EGI-Engage

Second release of the accounting and operational tools

D3.10

Date	09 March 2017
Activity	WP3
Lead Partner	CSIC, CNRS, STFC
Document Status	FINAL
Document Link	https://documents.egi.eu/document/3018

Abstract

This deliverable describes the second release of the EGI Accounting and Operational Tools during EGI-Engage project, including the developments made during the second year of the project for the Operations Portal, ARGO, Messaging, GOCDB, Security Monitoring, Accounting Repository and Portal. The evolution of these tools has been driven by the need to support new technologies (e.g. cloud) and to satisfy new requirements emerging from service providers and user communities, in particular from the Research Infrastructures contributing to EGI-Engage via the EGI Competence Centres (CCs) and the Resource Providers (RPs) who contribute infrastructure services to the federation. The development roadmap has been reviewed and updated according to a requirement gathering process, which has been accomplished in collaboration with the other EGI Engage WPs in charge of the communication with users and key stakeholders.

The EGI-Engage project is co-funded by the European Union (EU) Horizon 2020 program under Grant number 654142 http://go.egi.eu/eng

COPYRIGHT NOTICE

DELIVERY SLIP

	Name	***Partner/Activity***	***Date***
From:	Cyril Lorphelin Christos Kanellopoulos David Meredith Daniel Kouril Adrian Coveney Ivan Diaz Alvarez Diego Scardaci	CNRS/WP3 GRNET/WP3 STFC/WP3 CESNET/WP3 STFC/WP3 CSIC/WP3 EGI F.-INFN/WP3	13/02/2017
Moderated by:	Małgorzata Krakowian	EGI Foundation/WP1	
Reviewed by	Yannick Legre Alessandro Paolini	EGI Foundation/WP1 EGI Foundation/WP5	02/03/2017 17/02/2017
Approved by:	AMB and PMB		6/03/2017

DOCUMENT LOG

Issue	***Date***	***Comment***	***Author/Partner***
v.1	13/02/2017	Full draft ready for internal review	Cyril Lorphelin/CNRS Christos Kanellopoulos/GRNET David Meredith/STFC Daniel Kouril/CESNET Adrian Coveney/STFC Ivan Diaz Alvarez/CSIC Diego Scardaci/EGI F. - INFN
v.2	15/02/2017	Full draft ready for external review	Diego Scardaci/EGI F. - INFN
FINAL	6/03/2017	Final version	

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

Executive summary

This deliverable describes the second release of the EGI Accounting and Operational Tools during EGI-Engage project, including the developments made during the second year of the project for the Operations Portal, ARGO, Messaging, GOCDB, Security Monitoring, Accounting Repository and Portal. The evolution of these tools has been driven by the need to support new technologies (e.g. cloud) and to satisfy new requirements emerging from service providers and user communities, in particular from the Research Infrastructures contributing to EGI-Engage via the EGI Competence Centres (CCs) and the Resource Providers (RPs) who contribute infrastructure services to the federation. The development roadmap has been reviewed and updated according to a requirement gathering process, which has been accomplished in collaboration with the other EGI Engage WPs in charge of the communication with users and key stakeholders.

The Operations Portal team upgraded the different technologies adopted by the portal to ensure a better maintainability and an enhancement of the performances. A continuous integration process has been established to improve the release quality and reduce the time to production. Further information has been added to the VO ID card and a complete replacement of GSTAT is now available through the VAPOR module.

The deployment of a central ARGO monitoring engine, able to serve a large infrastructure with a high availability setup, is now possible. Such deployment requires less maintenance effort and enables faster and streamlined deployment of new tests or updating of existing ones. This leads to improvements in the performance, robustness and reliability of the ARGO Monitoring Service.

A new version of the Messaging Service has been released. It provides an HTTP API that enables users/systems to implement a message-oriented service using the Publish/Subscribe Model over plain HTTP. This new interface makes the implementation of new clients easier and the implementation more robust. The ARGO monitoring system, the Operations Portal and the accounting system will migrate to the new Messaging Service by the end of the project.

During the second year, the GOCDB team focused its effort on developing a new write API that 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. Furthermore, the GOCDB has been integrated with the new EGI CheckIn service to manage users' authentication and authorisation.

Finally, the accounting team deployed in production a new cloud usage record that collect additional attributes about the VM instantiated in the EGI Federated Cloud and the new portal, which replaced the old one, with a completed revised look & feel, a contextualised online help and several new features available.

1 Operations Portal

1.1 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

1.2 Service architecture

1.2.1 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] and RTIR[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.

