

EGI-InSPIRE

SOFTWARE PROVISIONING PROCESS

EU MILESTONE: MS516

Update on the software provisioning process

Document identifier:	EGI-InSPIRE- MS516-FINAL
Date:	01/08/2013
Activity:	SA2
Lead Partner:	EGI.eu
Document Status:	FINAL
Dissemination Level:	PUBLIC
Document Link:	https://documents.egi.eu/document/1861

Abstract

This document describes the process by which new product releases will be distributed through the EGI Software Repository, processed and released for deployment into production. This document is a revision of the MS512 milestone. It describes the assessment process and the criteria that will be applied to software components and outlines some of the component specific tests that may be applied as part of the software validation process.

I. COPYRIGHT NOTICE

Copyright © Members of the EGI-InSPIRE Collaboration, 2010-2014. See www.egi.eu for details of the EGI-InSPIRE project and the collaboration. EGI-InSPIRE (“European Grid Initiative: Integrated Sustainable Pan-European Infrastructure for Researchers in Europe”) is a project co-funded by the European Commission as an Integrated Infrastructure Initiative within the 7th Framework Programme. EGI-InSPIRE began in May 2010 and will run for 4 years. This work is licensed under the Creative Commons Attribution-Noncommercial 3.0 License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc/3.0/> or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, and USA. The work must be attributed by attaching the following reference to the copied elements: “Copyright © Members of the EGI-InSPIRE Collaboration, 2010-2014. See www.egi.eu for details of the EGI-InSPIRE project and the collaboration”. Using this document in a way and/or for purposes not foreseen in the license requires the prior written permission of the copyright holders. The information contained in this document represents the views of the copyright holders as of the date such views are published.

II. DELIVERY SLIP

	Name	Partner/Activity	Date
From	Peter Solagna	EGI.eu/SA2	26/7/2013
Reviewed by	Moderator: Ales Krenek Reviewers: Tomasz Piontek	CESNET ICBP	22/7/2013 18/7/2013
Approved by	AMB & PMB		01/08/2013

III. DOCUMENT LOG

Issue	Date	Comment	Author/Partner
1	12/07/2013	Final version for review	Peter Solagna / EGI.eu
2	25/7/2013	Revised version.	Peter Solagna / EGI.eu
3	26/7/2013	Final revised version	Peter Solagna / EGI.eu

IV. APPLICATION AREA

This document is a formal deliverable for the European Commission, applicable to all members of the EGI-InSPIRE project, beneficiaries and Joint Research Unit members, as well as its collaborating projects.

V. DOCUMENT AMENDMENT PROCEDURE

Amendments, comments and suggestions should be sent to the authors. The procedures documented in the EGI-InSPIRE “Document Management Procedure” will be followed:

<https://wiki.egi.eu/wiki/Procedures>

VI. TERMINOLOGY

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



VII. PROJECT SUMMARY

To support science and innovation, a lasting operational model for e-Science is needed – both for coordinating the infrastructure and for delivering integrated services that cross national borders.

The EGI-InSPIRE project will support the transition from a project-based system to a sustainable pan-European e-Infrastructure, by supporting ‘grids’ of high-performance computing (HPC) and high-throughput computing (HTC) resources. EGI-InSPIRE will also be ideally placed to integrate new Distributed Computing Infrastructures (DCIs) such as clouds, supercomputing networks and desktop grids, to benefit user communities within the European Research Area.

EGI-InSPIRE will collect user requirements and provide support for the current and potential new user communities, for example within the ESFRI projects. Additional support will also be given to the current heavy users of the infrastructure, such as high energy physics, computational chemistry and life sciences, as they move their critical services and tools from a centralised support model to one driven by their own individual communities.

The objectives of the project are:

1. The continued operation and expansion of today’s production infrastructure by transitioning to a governance model and operational infrastructure that can be increasingly sustained outside of specific project funding.
2. The continued support of researchers within Europe and their international collaborators that are using the current production infrastructure.
3. The support for current heavy users of the infrastructure in earth science, astronomy and astrophysics, fusion, computational chemistry and materials science technology, life sciences and high energy physics as they move to sustainable support models for their own communities.
4. Interfaces that expand access to new user communities including new potential heavy users of the infrastructure from the ESFRI projects.
5. Mechanisms to integrate existing infrastructure providers in Europe and around the world into the production infrastructure, so as to provide transparent access to all authorised users.
6. Establish processes and procedures to allow the integration of new DCI technologies (e.g. clouds, volunteer desktop grids) and heterogeneous resources (e.g. HTC and HPC) into a seamless production infrastructure as they mature and demonstrate value to the EGI community.

The EGI community is a federation of independent national and community resource providers, whose resources support specific research communities and international collaborators both within Europe and worldwide. EGI.eu, coordinator of EGI-InSPIRE, brings together partner institutions established within the community to provide a set of essential human and technical services that enable secure integrated access to distributed resources on behalf of the community.

The production infrastructure supports Virtual Research Communities (VRCs) – structured international user communities – that are grouped into specific research domains. VRCs are formally represented within EGI at both a technical and strategic level.



VIII. EXECUTIVE SUMMARY

This document describes the changes in the Unified Middleware Distribution (UMD) release process implemented during the last year. Since the process has been detailed in the previous milestones MS508 and MS512 this document focuses on the improvements implemented in the process and the planned changes that will be deployed in the coming months.

During PY3 were implemented improvements across all the steps for the software provisioning process. New criteria have been defined to fulfil requirements coming from new middleware or operational requirements. The test-bed infrastructure improved to make the work of verifiers easier and reduce the overhead. The release candidate candidates are automatically tested for dependencies issues. The infrastructure has been extended to support multiple operating systems and major releases of UMD.

