EGI-Engage

First release of the Operational tools

D3.4

Date	7 March 2016
Activity	WP3
Lead Partner	CNRS
Document Status	FINAL
Document Link	https://documents.egi.eu/document/2679

Abstract

This deliverable describes the first release of the EGI Operational Tools during EGI-Engage project, including the developments made during the first year of the project for the Operations Portal, ARGO, GOCDB and Security Monitoring. The evolution of these tools have 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 Centers (CCs) and the Resource Providers (RPs) who contribute infrastructure services to the federation. The development roadmap has been defined 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	CNRS GRNET STFC CESNET	16/02/2016
Moderated by:	Małgorzata Krakowian	EGI.eu/NA1	
Reviewed by	P. Solagna Giuseppe La Rocca Alessandro Paolini Claire Devereux	EGI.eu/SA1 EGI.eu/SA2 EGI.eu/SA1 STFC/PMB	22/02/2016 17/02/2016 17/02/2016 25/02/2016
Approved by:	AMB and PMB		9/03/2016

DOCUMENT LOG

Issue	***Date***	***Comment***	***Author/Partner***
v.1	16.02.2016	Version ready for external review	Cyril Lorphelin Christos Kanellopoulos David Meredith Daniel Kouril
FINAL	07.03.2016	Final version	Diego Scardaci Cyril Lorphelin Christos Kanellopoulos David Meredith Daniel Kouril

TERMINOLOGY

A complete project glossary is provided at the following page: http://www.egi.eu/about/glossary/

Contents

Executive summary

This deliverable describes the first release of several EGI Operational Tools, such as: the Operations Portal, ARGO, GOCDB and the Security Monitoring tools, developed during the first year of the EGI-Engage project. The technological 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 (RIs) contributing to EGI-Engage via the EGI Competence Centers (CCs) and the Resource Providers (RPs) who contribute infrastructure services to the federation. The development roadmap has been defined 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 features have been broadened with the integration of the VO Administration and operations Portal (VAPOR), which now supports cloud technology and has been extended with a GLUE2 based resource browser. Further information has been added to the VO ID card and proper interfaces are available to retrieve the data stored in the portal.

The latest release of ARGO offers the multi-tenants support. This new functionality provides a Monitoring as a service to communities within the EGI collaboration. The set of probes that monitors the EGI Federated Cloud resources, have been expanded to validate more functionalities. A centralized architecture for the EGI infrastructure monitoring has been designed and will be deployed in production in the first half of 2016. It will allow a more agile management of the whole EGI monitoring system.

During the first year, the GOCDB team focused its effort on developing new features to better support multiple projects in the same server instance and easily extend the GOCDB data model to satisfy community needs. In addition, the log system has been improved and a fine-grained access model has been implemented to restrict the access for unauthorized users.

Finally, EGI security experts released the first version of SECANT, a tool for monitoring virtual machines running in the EGI Federated Cloud.

Furthermore, Operations Portal, ARGO and GOCDB product teams have worked together to make their AAI architectures compliant with this new EGI access layer. As result of this activity, GOCDB has already implemented a new AAI architecture to authenticate users via Federated Identity Management (FIM) system, and Operations Portal and ARGO has started the development of their new AAI modules which will be completed in 2016. By the end of the second year the three tools will be accessible by end users via the new EGI AAI.

Finally, EGI security experts released the first version of SECANT, a tool to monitor virtual machines running in the EGI Federated Cloud.

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 **V**o **A**dministration and operations **POR**tal, 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.
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	OPERATIONS PORTAL: https://forge.in2p3.fr/projects/opsportaluser/wiki/Main_Features_of_the_dashboard VAPOR : http://operations-portal.egi.eu/vapor_dev/globalHelp
Technical Documentation	https://forge.in2p3.fr/projects/opsportaluser/wiki/Main_Features_of_the_dashboard
Product team	IN2P3/CNRS
License	Apache2
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. Finally, 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 there are 2 critical dependencies: GGUS[1] and RTIR[2] (red arrows on the right of Figure 1).