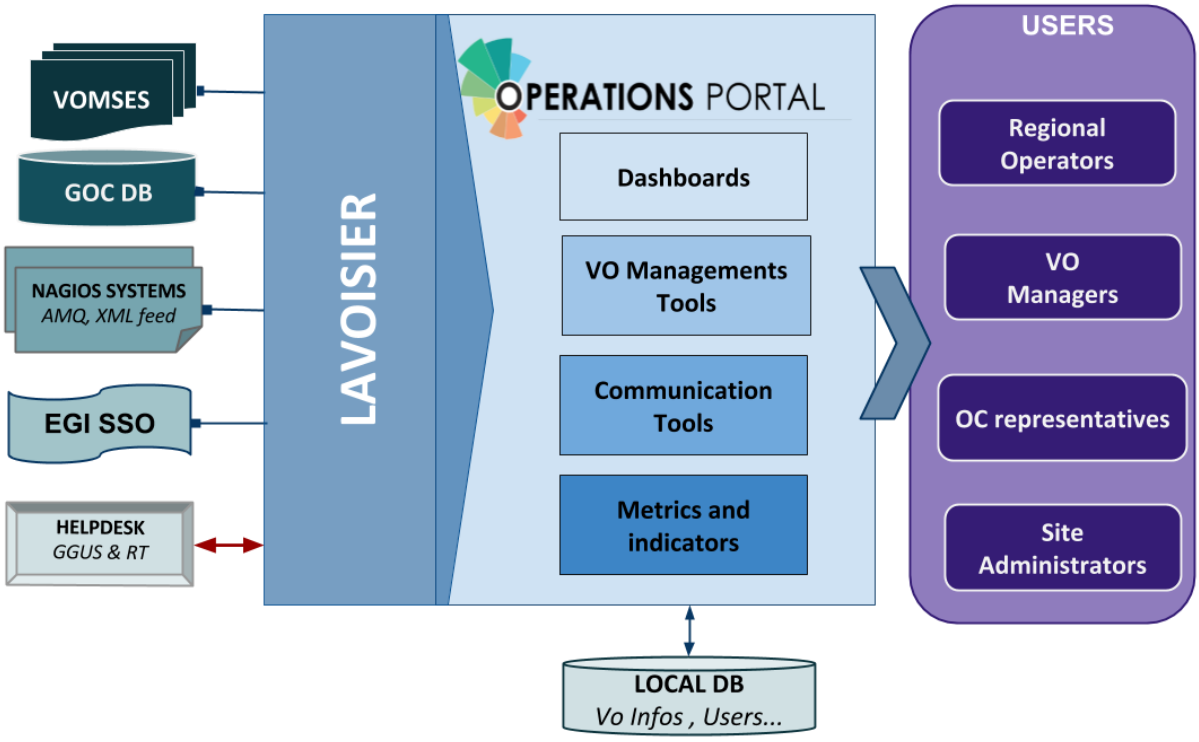


Figure 1. Operations Portal architecture

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

[1] www.ggus.eu
[2] https://wiki.egi.eu/wiki/EGI_CSIRT:Main_Page

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.

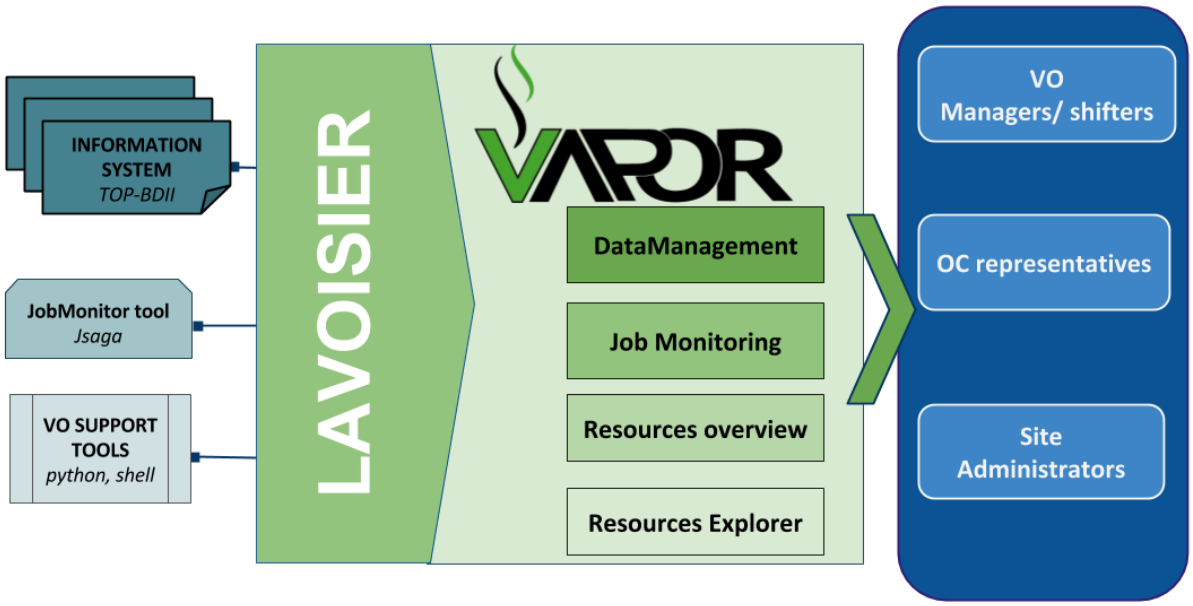


Figure 2. VAPOR architecture

1.2.2 Integration and dependencies

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

1.3 Release notes

1.3.1 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:

a) 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).

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 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).

d) 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).

e) 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

1.3.2 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.

1.3.3 VAPOR 2.0

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

[3] https://documents.egi.eu/document/2660

1.3.4 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.

1.3.5 VAPOR 2.2

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

1.4 Feedback on satisfaction

Prioritization and testing has been done by dedicated Operations Portal Advisory and Testing Board (OPAnTG)[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.

[4] https://wiki.egi.eu/wiki/OTAG#Operations_Portal_Advisory_and_Testing_Board

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.