To adapt UMD to the changes in the technology ecosystem SA2 implemented new work group -the UMD Release Team- and a “repository as a service” tool, to support the work of product teams who want to contribute to UMD.

TABLE OF CONTENTS

1	INTRODUCTION	6
2	UMD SOFTWARE PROVISIONING PROCESS.....	7
2.1	Criteria Definition	8
2.2	Criteria verification	8
2.3	Staged Rollout.....	8
2.4	Building the release, the provisioning infrastructure	9
2.4.1	Producing and testing the release candidates	9
3	METRICS.....	11
3.1	Verification	11
3.2	UMD package downloads statistics system.....	11
4	UMD RELEASE TEAM	14
5	COMMUNITY SOFTWARE REPOSITORY.....	15
6	PLANS FOR THE FUTURE.....	17
7	CONCLUSIONS.....	19
8	REFERENCES.....	20



1 INTRODUCTION

This document describes the changes in the Unified Middleware Distribution (UMD) release process implemented during the last year. Since the process has been detailed in the previous milestones [MS508] and [MS512] this document focuses on the improvements implemented in the process and the planned changes that will be deployed in the coming months.

The overall process defined during the first two project years proved to be effective and to scale with the need of EGI's production infrastructure. The changes introduced in the different steps have reduced the effort overhead needed to handle the process, and extended the infrastructure to support multiple UMD major releases and three operating systems.

Chapter 2 describes the UMD software provisioning process focusing on the changes implemented in the different steps of the workflow, and how the changes improved the process. Chapter 4 and 5 describes the new activity (URT) and service (an extension of AppDB) introduced during PY3 to extend the software provisioning process. Chapter 6 summarizes the work currently undergoing to adapt the software provisioning infrastructure to the changes in the new technology ecosystem.

The previous milestones provided a detailed description of the workflows, therefore every section contains only a short description of the process involved, plus the changes introduced during this year.

2 UMD SOFTWARE PROVISIONING PROCESS

The software provisioning process is the workflow by which the software released by the product teams composes the UMD releases and are distributed through the EGI software repositories.

Improvements have been implemented in all the steps of the workflow, but the overall process did not change during PY3. The changes will be detailed in the next paragraphs.

A summary of the workflow can be described using Figure 1.

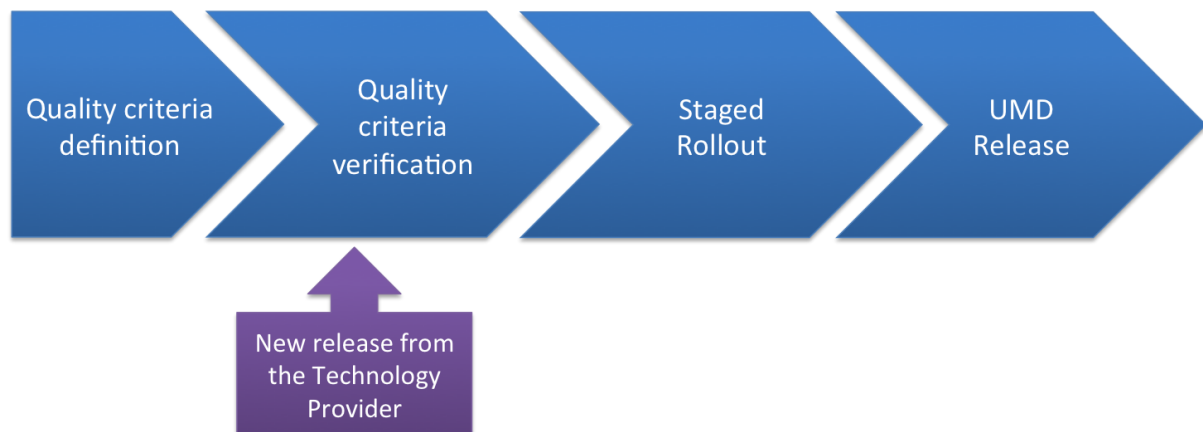


Figure 1, the UMD software provisioning workflow

The process can be summarized in four main steps:

Quality criteria definition: the software released in UMD must satisfy a set of criteria, these are functional criteria to test the capabilities provided by the software and general purpose software quality applicable to all the products. Every new software product added to the UMD portfolio is analysed to identify the capabilities provided. The capabilities are then associated to criteria to be verified during the software provisioning workflow. New criteria are added to cover new capabilities or to answer to specific requirements coming from the operations community.

Quality criteria verification: the new product releases are deployed in a testbed and verified against the associated quality criteria. This step is mainly performed within the SA2 team in the testbed provided by CESGA. The result of the verification is summarized in a verification report submitted to DocumentDB

Staged rollout: products that succeeded in the verification step are then deployed in a production environment. Expert site managers, familiar with the product under verification, install the new release in their resource centres, as production services. Doing this the services are exposed to real user and their use case, this allows a more extensive testing in addition to the capability tests performed during verification. The site administrator, at the end of staged rollout, submits a report with the outcome of the testing.

UMD release: the products successfully tested in verification and staged rollouts are candidates for the UMD release. A UMD release usually has a target major release, updates for different major releases are released independently, but includes the packages for multiple operating systems (SL5, SL6 and Debian). A release candidate is created with the products ready for the release and automatically tested for installation, before the official release.

The release notes include the information from the verification and staged rollout reports, these include: non critical issues, workarounds and installation tips.