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.

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 the GOCDB and also local scripts.

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

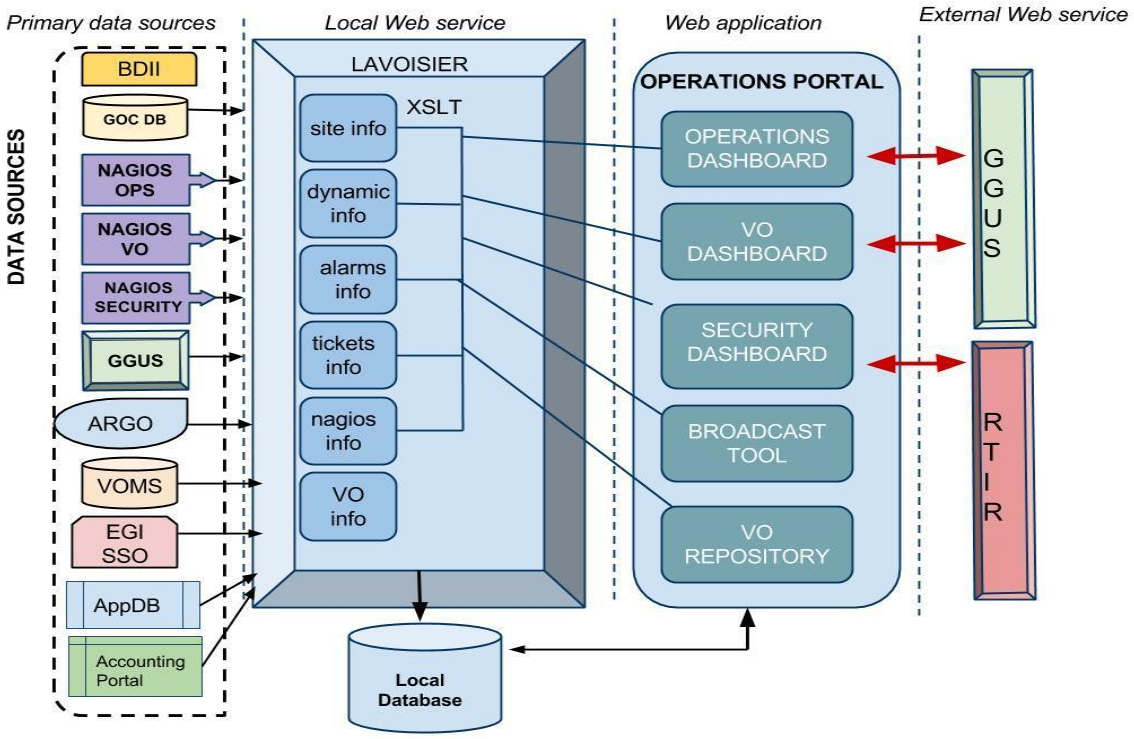


Figure 1 Operational Portal Architecture

1.2.2 Integration and dependencies

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

1.3 Release notes

This section describes separately the development done within the historical Operations Portal and the new application called Vapor, which has been integrated in the portal during the project year 1.

1.3.1 Requirements covered: Operations Portal

During this period, we have improved existing features and we have also adapted some of them for new/emerging needs.

1.3.1.1 Release 3.2

Release Note: http://operations-portal.egi.eu/home/tasksList/release_id/12

Main achievements:

- The possibility to create from the security dashboard a ticket into EGI RTIR;
- Metrics for the security dashboard;
- Changes into the overview of the security dashboard;
- Modification on the update of the age of the alarms;
- Improvement on the metrics related to users.