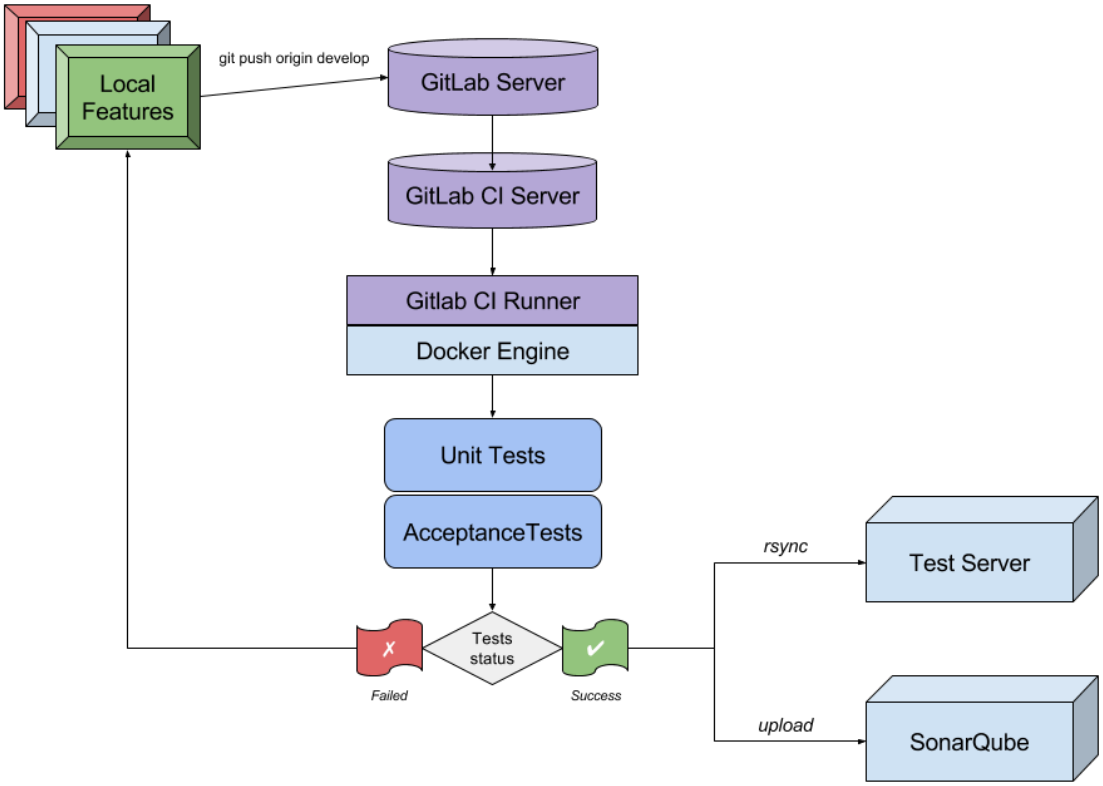


Figure 3. Operations Portal - Automatic test suite.

1.5 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

1.6 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, the following activities will be completed by the end of the project:

- VAPOR
  - V2.2 release in production;
  - Implementation of improvements asked by users.
- Operations Portal

- Integration of complementary metrics for the VO: accounting data and AppDB changes;
- Improvements on the Vo ID Card;
- Support of the new EGI AAI based on the CheckIn service (IdP/SP Proxy).

2 ARGO

2.1 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/

2.2 Service architecture

2.2.1 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.

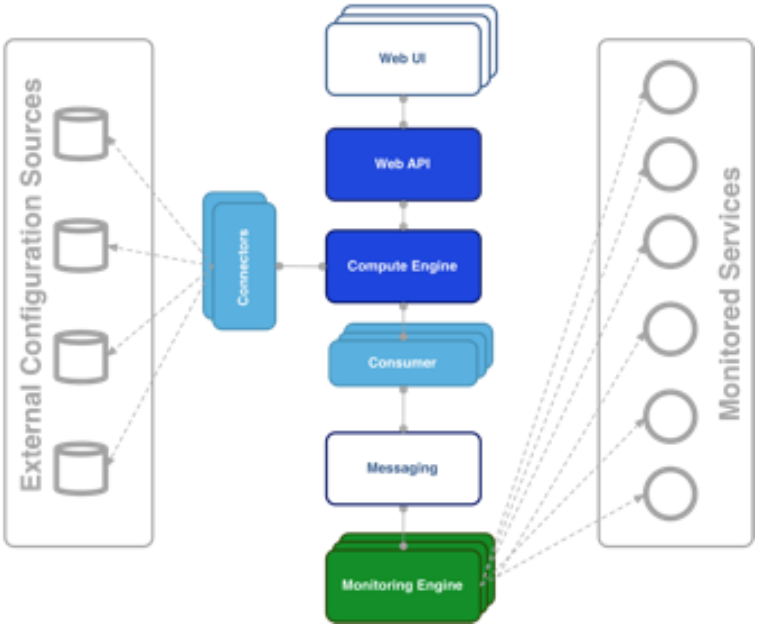


Figure 4. Argo architecture

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

2.2.1.1 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 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 EGI 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.

2.2.1.2 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.

2.2.1.3 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] before passing to the Compute Engine.

2.2.1.4 The ARGO Compute Engine

A powerful and scalable analytics engine built on top of Hadoop and HDFS[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.

2.2.1.5 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.

2.2.1.6 The ARGO Web UI

The default web UI is based on the Lavoisier Data Aggregation Framework[7].

2.2.2 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] 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.

[5] https://avro.apache.org/docs/1.2.0
[6] http://hadoop.apache.org/
[7] http://software.in2p3.fr/lavoisier/
[8] http://w3.hepix.org/benchmarks/doku.php

2.3 Release notes

2.3.1 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 second year of the project, there have been 12 releases of the ARGO components that covered the following requirements:

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 EGI Fedcloud services (cloudmon) monitoring to the Centralised Monitoring Engine;
- Initial support for GOCDB as a single source of topology;
- New probes (OneData);
- EGI 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

2.3.1.1 Changelog

- **24/12/2016**
  - **ARGO Web UI [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**
  - **ARGO Web UI [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**
- **12/12/2016**
  - **ARGO Web API [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**
  - **Poem [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**
  - **ARGO Compute Engine [v1.6.9-1]**
    - **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

- **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
    - **Poem [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

2.4 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.

2.5 Plan for Exploitation and Dissemination

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. Deployment of a central ARGO monitoring engine able to serve a large infrastructure with a high availability setup. 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 periods for problematic monitoring engines.
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.
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.

2.6 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
- EGI Fedcloud probes update
- Use of the messaging API
- Stability and performance improvements

ARGO Web UI

- UI Enhancements
- Connection 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

3 Messaging service

3.1 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/

3.2 Service architecture

3.2.1 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.