2.1 Criteria Definition

The quality criteria definition activity continued producing two releases per year of the Quality Criteria Document, each one following two public drafts reviewed with the contribution of the technology providers. The updates in the quality criteria documents, in terms of new criteria, are summarized in [D59].

The task during PY3 consolidated on the EGI wiki the manuals for the verifiers, containing the how-to for the criteria verification. The how-to provides clear guidelines reducing the effort required from verifiers to perform the tests, and making uniform the tests results between different verifiers.

2.2 Criteria verification

The verification is performed using the resources of a dedicated testbed, which is a private cloud infrastructure hosted by CESGA. The main changes in the verification workflow are improvement in the cloud infrastructure, in terms of usability and efficiency.

The cloud service exposes a rOCCI¹ interface, used also in the EGI Federated Cloud testbed, authorized SA2 verifiers can autonomously deploy the machines as needed, without the need of actions from the administrators of the IaaS platform.

The contextualization scripts, which configure the operating system and the services on a freshly instantiated virtual machine with the test bed specific parameters, have been improved in order to allow verifiers to find a working environment ready in few minutes from their request of a new virtual machine. The SSH keys of the requestor are automatically deployed in the system, enabling verifiers to log in without configuring a password. The contextualization scripts make available also the configuration templates of the service under testing, in the file system of the virtual machine. The templates include the configuration details of the testbed environment and require only small edits to have the service configured and running.

The testbed virtual machine management infrastructure saves the virtual images of the working services deployed during verification as golden copies, to be used to re-deploy the service in few seconds if any further test becomes necessary, for example to test the interaction between products.

The verification templates² that must be filled during verification are generated automatically, starting from the quality criteria document and the mapping between criteria and products, in order to include only the criteria relevant to the specific product.

The OCCI interface that allows verifiers to instantiate their own machines, the improved contextualization scripts and the other improvements reduced the overhead of every single product verification.

2.3 Staged Rollout

As described at the beginning of the chapter during the Staged Rollout (SR) phase, the products are deployed as production services by volunteer site administrators. Some of the critical issues discovered during the PY3 were identified during staged rollout, since only some usage patterns were affected, and some of these could not be tested during verification. A small group of expert site administrators can participate to Staged Rollout without the risk that any issue can affect a bigger fraction of the infrastructure.

¹ <https://github.com/gwdg/rOCCI-server#rocci-server---a-ruby-occi-server>

² <https://documents.egi.eu/document/318>

The activity of the Early Adopters (EA) is mainly provided as a volunteer task, and the coordination is provided as an SA1 task, the role of the coordinator is mainly to assign the SR tasks to the EA teams, assess the status of the entries in the SR queue, to collect the staged rollout reports and to produce the release notes, starting from the information in the reports.

One important activity carried out by the SR coordination is the continuous assessment of the EA activity, in order to spot the products with missing EAs and to try and find new EA volunteers.

During PY3 the decommissioning of UMD-1 and the release of UMD-3 requested to re-assess the availability of EA for the new major release version of the products.

2.4 Building the release, the provisioning infrastructure

The EGI Software Repository infrastructure is composed by two main sections: the front end and the back end.

The front-end portal provides a unified point of access to the Universal Middleware Distribution (UMD), in terms of release announcements, release notes, repository configuration guides and the list of distributed products grouped by major distribution or capabilities.

The back-end is the framework that controls the import of products from the technology provider's repository, the advancement in the software provisioning workflows, the creation of the repositories. The back end is also interfaced with other tools used within the EGI activities such as the RT tracker and the EGI SSO.

In the previous years the product version structure was fixed to the three levels: x.y.z, this caused some in congruencies in the UMD releases as technology providers released products with a x.y.z.-w release format. In the back-end the version structure did not change, but during the PY3 the RT integration has been improved in order to make possible to assign a custom version to the product to be used in the UMD release notes. This allows the release team to publish in the release announcement exactly the same version as used by the technology providers.

UMD provides also mirroring facilities to distribute through the EGI repositories components needed by the EGI communities. During PY3 the EMI-WN tarball version produced by NGI_UK was made available for the whole of EGI, mirrored in the EGI repositories.

2.4.1 Producing and testing the release candidates

Once the products for the next UMD updates have been identified, a Release Candidate (RC) is created, that is a volatile test repository containing the candidate products for the UMD update.

Starting from PY3 the RCs are extensively tested for installability. This includes both the fresh installation and the update from a previous release of the products in the update, and the same tests for the other products, not updated in the RC, but already available in the UMD repositories.

This activity done manually would be very time consuming, and it is not compatible with the available effort for SA2 and neither with the timeline for a UMD release: a RC must be tested in few hours.

To automatize the testing of the RC, TSA2.3 implemented a script-based test suite (RC_tester³), which is available after each SA2 testbed virtual machine instantiation, and can be used to test production and test repositories.

RC tester deploys two different tests; it simulates an update and an installation from scratch for each UMD product (or meta-package). The check is not performed package per package, which would be too timing consuming, but using the metapackages that identify the products, and their dependencies install the other packages. The update test is performed only for the products released in the RC, starting from an installation from the UMD production repository, the scripts updates it to the version

³ https://wiki.egi.eu/wiki/EGI_Quality_Criteria_Verification#UMD_Release_Candidate_Testing

released in the RC (this emulates a site which is upgrading CREAM-CE metapackage for example to the latest version). This mechanism simplifies the testing process and simulates a real environment. If RC tester finds any issue (with any package) it generates an error log to identify which packages have dependencies issues. These logs are analysed by SA2.3 staff. The second test is a metapackage installation from scratch, just using the latest version from UMD testing repo. Again if the RC tester finds any issue with any package it raises an alarm and generates a log describing which package/s is/are wrong. This kind of testing takes some time, between 4 and 5 hours for each OS, depending on the number of products (meta-packages) that are available for testing. By default during RC testing SA2 verifies the entire metapackages list available from UMD, not only the new ones.