1.3.1.2 Release 3.2.1

Release Note: http://operations-portal.egi.eu/home/tasksList/release_id/12

Main achievements:

- A new module related to metrics[3] has been added:
  - used for EGI reports
  - based on 6 months period
  - about users distribution over disciplines, CA , VO, and incoming VOs
  - possibility to export data in different formats
  - the details by month are available if needed

1.3.1.3 Release 3.2.2

Release Note: http://operations-portal.egi.eu/home/tasksList/release_id/18

Main achievements:

- A section dedicated to VO acknowledgment statement has been added in the VO ID Card.

1.3.1.4 Release 3.3

Release Note: http://operations-portal.egi.eu/home/tasksList/release_id/19

Main achievements:

- **Security dashboard**
  - Notifications about security issues[4];
  - Capture WN instead of host: the Worker node is displayed (when available) instead of the host name into the list of issues;
  - Metrics: put in place reports for the Nagios issues[5];
  - Sites under certification process: extended the list of sites to display potential issues on candidate sites.
- **VO ID cards**

[3] http://operations-portal.in2p3.fr/metrics/metricsReports

[4] https://operations-portal.egi.eu/csiDashboard/notificationsIssues

[5] https://operations-portal.egi.eu/csiDashboard/report

- New section in the VO ID Card for Portal/Web Service robot certificates[6] including support for Per-User Sub-Proxies[7];
- Add filter input in the VO update page.

- **Dashboard**
  - Non-production alarms in ROD dashboard: if an alarm is raised on a service which is not production in GOCDB it will not be visible in the Operator page;
  - The template of the tickets has been improved.

1.3.2 Requirements covered: Vapor

Main achievements:

- **Upgrade of Vapor configuration**
  - The application has been developed during the EGI-InSPIRE project[8] and was deployed partially at I3S[9] in Nice. Therefore, a part of the work was to migrate it at CCIN2P3 to be integrated with the operations portal;
  - During this phase we have also upgraded the configuration of Lavoisier to be compliant with last version.
- **Update of the Glue Schema**
  - 'Translation' of the queries from Glue1.3 to Glue2;
  - Reviewed the architecture to improve performances and to extend a part of the application to all VOs (currently restricted to a subset of VOs: biomed, compchem, enmr.eu, vlemed, shiwa-workflow.eu, see, sagrid, vo.france-grilles.fr).
- **Capture Cloud Resources**
  - Included new functionality to capture cloud resources;
  - The cloud resources are now visible into the resource browser[10].
- **New Module: GLUE2 resource browser**
  - The aim of this tool is to give an overview of the EGI resources;
  - The information is based on Glue2 publication into Top-BDII;
  - The main features are:
    - Distribution of the resources over NGI, Sites and VO;

[6] https://operations-portal.egi.eu/vo/rbCert

[7] https://wiki.egi.eu/wiki/Long-tail_of_science#Per-user_sub-proxies

[8] https://www.egi.eu/about/egi-inspire/

[9] http://www.i3s.unice.fr/

[10] The set of information associated to the cloud resources will be extended according to the future evolution of the EGI Federated Cloud.

- Information about Computing, Storage and other services (VOMS, LFC, BDII);
- Information about Access/Mapping Policy;
- Information about statuses of the services and the potential downtimes;
- Different format to export information: CSV, XML, JSON;
- Information about cloud services;
- Information about badly published resources.

1.4 Feedback on satisfaction

Different "thematic" releases of the software have been scheduled regularly to test and check the new functionalities with some targeted users (e.g. for a VO oriented release the tests will be done by VO experts, for a dashboard oriented release tests will be performed by operators, etc.).

These different test phases have been coordinated with the OTAG[11] team.

Some releases have been postponed by lack of feedback or by lack of testers.

Release ID	Release Date	Status	BugFix	Testing	Prototype
3.2	26.05	Postponed due to a lack of details into the feedback	25-26.05	18.05-22.05	15.05
3.2	07.07	Released	23-26.06	16-22.06	15.06
3.2.1	26.08	Postponed to September due to a lack of testing users	24-26.08	19-24.08	
3.2.1	29.09	Released			
3.2.2	05.11	Released	03-05.11	27-29.10	
3.3	28.12	Released	27-28.12	20-26.12	

For the VAPOR application, a prototype[12] is available since December 2015 and we are currently collecting feedback to provide a release in production before the end of February.

