part of EGI PROC 15[[8]](#footnote-8).
7. **Pledge metadata**: The WLCG reports have to contain only those sites where MoUs or other pledges between VOs and sites are honoured, so the validity date and pledged hours are needed. Information is gathered from WLCG using the REBUS service.
8. ***Other metadata:*** There are also other metadata like local privileges, SpecInt calculations, publication status, VO activities and more. Some of these metadata is calculated internally using other types of metadata and published for other EGI operational tools, like VO activity data and Site UserDN publishing

## Release notes

### Requirements covered in the release

*List requirements that have been implemented in the release*

* Changed landing page according to new requirements.
* New WLCG sub-portal with dedicated navigation and menu bar
* Changed WLCG reports, integrated REBUS Tier1 report.
* Added contextual help
* Added context-sensitive info on each page.
* Renamed Sites to Resource Centres, NGIs to Operations Centres.
* New EGI Resource Centre Report with per-country, top10 and top100 reports for both HTC and Cloud.
* Added Year, Half-year and Quarter Granularity.
* Improved discipline view.
* Reorganization of the menus.
* Subdiscipline views switch to VO-based view when no subdisciplines are present.
* Changed EGI country view to only include EGI federation countries, re-implemented view with all countries for WLCG.
* Modified CSV support back to the server so it can be linked.
* Changed landing page graph to commas even on dot-based locales.
* Added extra padding on tables.
* Reimplemented VO metrics support.
* Added EGI branding on every page.

## 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* | Accounting Portal |
| *DEFINITION* | |
| *Category of result* | Software & service innovation |
| *Description of the result* | Improved the UI, added a WLCG specific subportal, improved landing page, added contextual help, added new EGI reports, added 3 and 6 months and yearly time intervals, improved discipline view, reorganized menus, restricted country view, improved CSV support, reimplemented VO metrics. Added EGI branding. |
| *EXPLOITATION* | |
| *Target group(s)* | Infrastructure users, VO Managers, Operations Centres, Sites and the general public. |
| *Needs* | UI Improvements, new ways to access data, new reports. |
| *How the target groups will use the result?* | Reporting activities, problem solving, MoU estimation. |
| *Benefits* | Better reports, better problem solving, better MoU estimation. |
| *How will you protect the results?* | Attribution via open source license |
| *Actions for exploitation* | The result is a public web page, immediately exploitable. |
| *URL to project result* | accounting-next.egi.eu |
| *Success criteria* | Continued use. |
| *DISSEMINATION* | |
| *Key messages* | The UI is improved and new features are available. |
| *Channels* | EGI PROC 23, mailing lists. |
| *Actions for dissemination* | Send mails to mailing lists. |
| *Cost* | Negligible. |
| *Evaluation* | Increased petitions to the web server. |

## Future plans

Continue new releases according to the Roadmap.

1. Related work

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://wiki.egi.eu/wiki/EGI-Engage:TASK_JRA1.4_Operations_Tools#GOCDB> [↑](#footnote-ref-3)
4. <https://github.com/GOCDB/gocdb/blob/dev/changeLog.txt> [↑](#footnote-ref-4)
5. <https://github.com/apel/ssm> [↑](#footnote-ref-5)
6. <https://documents.egi.eu/document/2739> [↑](#footnote-ref-6)
7. <https://wiki.egi.eu/wiki/APEL/SSM> [↑](#footnote-ref-7)
8. <https://wiki.egi.eu/wiki/PROC15_Resource_Center_renaming> [↑](#footnote-ref-8)