In summary, the RC_tester tasks are:

- Configure UMD/EMI/IGE/QCG repository keys.
- Detect any package inconsistency.
- Simulate UMD products installation.
- Check product update issues.
- Generate exit logs and it reports any failure to SA2 verifiers before UMD release.

The testing tool produces an extensive report that is then used to confirm the RC as production, or to fix the identified issues.

3 METRICS

3.1 Verification

The main metric defined to assess the efficiency of the verification process is the mean time spent to verify a product. The effort spent for verification is reported in the related RT tickets by the verifiers and statistics are automatically produced every day.

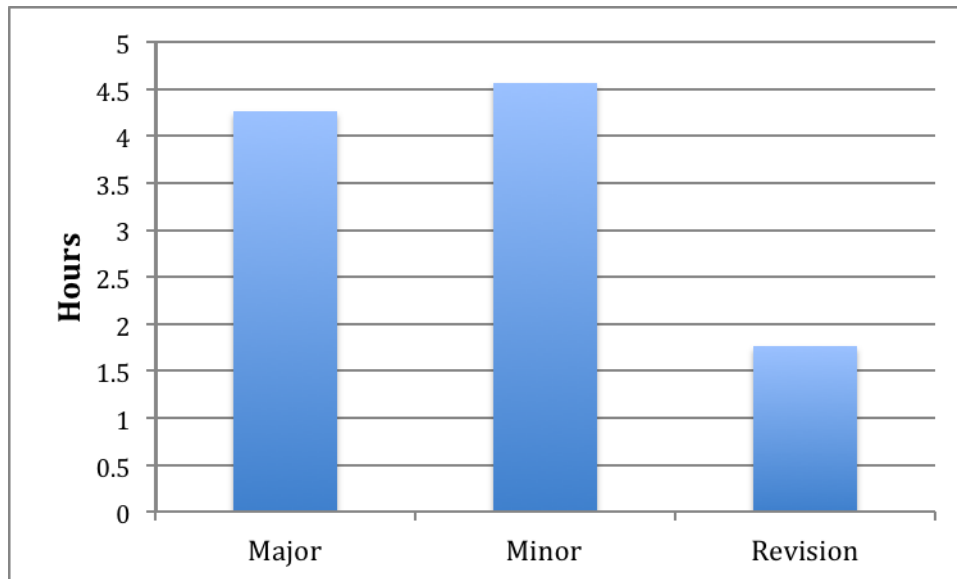


Figure 2, mean verification effort time in hours, per UMD release type

Figure 2 shows the mean verification time, per product, per UMD release type. The statistics included the releases during PY3 (one major release UMD2.0.0) plus the UMD3.0.0 major release and UMD 3.1.0. The overall trend is decreasing with major (84 products released) and minor releases (219 products updates released) having basically the same level of effort required, while the revision updates (30 product updates released) have in general a quick verification. Compared to PY2 the effort for Major releases improved drastically – from 19h per product to less than 4.5 hours – and Minor releases improved – from about 7 hours per product to the current 4.5 hours. During PY2 a total of 150 entries were processed by the software provisioning workflow, less than half during PY3 (350). The increase in the workload has been compensated by the reduction in the average verification time: from 14h to less than 5h per product⁴, it results in an overall small reduction of the effort necessary for verification. The reduction in the verification effort allowed verifiers to run additional tasks such as the installability test of the release candidates.

3.2 UMD package downloads statistics system

During PY2 the download data was loaded on MySQL to use a relational database to run the statistical computations. Due to the amount of the data the previous solution proved to be not scalable and we switched to use Hadoop⁵ and Hive⁶, which proved a lot more reliable and scalable.

⁴ The mean values, 14h for PY2 and 5h for PY3, are calculated including all types of releases.

⁵ <http://hadoop.apache.org/>

⁶ <http://hive.apache.org/>

The UMD package downloads statistics system is comprised of 3 sub-systems:

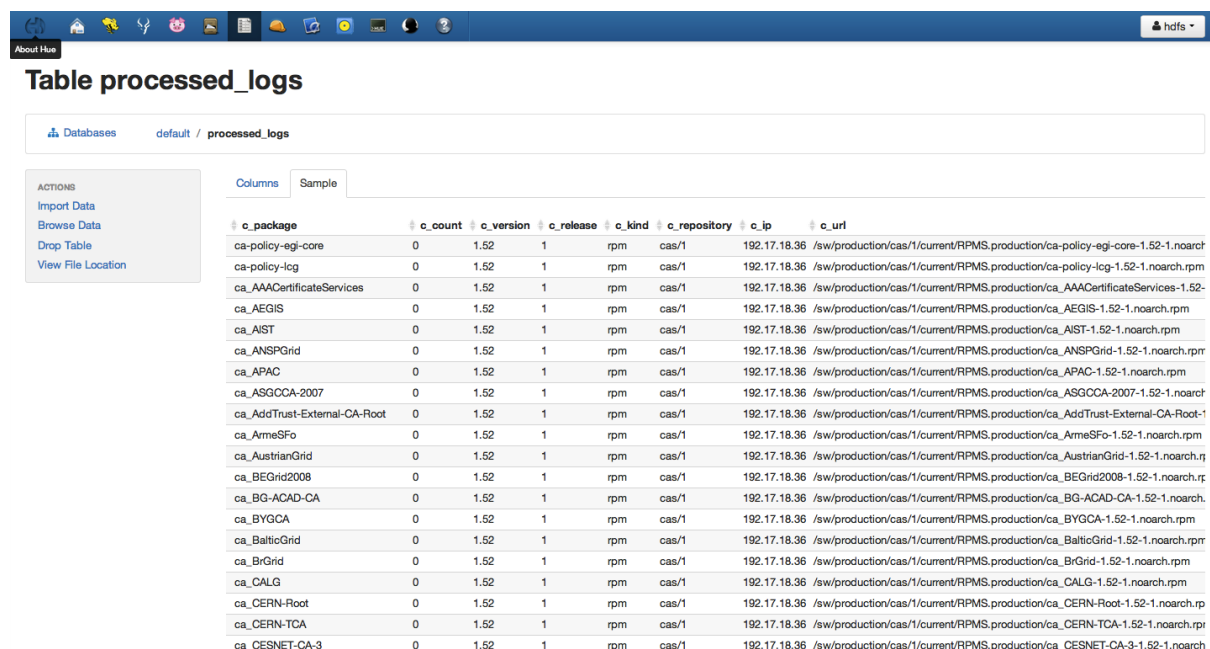
- The logging aggregation subsystem, which is based on a syslog-ng implementation. This subsystem collects centrally access logs from all web servers that serve the UMD repository.
- The log publishing subsystem. This is a workflow that uploads the logs from central system log facility to the Hadoop DFS.
- The user interface subsystem: Is based on a web application named Hue, this interface enables the user to processes the logs and download results as CVS or XLS files.

The logging aggregation subsystem is implemented according to the current best practices for central system logging. The central system logger has as destination one simple file for each day. This file will be processed from the publishing subsystem, later on.

The log publishing subsystem⁷ is an application written in Python that is parsing the aggregated logs through regular expressions, completing the missing geo data information (using GeoIP database and relevant information of country, city and ASN), archiving the logs by date in a compressed form and uploading the information in a structured form on Hadoop DFS.

The user interface consists of:

- One big table “processed_logs”: This table is being processed by a parallel SQL database, called Hive. There is no actual database binary file. The data are saved in plain text and each column is separated with a ‘\001’ character. Then, Hive has a metadata store that is matching each separated column with a column name for the convenience of the user.
- A query interface: Hue web application is enabling the user with a web editor, to write and edit SQL queries on the fly to analyze the data.
- The output interface: After the queries are finished, the user can collect the statistics from the Hue web application as a XLS or CVS file.



c_package	c_count	c_version	c_release	c_kind	c_repository	c_ip	c_url
ca-policy-egi-core	0	1.52	1	rpm	cas/1	192.17.18.36	/sw/production/cas/1/current/RPMS.production/ca-policy-egi-core-1.52-1.noarch.rpm
ca-policy-lcg	0	1.52	1	rpm	cas/1	192.17.18.36	/sw/production/cas/1/current/RPMS.production/ca-policy-lcg-1.52-1.noarch.rpm
ca_AAACertificateServices	0	1.52	1	rpm	cas/1	192.17.18.36	/sw/production/cas/1/current/RPMS.production/ca_AAACertificateServices-1.52-1.noarch.rpm
ca_AEGIS	0	1.52	1	rpm	cas/1	192.17.18.36	/sw/production/cas/1/current/RPMS.production/ca_AEGIS-1.52-1.noarch.rpm
ca_AIST	0	1.52	1	rpm	cas/1	192.17.18.36	/sw/production/cas/1/current/RPMS.production/ca_AIST-1.52-1.noarch.rpm
ca_ANSPGrid	0	1.52	1	rpm	cas/1	192.17.18.36	/sw/production/cas/1/current/RPMS.production/ca_ANSPGrid-1.52-1.noarch.rpm
ca_APAC	0	1.52	1	rpm	cas/1	192.17.18.36	/sw/production/cas/1/current/RPMS.production/ca_APAC-1.52-1.noarch.rpm
ca_ASGCCA-2007	0	1.52	1	rpm	cas/1	192.17.18.36	/sw/production/cas/1/current/RPMS.production/ca_ASGCCA-2007-1.52-1.noarch.rpm
ca_AddTrust-External-CA-Root	0	1.52	1	rpm	cas/1	192.17.18.36	/sw/production/cas/1/current/RPMS.production/ca_AddTrust-External-CA-Root-1.52-1.noarch.rpm
ca_ArmeSfo	0	1.52	1	rpm	cas/1	192.17.18.36	/sw/production/cas/1/current/RPMS.production/ca_ArmeSfo-1.52-1.noarch.rpm
ca_AustrianGrid	0	1.52	1	rpm	cas/1	192.17.18.36	/sw/production/cas/1/current/RPMS.production/ca_AustrianGrid-1.52-1.noarch.rpm
ca_BEGrid2008	0	1.52	1	rpm	cas/1	192.17.18.36	/sw/production/cas/1/current/RPMS.production/ca_BEGrid2008-1.52-1.noarch.rpm
ca_BG-ACAD-CA	0	1.52	1	rpm	cas/1	192.17.18.36	/sw/production/cas/1/current/RPMS.production/ca_BG-ACAD-CA-1.52-1.noarch.rpm
ca_BYGCA	0	1.52	1	rpm	cas/1	192.17.18.36	/sw/production/cas/1/current/RPMS.production/ca_BYGCA-1.52-1.noarch.rpm
ca_BalticGrid	0	1.52	1	rpm	cas/1	192.17.18.36	/sw/production/cas/1/current/RPMS.production/ca_BalticGrid-1.52-1.noarch.rpm
ca_BrGrid	0	1.52	1	rpm	cas/1	192.17.18.36	/sw/production/cas/1/current/RPMS.production/ca_BrGrid-1.52-1.noarch.rpm
ca_CALG	0	1.52	1	rpm	cas/1	192.17.18.36	/sw/production/cas/1/current/RPMS.production/ca_CALG-1.52-1.noarch.rpm
ca_CERN-Root	0	1.52	1	rpm	cas/1	192.17.18.36	/sw/production/cas/1/current/RPMS.production/ca_CERN-Root-1.52-1.noarch.rpm
ca_CERN-TCA	0	1.52	1	rpm	cas/1	192.17.18.36	/sw/production/cas/1/current/RPMS.production/ca_CERN-TCA-1.52-1.noarch.rpm
ca_CESNET-CA-3	0	1.52	1	rpm	cas/1	192.17.18.36	/sw/production/cas/1/current/RPMS.production/ca_CESNET-CA-3-1.52-1.noarch.rpm