1.5 Future plans

1.5.1 Operations Portal

- **Software quality (February – August 2016)**

[11] https://wiki.egi.eu/wiki/OTAG#Operations_Portal_Advisory_and_Testing_Board
[12] http://operations-portal.egi.eu/vapor_dev

- Migration of the whole code under gitlab;
- Integration of the release process with gitflow;
- Automation of deployment;
- Migration from symfony 1.5 to symfony 3:
    - with a refactorisation of the code,
    - with the multiplication of unit tests.

- **Features (April – September 2016)**
    - Complete rewriting of the Downtime Notification System;
    - Integration of Perun authentication system;
    - Support for federated logins using SAML.

1.5.2 Vapor

- **Software quality (February – July 2016)**
    - Usage of SonarQube: open source software used to measure the code quality and improve the continuous integration.
- **Features (April – December 2016)**
    - Replace Gstat[13] Main Features;
    - Extend APIs;
    - Integrate cloud monitoring to be discussed with Cloud Working Group;
    - Support for federated logins using SAML.

[13] http://gstat.egi.eu/

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
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 allows the integration 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.

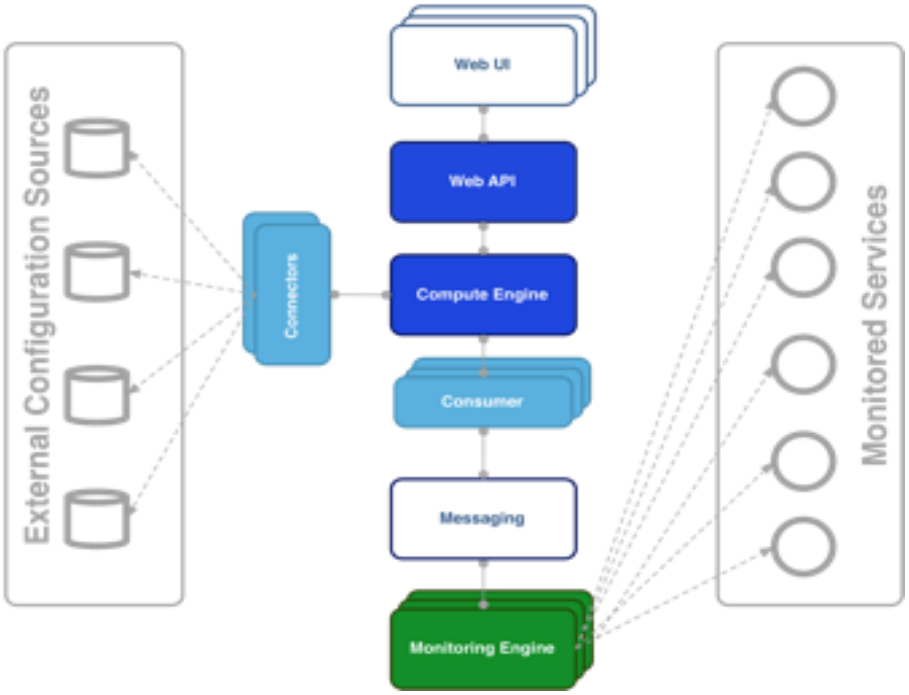


Figure 2 ARGO architecture

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[14]. 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.

2.2.1.2 Messaging

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[15] services and uses the STOMP protocol[16] 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 more robust.

[14] https://www.nagios.org/

[15] http://activemq.apache.org/

[16] https://stomp.github.io/

2.2.1.3 The ARGO Connectors

With custom connectors, ARGO can connect to multiple external Configuration Management Databases and Service Catalogs. There are already connectors for the EGI and EUDAT e-Infrastructures[17].