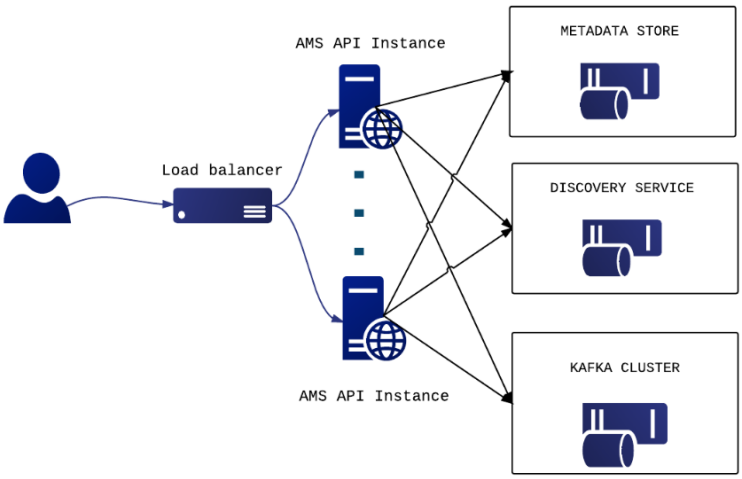


Figure 5. Messaging service architecture

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.

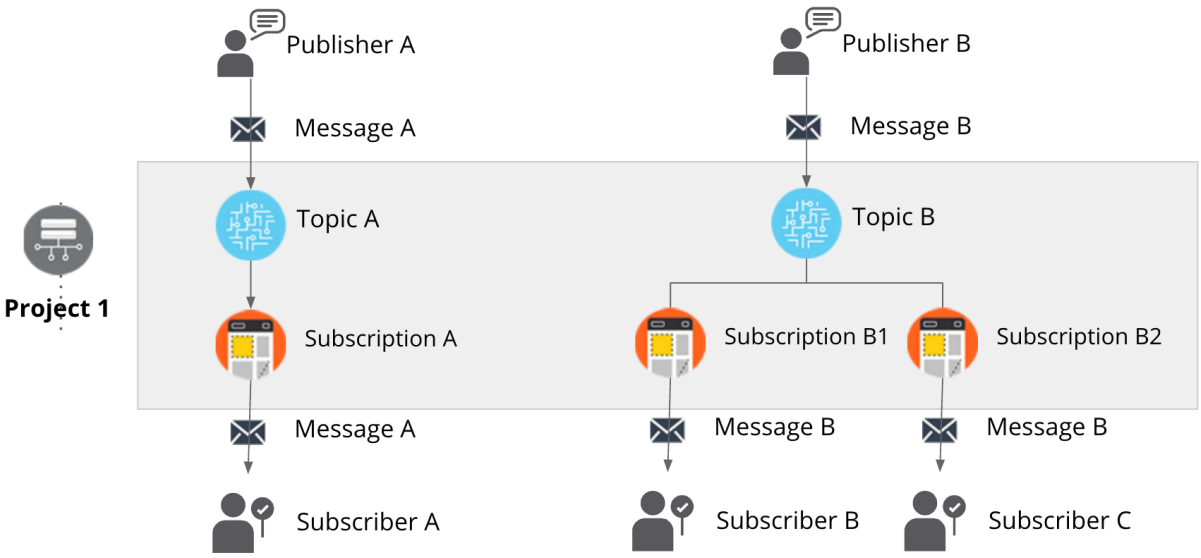


Figure 6. The new ARGO messaging service

3.2.2 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 at the moment.

3.3 Release notes

3.3.1 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

3.3.2 Changelog

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

3.4 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.

3.5 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; • 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.

3.6 Future plans

- APIv1 final specification
- Message Service Accounting: metrics for Messaging Service
- Operational statistics
- Usage Statistics
- Stability and performance improvements

4 GOCDB

4.1 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 do not 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

4.2 Service architecture

4.2.1 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.

With the current release, a new dependency exists on the EGI CheckIn 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.

4.2.2 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 over tool.

4.3 Release notes

4.3.1 Requirements covered in the release

- Prioritized Roadmap[9]
- Full change log[10] (includes smaller changes/bug fixes)

[9] https://wiki.egi.eu/wiki/EGI-Engage:TASK_JRA1.4_Operations_Tools#GOCDB
[10] https://github.com/GOCDB/gocdb/blob/dev/changeLog.txt

4.3.1.1 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

4.4 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 (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. Recent GOCDB releases successfully passed this stage.

The GOCDB development process is described in Appendix II.

4.5 Plan for Exploitation and Dissemination

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 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, Resource/service provider 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 do not 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://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.

4.6 Future plans

Below, a list of the main planned activities classified accordingly to their priorities:

- High priority
  - Write API Extensions (11020)
  - Verify data freshness check (8240)
- Intermediate priority
  - NGI Certification Status Rules (9084): Useful to have a quick way of changing CertificationStatus of all the resource centres belonging to an NGI.
- Lower priority
  - Downtime classification changes (10845)
  - Add unique constraint on HostName + ServiceType pair (10368)

In future projects, GOCDB development activity will focus on replacing the UI with a modern Web framework, extending GOCDB in the info-service space supporting dynamic attributes and improve the change logging.

5 Security Monitoring

5.1 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

5.2 Service architecture

5.2.1 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.

5.2.2 Integration and dependencies

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 the EGI infrastructure management. In any case, Secant has to be integrated with a cloud management framework. The current implementation uses OpenNebula commands to manage virtual machines and their images.

5.3 Release notes

5.3.1 Requirements covered in the release

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

5.4 Feedback on satisfaction

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.

5.5 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	The tools will facilitate the endorsement process and will help the endorsers

groups will use the result?	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	Possibilities will be examined how to integrate Secant with the AppDB to support endorsement process.
Cost	
Evaluation	Utilization of Secant in endorsement process.

5.6 Future plans

In next months, the product team will focus on analysing how the Secant tool could assist the endorsement process and on the definition of the related interfaces with the AppDB.

6 Accounting Repository

6.1 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, resource provider 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	Improved information about the usage of the cloud resources within the EGI infrastructure.
Customer of the tool	e-Infrastructures, research infrastructures and, in general, distributed infrastructures.
User of the service	Resource providers, NGI admins, EGI operations, end users.
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.

6.2 Service architecture

6.2.1 High-Level Service architecture

Figure 7 shows how the APEL client, central APEL server and EGI Accounting Portals interact.