Figure 3, screenshot of Query Editor

⁷ <https://github.com/auth-scc/Log-Transporter>

Table processed_logs

Databases default / processed_logs

ACTIONS
[Import Data](#)
[Browse Data](#)
[Drop Table](#)
[View File Location](#)

c_package	c_count	c_version	c_release	c_kind	c_repository	c_ip	c_url
ca-policy-egi-core	0	1.52	1	rpm	cas/1	192.17.18.36	/sw/production/cas/1/current/RPMS.production/ca-policy-egi-core-1.52-1.noarch.rpm
ca-policy-icg	0	1.52	1	rpm	cas/1	192.17.18.36	/sw/production/cas/1/current/RPMS.production/ca-policy-icg-1.52-1.noarch.rpm
ca_AAACertificateServices	0	1.52	1	rpm	cas/1	192.17.18.36	/sw/production/cas/1/current/RPMS.production/ca_AAACertificateServices-1.52-1.noarch.rpm
ca_AEGIS	0	1.52	1	rpm	cas/1	192.17.18.36	/sw/production/cas/1/current/RPMS.production/ca_AEGIS-1.52-1.noarch.rpm
ca_AIST	0	1.52	1	rpm	cas/1	192.17.18.36	/sw/production/cas/1/current/RPMS.production/ca_AIST-1.52-1.noarch.rpm
ca_ANSPGrid	0	1.52	1	rpm	cas/1	192.17.18.36	/sw/production/cas/1/current/RPMS.production/ca_ANSPGrid-1.52-1.noarch.rpm
ca_APAC	0	1.52	1	rpm	cas/1	192.17.18.36	/sw/production/cas/1/current/RPMS.production/ca_APAC-1.52-1.noarch.rpm
ca_ASGCCA-2007	0	1.52	1	rpm	cas/1	192.17.18.36	/sw/production/cas/1/current/RPMS.production/ca_ASGCCA-2007-1.52-1.noarch.rpm
ca_AddTrust-External-CA-Root	0	1.52	1	rpm	cas/1	192.17.18.36	/sw/production/cas/1/current/RPMS.production/ca_AddTrust-External-CA-Root-1.52-1.noarch.rpm
ca_ArmeSFo	0	1.52	1	rpm	cas/1	192.17.18.36	/sw/production/cas/1/current/RPMS.production/ca_ArmeSFo-1.52-1.noarch.rpm
ca_AustrianGrid	0	1.52	1	rpm	cas/1	192.17.18.36	/sw/production/cas/1/current/RPMS.production/ca_AustrianGrid-1.52-1.noarch.rpm
ca_BEGrid2008	0	1.52	1	rpm	cas/1	192.17.18.36	/sw/production/cas/1/current/RPMS.production/ca_BEGrid2008-1.52-1.noarch.rpm
ca_BG-ACAD-CA	0	1.52	1	rpm	cas/1	192.17.18.36	/sw/production/cas/1/current/RPMS.production/ca_BG-ACAD-CA-1.52-1.noarch.rpm
ca_BYGCA	0	1.52	1	rpm	cas/1	192.17.18.36	/sw/production/cas/1/current/RPMS.production/ca_BYGCA-1.52-1.noarch.rpm
ca_BalticGrid	0	1.52	1	rpm	cas/1	192.17.18.36	/sw/production/cas/1/current/RPMS.production/ca_BalticGrid-1.52-1.noarch.rpm
ca_BrGrid	0	1.52	1	rpm	cas/1	192.17.18.36	/sw/production/cas/1/current/RPMS.production/ca_BrGrid-1.52-1.noarch.rpm
ca_CALG	0	1.52	1	rpm	cas/1	192.17.18.36	/sw/production/cas/1/current/RPMS.production/ca_CALG-1.52-1.noarch.rpm
ca_CERN-Root	0	1.52	1	rpm	cas/1	192.17.18.36	/sw/production/cas/1/current/RPMS.production/ca_CERN-Root-1.52-1.noarch.rpm
ca_CERN-TCA	0	1.52	1	rpm	cas/1	192.17.18.36	/sw/production/cas/1/current/RPMS.production/ca_CERN-TCA-1.52-1.noarch.rpm
ca_CESNET-CA-3	0	1.52	1	rpm	cas/1	192.17.18.36	/sw/production/cas/1/current/RPMS.production/ca_CESNET-CA-3-1.52-1.noarch.rpm

Figure 4, screenshot of Data in Hive

4 UMD RELEASE TEAM

The UMD Release Team (URT) is a new lightweight coordination activity run by EGI SA2 to fill some gaps left after the end of the European middleware projects – European Middleware Initiative (EMI) and Initiative for Globus in Europe (IGE). This activity started at the end of PY3. The main purpose of URT is to keep open the communication channels between the product teams, as they existed previously within the middleware projects, and open new lines of communications between EGI and the product teams, as before they were mediated by the middleware projects technical coordination.