2.2.1.4 The ARGO Consumer

The ARGO Consumer is ingesting monitoring results in real-time from external Message Brokers and it is responsible for the initial pre-filtering of the monitoring results that are encoded using AVRO serialization format[18] and then passed to the Compute Engine.

2.2.1.5 The ARGO Compute Engine

The ARGO Compute Engine is a powerful and scalable analytics engine built on top of Hadoop and HDFS[19]: it is responsible for the aggregation of the status results and the computation of availability and reliability of composite services using customer defined algorithms.

2.2.1.6 The ARGO Web API

The ARGO Web API provides the serving layer of ARGO. It is composed 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 of the monitored resources and the actual raw metric results.

2.2.1.7 The ARGO Web UI

The default web UI is based on the Lavoisier Data Aggregation Framework[20].

2.2.2 Integration and dependencies

Insert 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.
- GSTAT: It is used as the source for custom factor values, which in the case of EGI it is the HEPSPEC values of the sites.
- SAM ATP: It is used as the source of Service Endpoint information for the Grid sites.

[17] http://eudat.eu/

[18] https://avro.apache.org/docs/1.2.0/

[19] http://hadoop.apache.org/

[20] http://software.in2p3.fr/lavoisier/

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.

2.3 Release notes

2.3.1 Requirements covered in the release

As already mentioned, ARGO is not a single software, but a suite of software components, each one managed independently. During the first year of the project, there have been 15 releases of the ARGO components that covered the following requirements:

ARGO Compute Engine & Web API

- Support for automatic re-computation triggers;
- Support for multiple tenants;
- Specification of the Data Ingestion API;
- Specification and implementation of APIv2;
- Delivery of computed status results through the API.

ARGO Monitoring Engine

- Probe framework;
- EGI Federated Cloud probes;
- Implementation of central monitoring engine;
- Support documentation guides.

ARGO EGI Consumer and Connectors

- improved support for VOs.

ARGO EGI Web UI

- ACL mechanism (support groups/roles);
- UI Enhancements;
- Initial support for federated logins using SAML.

ARGO POEM

- ACL mechanism (support groups/roles);
- Initial support for federated logins using SAML;
- Design of probe publishing and management service.

2.3.1.1 Changelog

- 24/12/2015
  - POEM [v0.11.0-4]: https://github.com/ARGOeu/poem/releases/tag/v0.11.0-4
- 30/11/2015
  - EGI Web UI [v1.1.2-1]: https://github.com/ARGOeu/argo-egi-web/releases/tag/v1.1.2-1
- 24/11/2015
  - Compute Engine [v1.6.5-2]: https://github.com/ARGOeu/argo-compute-engine/releases/tag/v1.6.5-2
- 29/10/2015
  - Compute Engine [v1.6.5-1]: https://github.com/ARGOeu/argo-compute-engine/releases/tag/v1.6.5-1
  - Web API [v1.6.0-3]: https://github.com/ARGOeu/argo-web-api/releases/tag/v1.6.0-3
  - EGI Consumer [v1.4.1-1]: https://github.com/ARGOeu/argo-egi-consumer/releases/tag/v1.4.1-1
  - EGI Connectors [v1.4.4-6]: https://github.com/ARGOeu/argo-egi-connectors/releases/tag/v1.4.4-6
  - EGI Web UI [v1.1.0-1]: https://github.com/ARGOeu/argo-egi-web/releases/tag/v1.1.0-1
- 11/09/2015
  - EGI Connectors [v1.4.3-3]: https://github.com/ARGOeu/argo-egi-connectors/releases/tag/v1.4.3-3
- 23/07/2015
  - Compute Engine [v1.6.2-7]: https://github.com/ARGOeu/argo-compute-engine/releases/tag/v1.6.2-7
  - EGI Connectors [v1.4.2-2]: https://github.com/ARGOeu/argo-egi-connectors/releases/tag/v1.4.2-2
- 17/07/2015
  - Compute Engine [v1.6.2-6]: - https://github.com/ARGOeu/argo-compute-engine/releases/tag/v1.6.2-6
- 01/07/2015
  - EGI Web UI [v1.0.0-1]: https://github.com/ARGOeu/argo-egi-web/releases/tag/v1.0.0-1
- 30/06/2015
  - EGI Connectors [v1.4.2-1]: https://github.com/ARGOeu/argo-egi-connectors/releases/tag/v1.4.2-1
- 11/06/2015
  - EGI Web UI [v0.1.12-1]: https://github.com/ARGOeu/argo-egi-web/releases/tag/v0.1.12-1