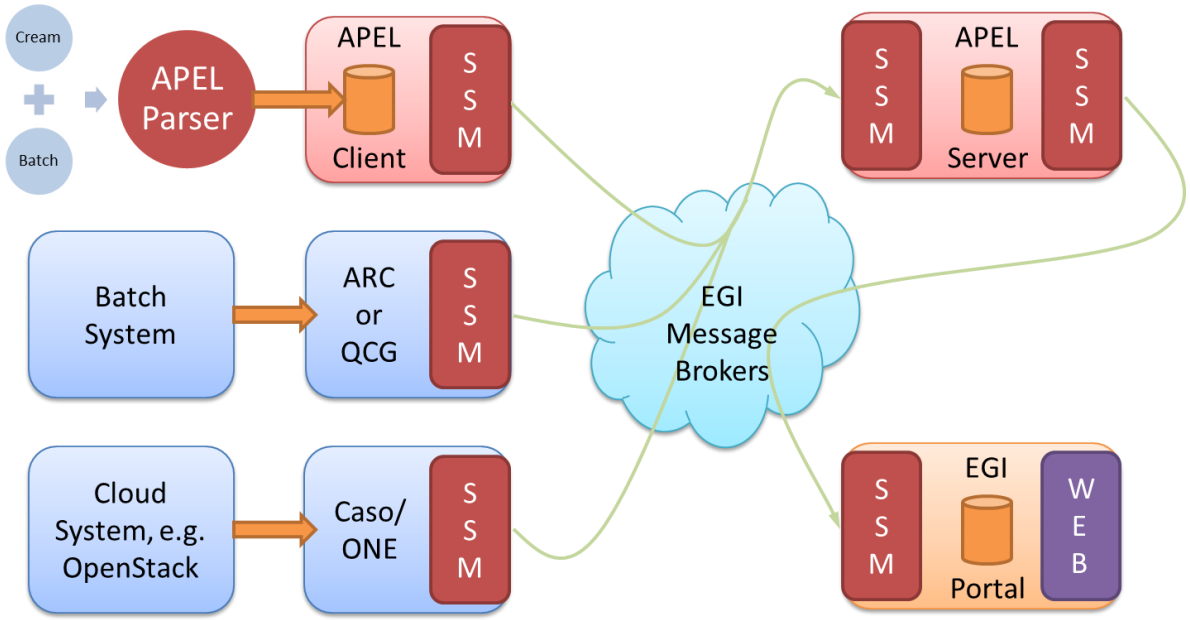


Figure 7. APEL components and their interactions. Components in red are provided by the APEL project.

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[11] (SSM) to send these messages containing records via the EGI Message Brokers to 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.

[11] https://github.com/apel/ssm

6.2.2 Integration and dependencies

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.

6.3 Release notes

6.3.1 Requirements covered in the release

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.

6.4 Feedback on satisfaction

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[12]. Table 2 summarises the results of testing this release.

[12] https://documents.egi.eu/document/2739

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

6.5 Plan for Exploitation and Dissemination

Name of the result	Accounting Repository
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 various 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	
Evaluation	Installation of new release and feedback on new features

6.6 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. This will include new batch parsers, support for additional storage systems, support for long-running virtual machines, provision of a method to extract APEL format records from non-APEL SQL databases, development of production requirements for data set usage accounting, an initial implementation of GPGPU usage accounting, and documenting of the support in the Repository for new AAI systems.

7 Accounting Portal

7.1 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	Improved look & feel. New views that allow to aggregate data in different ways. Improved support for scientific disciplines.
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

7.2 Service architecture

7.2.1 High-Level Service architecture

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.

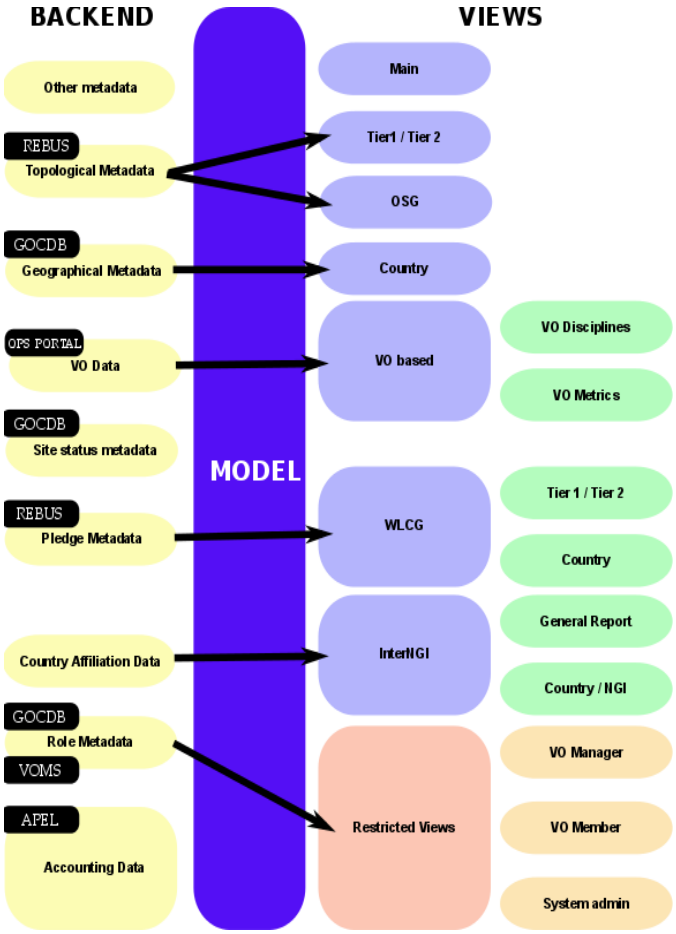
The basic architecture of the Portal consists on:

1. A backend, which aggregates both data and metadata in a MySQL database, using the APEL SSM messaging system[13] to interact with the Accounting Repository and several scripts, which periodically gather the data and metadata described below.