The URT provides coordination facilities for releases of software intended to be deployed on the EGI production infrastructure. To support this goal it will:

- Monitor and follow up issues and topics recorded and managed through the EGI Help Desk; this includes incidents & problem reports, requests for information, and feature requests written in the scope of single, identifiable components.
- Monitor and follow up issues recorded and managed by the EGI Security Vulnerability Group (SVG) and its associated Risk Assessment Team (RAT) [MS414]
- Coordinate release schedules across associated Technology Providers such that dependencies between Platforms and Products (where required) are respected.
- Monitor the development plans of the Technology Providers.
- Coordinate software provisioning, whether direct EGI Software Provisioning or delegated to Technology Providers with the common goal of making platforms and components available in the EGI Software Repository (and its major component, the UMD).
- To assess Technology Provider software provisioning processes and documents, including Quality Assurance.
- Serve as escalation point and board of arbitration for disputes and conflicts pertaining to issues recorded in the Help Desk and vulnerability trackers managed by the SVG and RAT.

The membership of the group is kept as much open as possible, trying to include all the product teams without the need of a formal agreement between the PT and EGI.

The membership consists of:

- SA2.1 representative (chair)
- Technology Provider Release Managers (including deputies) for Community Platforms
- Product teams representatives
- Deployed Middleare Support Unit⁸ representative, when required
- Representatives from the UMD software provisioning tasks (repository provisioning, verification and staged rollout), when required

URT meets every two weeks in a phone meeting, agendas are available in the EGI wiki and can be contributed directly by the product teams⁹.

⁸ DMSU wiki: https://wiki.egi.eu/wiki/EGI_DMSU

⁹ https://wiki.egi.eu/wiki/URT_meetings_agendas

5 COMMUNITY SOFTWARE REPOSITORY

One of the most important features implemented in PY3 in cooperation with TNA2.5 is the community repository offered as an extension to the EGI AppDB service¹⁰ with support for EGI Community Technology Providers - a specific type of grid/cloud software developers who will become part of EGI come May 2013 - effectively turning the AppDB into an entry point for the community software repository. Beta support was available on the development instance and presented during the EGI Community Forum, after which it was released into production with v4.2.0 on the 22nd of April 2013. The purpose of this tool is to provide an alternative level of engagement to the product teams who want to make available their products for EGI, but who do not want to, or cannot, engage with formal agreements. EGI, with the AppDB, provides a repository as a service tool, where authorized users can handle the releases of their software in a dedicated repository, hosted by EGI. This means that the teams will not be covered by MoUs or SLAs, but they will simply upload software packages they produce into the EGI Community Repository. AppDB has been chosen to act as the platform to connect the Community Product Teams with Community Platform integrators.⁴

With the aforementioned extension, the Applications Database service supports the sharing of software artifacts within and beyond EGI, as it now

- Offers the ability to manage and publish unlimited series of releases, per registered software item
- Supports a light-weight & community driven, release management process
- Populates software metadata required by administrators as well as typical developers
- Acts as a communication medium for contacting the software developers
- Disseminates new release through established communication channels (via RSS, emails, gadget instances, followers, etc.)

From a technical standpoint the AppDB provides support for:

- generic tarballs, RPM & DEB (32bit/64bit) binaries
- multiple flavor / operating system combinations
- Simplified, web-based process for uploading binary artifacts
- YUM & APT repositories for automatic software deployment
- Mechanisms for initiating, updating, removing, renaming, publishing, and even cloning releases and their associated repositories
- Flexible, user-defined, versioning schema
- Capable of creating:
 - candidate repositories per release, or
 - production-grade incremental repositories
- Ability to perform actions at the package level, such as:
 - individual package addition & removal
 - definition of special attributes (e.g. meta-packages)
 - automatic extraction of package (RPM or DEB) meta-information

¹⁰ <http://repository.egi.eu/2013/04/02/community-software/>



The service can manage both candidate and production-grade repositories of software releases, and these releases are accessible through the community branch of the EGI Software repository infrastructure.

The community repositories from a technical standpoint are part of the UMD repository infrastructure, which provides high availability and load balancing configurations.

In summary, differently from the UMD releases, the community repositories do not add any quality assurance nor coordination on top of the releasing activities of the product teams, it is – as specified above- a repository as a service tool and its target is to facilitate the distribution of non-UMD software among the EGI communities.

6 PLANS FOR THE FUTURE

The technological ecosystem around the European grid infrastructure has been fairly stable over the last three years, supported by the European funded middleware projects, but it changed at the end of PY3 with the end of EMI and IGE.

The previous situation, until the end of PY3, saw two main technology providers, coordinating the work of multiple product teams and producing coordinated releases. Technology providers were also acting as proxies for the communications between EGI and the product teams.