- 03/06/2015
  - Compute Engine [v1.6.2-1]: https://github.com/ARGOeu/argo-compute-engine/releases/tag/v1.6.2-1
  - Web API [v1.6.0-1]: https://github.com/ARGOeu/argo-web-api/releases/tag/v1.6.0-1
  - EGI Consumer [v1.4.0-15]: https://github.com/ARGOeu/argo-egi-consumer/releases/tag/v1.4.0-15
  - EGI Connectors [v1.4.1-5]: https://github.com/ARGOeu/argo-egi-connectors/releases/tag/v1.4.1-5
- 31/03/2015
  - Compute Engine [v1.6.1-1]: https://github.com/ARGOeu/argo-compute-engine/releases/tag/v1.6.1-1
  - POEM [v0.10.7-2]: https://github.com/ARGOeu/poem/releases/tag/v0.10.7-2
  - EGI Web UI [v0.1.8-1]: https://github.com/ARGOeu/argo-egi-web/releases/tag/v0.1.8-1
- 04/03/2015
  - Compute Engine [v1.6.0-6]: https://github.com/ARGOeu/argo-compute-engine/releases/tag/v1.6.0-6
  - Web API [v1.5.1-4]: https://github.com/ARGOeu/argo-web-api/releases/tag/v1.5.1-4)
  - EGI Connectors [v1.3.1-16]: https://github.com/ARGOeu/argo-egi-connectors/releases/tag/v1.3.1-16
  - POEM [v0.10.6-3]: https://github.com/ARGOeu/poem/releases/tag/v0.10.6-3
  - EGI Web UI [v0.1.5-1]: https://github.com/ARGOeu/argo-egi-web/releases/tag/v0.1.5-1
- 06/02/2015
  - Compute Engine [v1.6.0-2]: https://github.com/ARGOeu/argo-compute-engine/releases/tag/v1.6.0-2
  - Web API [v1.5.1-2]: https://github.com/ARGOeu/argo-web-api/releases/tag/v1.5.1-2
  - EGI Consumer [v1.3.2-8]: https://github.com/ARGOeu/argo-egi-consumer/releases/tag/v1.3.2-8
  - EGI Connectors [v1.3.1-12]: https://github.com/ARGOeu/argo-egi-connectors/releases/tag/v1.3.2-8
  - EGI Web UI [v0.1.0-1]: https://github.com/ARGOeu/argo-egi-web/releases/tag/v0.1.0-1

2.4 Feedback on satisfaction

The initial aim of ARGO was to replace the historical component SAM and especially the Web UI “My EGI”. This part has been completed and the new Web UI has been available in advance.

Different reviews have been coordinated with EGI Operations team to be sure that all needs were covered and all features were properly developed. After these reviews some corrections have been done to satisfy plenty the operations team.

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 Future plans

ARGO Compute Engine

- API for data ingestion specification;
- Separation of A/R and Metric stores;
- API for data ingestion implementation;
- Support for multiple monitoring engines running in active-active setup APIv2;
- Stability and performance improvements.

ARGO Monitoring Engine

- Completion of the Centralised Monitoring Engine;
- Fedcloud probes update;
- Stability and performance improvements.

ARGO Web UI

- UI Enhancements;
- Connect to the EGI IdP/SP Proxy.

ARGO EGI Consumers and Connectors

- Use of CE ingestion API;
- Stability and performance improvements.