13 https://wiki.egi.eu/wiki/APEL/SSM

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 parameterised 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 parameterised 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.



A graphical representation of these components is depicted on Figure 8.

Figure 8. Accounting portal architecture

7.2.2 Integration and dependencies

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

1. **Geographical Metadata**: Resource providers' country and NGI affiliation. Generally, this follows current borders, but there are important exceptions. This is gathered from GOCDB using its XML-based API.
2. **Topological Metadata**: Resource providers 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 anonymised 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[14].
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.

[14] https://wiki.egi.eu/wiki/PROC15_Resource_Center_renaming

7.3 Release notes

7.3.1 Requirements covered in the release

- New home page.
- New WLCG sub-portal with dedicated navigation and menu bar.
- Changed WLCG reports, integrated REBUS Tier1 report.
- Added contextual help.
- Added descriptive information on each page explaining the meaning of the input variables and of the several metrics showed.
- Terminology used in the portal completely revised.
- New EGI Resource Centre Report with per-country, top10 and top100 reports for both HTC and Cloud.
- Added Year, Half-year and Quarter granularity in all the views.
- Improved the scientific discipline view.
- Reorganization of the menus according to the EGI service catalogue.
- Sub-discipline views switch to VO-based view when no sub-disciplines 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.
- Improved table visualisation.
- Reimplemented VO metrics support.

7.4 Feedback on satisfaction

Several tests were executed in collaboration with the EGI UCST and Operations Team. User communities were involved in the testing phase and the portal was updated according to the gathered requirements.

Feedback collected on the final release by all the stakeholders involved in the testing phase was very positive.

7.5 Plan for Exploitation and Dissemination

Name of the result	Accounting Portal
DEFINITION	
Category of result	Software & service innovation
Description of the result	Completed refactored portal with a modern and more attractive look & feel and several new features such as new home page, a WLCG specific sub-portal, new

	EGI reports, improved scientific discipline support, reorganized menus, contextualised help inline, improved CSV support, reimplemented VO metrics.
EXPLOITATION	
Target group(s)	Infrastructure users, VO Managers, Operations Centres, Resource providers and the general public.
Needs	Modern look & feel, 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	http://accounting-next.egi.eu
Success criteria	Continued use.
DISSEMINATION	
Key messages	A modern accounting portal with several new features is now available.
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	Number of accesses.

7.6 Future plans

Two new releases are planned by the end of the project, one in May and another one in August.

The May release will include:

- A complete API to get accounting data directly from the accounting portal;
- Maps showing the graphical distribution of the accounting data;
- Additional options to aggregate data;
- Support of the new cloud usage record;

- Report to generate summaries about VOs that belong to the same discipline category;
- Revised restricted views in the new accounting portal
- Bug fixing.

The August release will include:

- Reports for spotting increasing/decreasing VO usage;
- Accounting data analytics;
- Dynamic pie charts;
- Change type of graph dynamically
- Support GPGPU Accounting;
- Support Data Accounting;
- Admin role;
- Bug fixing.

Appendix I. ARGO Development Process

The following text is a copy of the "ARGO Development Process" document. The latest version of the document can be found here:

https://docs.google.com/document/d/1W0pT-zcBHG1E_hfftW67DH01LBZC7zMKLlIgJTlsFh8/edit#

Open development

We follow an open development process. All the repositories of ARGO are hosted on GitHub under the ARGOeu organization. Each component that can be standalone, is hosted in its own repository in the ARGOeu organization.

Each component should have a CONTRIBUTING guidelines document, describing how contributions can be made. There will be a general CONTRIBUTING guidelines document. Components that are maintained in their own repositories can should link to the general CONTRIBUTING guidelines document or have their own set of guidelines if required.

- https://github.com/ARGOeu

Forked repositories

Following the spirit of DVCS, each of us forks the repositories from GitHub to her/his own account. We can work on new or on-going features on our own forks and when we feel it is ready or whenever we want feedback from the rest of the team, and then we can open a pull request towards the respective ARGO repository.

Useful information:

- https://help.github.com/articles/fork-a-repo
- https://help.github.com/articles/syncing-a-fork

Pull requests & core team

All of the members of the core team should be able to merge pull requests in the ARGO repositories. The person who opens a pull request never merges it {her,him}self, but asks/expects another core team member to review it and merge it. The idea behind this is that at least two people (the committer and the reviewer) will be involved for each new feature that we develop.

Advices for a committer:

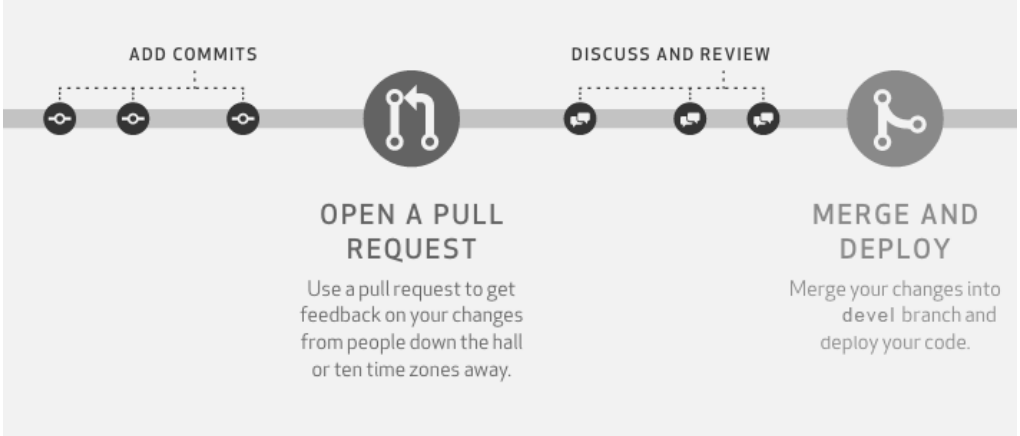
- Do commit early and often
- Do make useful commit messages (they will be used for the release CHANGELOG).