ARGO POEM

- Support for probe management;
- Connect to the EGI IdP/SP Proxy;
- Stability and performance improvements.

ARGO Messaging

- APIv1 test implementation;

- APIv1 final draft specification (ready for external party review);
- APIv1 final implementation;
- APIv1 final specification.

3 GOCDB

3.1 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 centres (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 and object-tagging for the purposes of fine-grained resource filtering.
Customer of the tool	EGI Operations
User of the service	Site/service admins, NGI managers and, Security teams in EGI, EUDAT and WLCG.
User Documentation	https://wiki.egi.eu/wiki/GOCDB/Documentation_Index
Technical Documentation	https://wiki.egi.eu/wiki/GOCDB/Documentation_Index
Product team	Contact gocdb-admins@mailman.egi.eu
License	Apache2
Source code	https://github.com/GOCDB/gocdb

3.2 Service architecture

3.2.1 High-Level Service architecture

GOCDB is a central information repository providing a web portal interface for CRUD operations (create, read, update, delete), and a REST API for read-only data queries.

It is a definitive information source where data is directly populated and managed manually by users. 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, for example, it does not query other systems to populate its core data model (note, augmenting the manually populated 'core' data model with supplementary data derived from other systems would be possible for certain scenarios). In future it will rely on the EGI Proxy IdP/SP to authenticate users via Federated Identity Management (FIM) system with a sufficient level of assurance (LoA) needed for operational users.

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.

3.2.2 Integration and dependencies

As a primary source of information, GOCDB does not depend on any other tool.

3.3 Release notes

3.3.1 Requirements covered in the release

The current production release is v5.5 with v5.6 released for testing. The main requirements addressed during PY1 are listed below, with more details available at:

- Prioritized Roadmap[21];
- Full change log[22] (includes smaller changes/bug fixes);
- All active/pending requirements in RT[23].

v5.4 (06/07/2015):

- Role Action Logging to record all role request actions (deny, approve and revoke).
- Fine Grained Content Rendering including permit-all and protected pages.
- Downtime Declaration in Site-Local Timezone with automatic conversion to UTC.

v5.5 (02/12/2015):

- Multi-Tenant to host multiple projects in the same GOCDB instance with separate roles per project (roles in one project do not apply in a different project). This was a large scale development and required significant re-write of the authorisation model.
- SAML/FIM Authentication to allow login using different Identity Providers. Please note, issues related to data protection and the ELIXIR/ GEANT Data Protection Code of Conduct[24]

[21] https://wiki.egi.eu/wiki/EGI-Engage:TASK_JRA1.4_Operations_Tools#GOCDB

[22] https://github.com/GOCDB/gocdb/blob/dev/changeLog.txt

[23] https://rt.egi.eu/rt/Dashboards/5541/GOCDB-Requirements

delay the production release of FIM. This is an EGI wide issue - all Ops tools need to use the same eduPersonPrincipalName: ePPN (most likely hashed given data protection concerns) in order to correlate accounts across services.

v5.6 (released for testing 02/2016, at the time of writing, not yet in production):

- Reserved (Protected) Scope Tags used to control the tagging of resources using cascading rules that limit which tags NGIs->Sites->Services->Downtimes can declare. This will allow WLCG/Elixir sites to apply tags only to their resources and prevent other sites from using the same tags. This allows controlled resource filtering in API and GUI.
- Bulk Addition/Upload of Multiple Custom Properties to allow the data models of Sites, ServiceGroups, Services and Endpoints to be easily extended and customized. Also allows fine-grained resource selection via the API by filtering on supported properties.
- Downtime Calendar to allow fine-grained filtering of downtimes.

3.4 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[25]. This testing phase is widely disseminated using the relevant mailing lists, and all operational tools and users are invited to perform tests against this instance.

The GOCDB development process is described in Appendix II.

3.5 Future plans

- **Medium Term (scheduled for v5.7+, mid to end 2016):**
  - Writable REST API to programmatically POST updates to sites and services. This has been requested by both WLCG and EUDAT operational communities. This will require site's to manage their own access control lists (ACLs) per site by recording the DNs of certificates that are authorised to post updates to a particular site/service.
  - Object Diff Auditing to record every change to a resource. Originally an EUDAT request, but now de-prioritised by EUDAT.
- **Longer Term (end 2016/17):**
  - The GOCDB Web interface needs to be re-implemented using a modern MVC GUI framework.

[24] https://wiki.refeds.org/display/CODE/Data+Protection+Code+of+Conduct+Home

[25] https://gocdb-test.esc.rl.ac.uk

4 Security Monitoring

4.1 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.
Customer of the tool	Cloud providers, 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

4.2 Service architecture

4.2.1 High-Level Service architecture

Secant is a new tool, which runs as a service that periodically checks for new images available for the monitored IaaS cloud instance 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 image from the image and template that is being verified and performs two phases of security checks. During the first phase Secant launches a series of external scans that try 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.

This is the very first release of Secant whose development started with the EGI-Engage project. The design and its implementation will be subject to changes and modification in the course of the subsequent evaluations.

4.2.2 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 EGI. 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.

4.3 Release notes

4.3.1 Requirements covered in the release

The design and first version of Secant was developed to cover requirements discussed in the EGI CSIRT, to reflect the main findings from security incidents that related to EGI:

- The first implementation of the framework as described in the documentation
- OpenNebula based management of VM's
- Several internal and external probes implemented
- Evaluation of results collected by the probes

4.4 Feedback on satisfaction

Secant is being tested at CESNET and its MetaCloud site. A well-structured testing framework will be available with the next releases.

4.5 Future plans

After an evaluation phase we are going to examine how Secant and the security assessment functionality can be integrated with AppDB to support the endorsement process. Based on evaluation of recent security incidents and threats, new security probes will be added.

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/

Open development

We follow an open development process. All the repositories of ARGO are hosted on Github under the ARGOeu organization[26]. 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.

Forked repositories

Following the spirit of DVCS (Distributed version control System), each of us forks the repositories from Github to her/his own account. We can work on new or ongoing features on our own forks and when we feel it is ready or whenever we want feedback from the rest of the team, 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[27]

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.

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. It is strongly encouraged that we open pull requests as soon as possible in the developer process in order to trigger prompt feedback. Pull requests that are not ready to be merged should be marked as Work-In-Progress (WIP). Having the pull request open, means that

[26] https://github.com/ARGOeu
[27] https://help.github.com/articles/syncing-a-fork

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/merging-a-pull-request
- https://quickleft.com/blog/pull-request-templates-make-code-review-easier

Pull request review process[28]

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

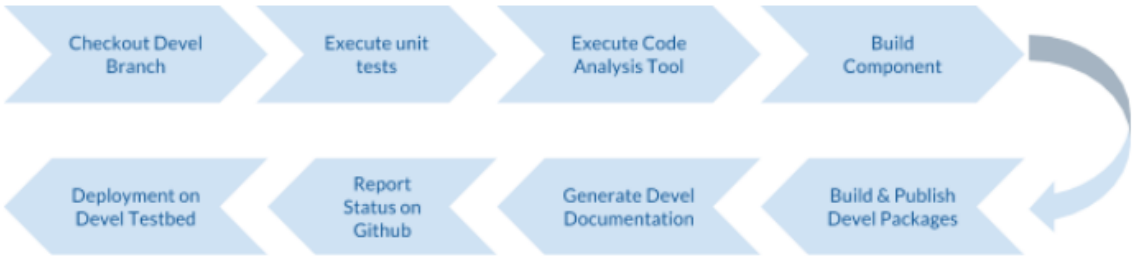
[28] https://help.github.com/articles/merging-a-pull-request

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

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.

Useful information:

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

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 behavior 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.