Creating insightful and descriptive commit messages is one of the best things you can do for others who use the repository. It lets people quickly understand changes without having to read

code. When doing “history archaeology” to answer some question, good commit messages become very important.

Format of a commit message:

- Title: [Jira issue ID] - descriptive title
- Description: summary of your job with enough information so that a can understand the context and the intention of the change.



The person who opens a pull request should make sure that {s}he includes enough information so that the reviewer can understand the context and the intention of the changes proposed in the pull request. A member can use the PULL_REQUEST_TEMPLATE that is supported by GitHub since earlier this year. https://github.com/blog/2111-issue-and-pull-request-templates. It is strongly encouraged that we open pull requests as soon as possible in the developer process in order to trigger prompt feedback.

1 pull request should refer to 1 feature, task, bug. Pull requests that are not ready to be merged should be marked as Work-In-Progress (WIP). Having the pull request open, means that each commit is visible to the ARGO CI, which can then build the component, run all the unit tests and attempt to package the component and at the end provide status feedback within the pull request.

Useful information:

- https://help.github.com/articles/creating-a-pull-request
- https://help.github.com/articles/checking-out-pull-requests-locally
- https://help.github.com/articles/creating-a-pull-request
- https://help.github.com/articles/merging-a-pull-request
- https://quickleft.com/blog/pull-request-templates-make-code-review-easier
- https://help.github.com/articles/merging-a-pull-request

Pull request review process

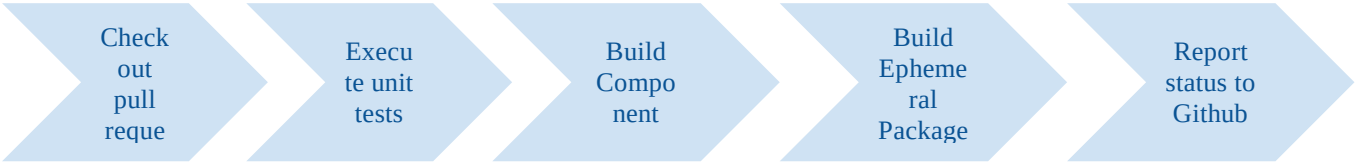
When a feature is ready, the developer removes the WIP mark from the pull request. Removing the WIP mark effectively signals the rest of the team that the pull request can be peer reviewed. At least one team member (other than the committer) has to act as the reviewer of the pull request. During the peer review process, the reviewer has to check the feature implemented, the code quality, the unit test coverage as computed, the existence of proper documentation and whether the component can be packaged successfully. If all these checks pass, then the reviewer can accept the pull request in order to be merged in the devel branch.

Branches and builds

Each repository should have at least 2 long-term branches:

- the devel branch, which should always be deployable
- the master branch, which should always be releasable
- Pull requests

Pull requests for new features should be opened initially against the devel branch. For every pull request that is opened, the ARGO CI will execute the following workflow



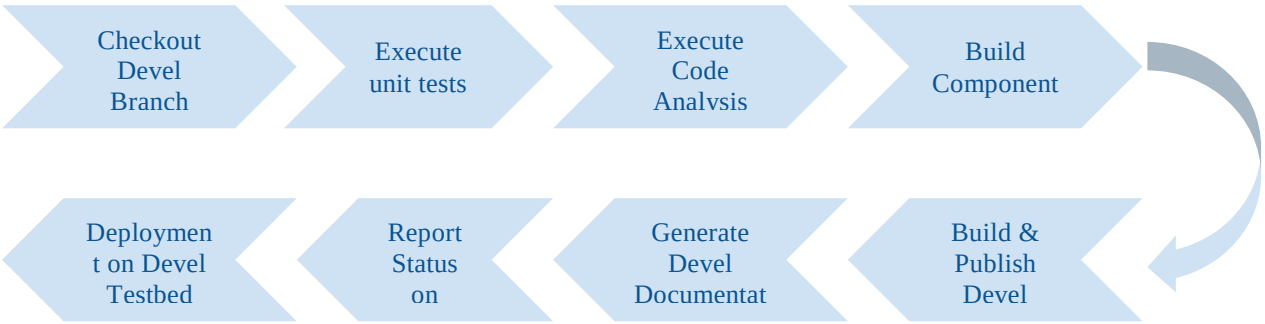
Before a pull request can be merged in the devel branch, a member of the development team (other than the original committer) has to review the pull request and check the following according to the “Definition of Done”:

#	Check	Status
1	Quality of Code	
2	Passes acceptance criteria automatic Unit tests for non-UI (80% or greater code coverage for business logic tier for new code)	
3	CI build job is up-to-date and compiles, tests, and analyses the existing & newly added code	
4	DB migration script for DB Schema tasks	
5	Sufficient documentation: • APIs + Interfaces (public)	

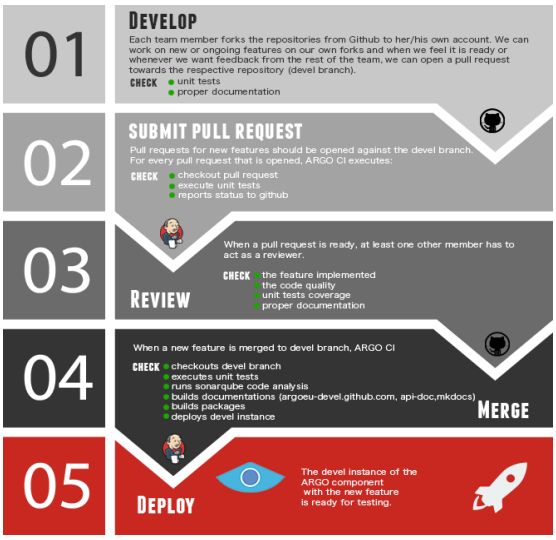
	• Manuals (where applicable) • Changelog / Release Notes • Inline comments where 'complex' code	
6	Ability to be properly packaged	

Devel branches

When new code is merged on the devel branch of each component, the CI system (a) picks it up, (b) builds the codebase, (c) runs again the unit tests, (d) runs the sonarqube code analysis suite and publishes the results on the ARGO sonarqube instance, (e) builds the devel packages and publishes them on the ARGO devel RPM repository, (f) extracts, builds the documentation and publishes it on the devel website and (g) reports the status of the CI on Github. New RPMs published on the devel RPM repository are automatically installed on the devel testbed.



The devel testbed is using actual production data and is being operationally monitoring by the same monitoring probes that are used to monitor also the production instance. Furthermore at the end of each sprint, the product team performs the sprint review ceremony in which the important features are presented to the ARGO stakeholders and live tested on the devel testbed. After the successful completion of the sprint review, the new code base is merged on each component's master branch.



In case more than one developer is working on the same component or a developer is working in parallel in more than one feature for the same component, the use of feature branches is advised.

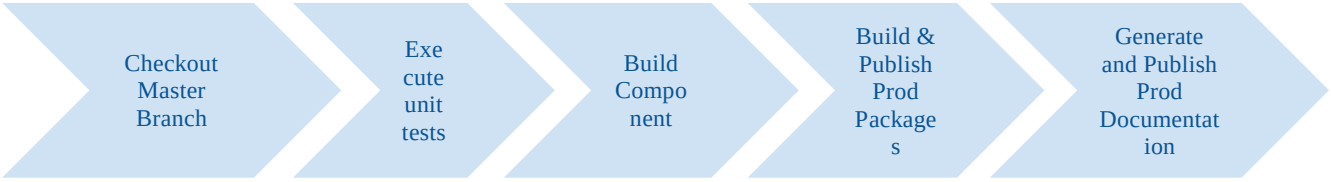
The Devel branch is considered to be the main branch where the source code of HEAD always reflects a state with the latest delivered

development changes for the next release. Some would call this the **"integration branch".** This is where automatic builds are built from.

When the source code in the develop branch reaches a stable point and is ready to be released, all of the changes should be merged back into master somehow and then tagged with a release number.

Master Branches

When new code is merged in the master branch of each component, the CI system picks it up and execute the follow workflow: (a) builds the codebase, (b) runs the unit tests again, (c) builds the production packages, (d) publishes them on the ARGO production RPM repository and (e) extracts & builds the documentation and publishes it on the ARGO website.



Each time changes are merged back into master; this ***is a new production release by definition***.

Useful information:

- http://martinfowler.com/bliki/FeatureBranch.html

Releases

The release follows the process when new code is merged in the master branch of each component. Some prerequisites for a helpful release:

Spec files should follow the correct release number shown in the following table. Spec files (%changelog) should not contain information about features or fixes, but information about changes in the package[15]. Do NOT put software's changelog at here. This changelog is for RPM itself. If the package has no changes, the description should say "New RPM package release".

Release: New release is created in the component repository. (Go to releases → Draft new release) The release contains the release number and detailed information. The information is created via the PR descriptions, so the PRs should have descriptive titles and messages. The release description should have the following sections:

[15] https://docs.fedoraproject.org/en-US/Fedora_Draft_Documentation/0.1/html/Packagers_Guide/sect-Packagers_Guide-Creating_a_Basic_Spec_File.html

```
# New features/Enhancements

# Fixes

# Documentation updates
```

Release numbers

v1.0.**[1]**	Patch release. A new minor release typically including just backwards-compatible bug fixes. No new functionality is added.
v1.**[1]**.1	Feature release. MINOR version when you add new functionality in a backwards-compatible manner.
v**[1]**.1.1	Major release. Significant changes in the functionality. Mandatory if the changes are breaking backward compatibility.

A todo list of a release is described in this document.

Releases process

Planning: On every **first meeting** of the month we plan the new features, functionalities (jira tasks) of the components. It is not obligatory to have new features, functionalities, fixes for all components. For the planning process a Jira Sprint will be used, with the selected jira tasks. It will be nice to comment and update the status of each Jira task.

Testing: All the new features, functionalities and fixes must be tested for 2 weeks at least in the devel infrastructure. This effectively means that, in the next release, only the features that are ready to be tested in the middle of the month will be included.

Release: All tested features, functionalities and fixes will be deployed to the production infrastructure at the beginning of the next month. If a feature, functionality, fix is not properly tested or requires more development it will be added to the next release.

Appendix II. GOCDB development process

Testing:

- The GOCDB source code includes DBUnit and Unit tests for selected core packages. For a data-centric product like GOCDB, emphasis is placed on the DBUnit tests, which are essential to assert expected behaviour on the deployed RDBMS.
- The GOCDB test suite prioritizes quality functional testing of the most critical code-paths rather than achieving high blanket coverage of less meaningful tests.
- As of Jan/2016 this includes 67 DBUnit tests with 668 assertions.
- Coverage reporting is included for selected core packages (DAOs – 55%, Doctrine 35%, Gocdb_Services 17%) and it is acknowledged that a higher coverage should be achieved for these packages.
- Continuous Integration is not yet supported but will be investigated in future.

Approach to Source Control:

- The GOCDB project is hosted in GitHub under the GOCDB organization.
- The main GOCDB repository has two main branches ‘master’ and ‘dev’.
- The master branch is always ‘releasable’.
- The dev branch is always ‘deployable’.
- Developers fork the repository into their own personal repository to work on features using Topic branches.
- When ready, a pull request is opened against the ‘dev’ branch in the main repository for review by other team members.
- After review, the pull request is merged into the ‘dev’ branch.
- When ready, the dev branch is merged into master.
- Tags are subsequently created from the master branch to identify specific releases (v5.5. v5.6 etc).
- Throughout this process, the test suite is continuously executed and any failing tests addressed before creating pull requests and/or merging.
- For certain scenarios, we consider it acceptable to push commits directly to the dev branch rather than always enforcing pull requests which may add unnecessary overhead, such as making documentation changes or small rendering updates.