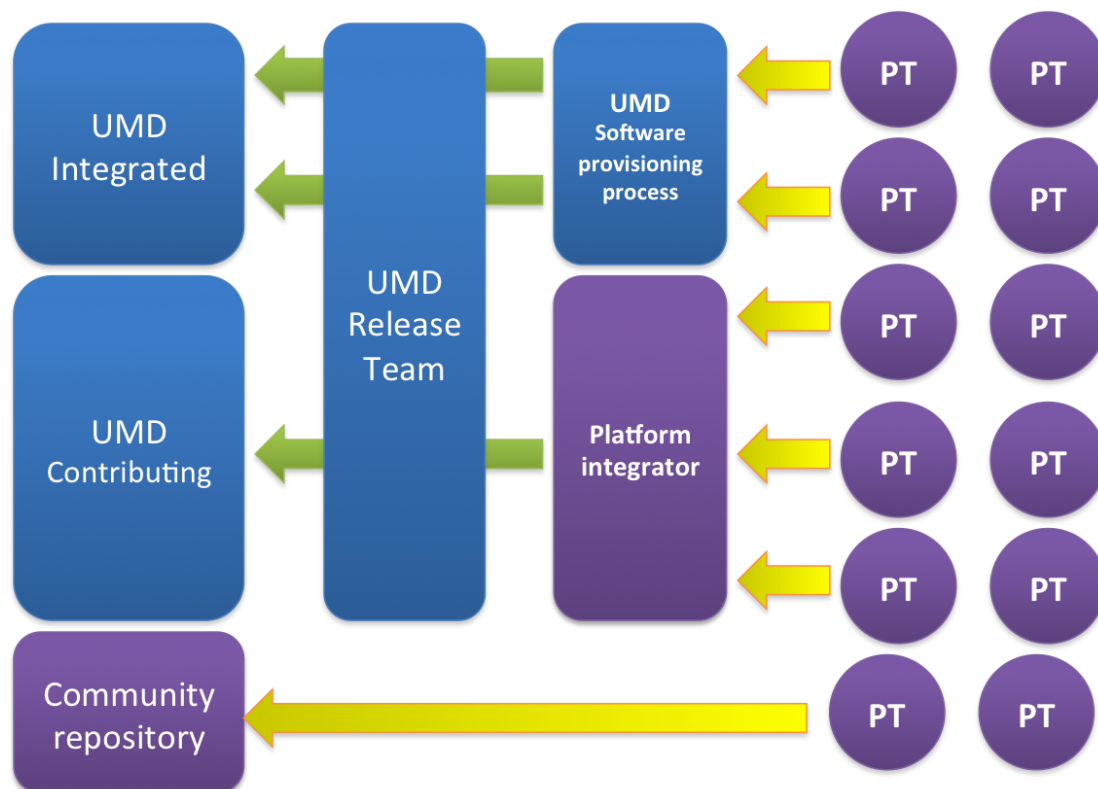


Figure 5, the new technology ecosystem structure. The blue boxes are the elements of the workflow that will continue to be coordinated by EGI. The yellow arrows are the products released by the product team, but not yet processed by the UMD quality assurance.

Figure 5 shows the current ecosystem: a bigger number of technology providers (the product teams) less coordinated, who continue to produce software releases and to contribute to UMD.

Some of these product teams – the integrated ones - will be released in UMD with a workflow as in the past. The products released as integrated in UMD are the critical services needed to run the infrastructure. The components part of the Core Platform (see [R3]), for example, will undergo this process.

The second set of products, probably the largest one, will have an equivalent quality assurance, as the software verified by SA2, but with verification outsourced to the developers or the communities depending on the middleware. The final step, the release of these products in UMD updates, will continue to be coordinated by EGI.

The third set of product teams will distribute their software among the EGI sites using the Community Repository facility - as described in Chapter 5- without any additional quality assurance process added on top of the internal product team certification.

The URT will collect and disseminate the information about the updates planned by the product teams in order to minimize any conflict or problem associated with products depending to each other, and facilitate – where necessary – the synchronization with the UMD releases. URT is not coordinating the technical work or the release plans of the product teams, but only assuring that the information are correctly circulated and that any issue related to cross-product dependencies is not affecting the UMD releases.

The software provisioning process is still in a transition phase, and most of these changes will be implemented in the next 6 months.

The software provisioning infrastructure currently supports releases distributed in the EMI, IGE and the QCG repositories. The infrastructure will need to be extended to include more repositories as the product team will change their target repository for their releases.

One of the first repositories to be explored for the integration is EPEL (Extra Packages for Enterprise Linux) as already many product teams are planning to use it as primary repository: e.g. CERN information system, CERN data management and ARC stack. This integration may bring advantages to both UMD and the product teams: starting the UMD software provisioning process when the updates are still in the EPEL testing repository (before the official release in EPEL stable) allows to provide quick feedback to the product team developers about any problem found during staged rollout or verification, and reduces the time between the product team release and the UMD release of the same update.

SA2 will push the usage of AppDB also as an entry point for UMD; in this way many product teams using different repositories will be able to submit their updates to UMD in an uniform way.



7 CONCLUSIONS

The process proved to be efficient and flexible during PY2 and PY3, the changes introduced were not substantial but important to improve the efficiency of the workflow and improvements have been implemented almost in all the steps of the workflow. The new criteria test for specific operational issues and new capabilities introduced. The infrastructure has been extended at different levels: the support infrastructure now supports multiple o.s. and major releases of UMD and the testbed is easier to use for the verifiers, who can be more autonomous in their work.

The changes planned for PY4 focus on implementing a more sustainable and scalable software provisioning in order to maintain the quality of the software in UMD, in the current scenario of many product teams working independently. URT and the community repository are the first solutions implemented to work towards this target.



8 REFERENCES

MS508	MS508 Software provisioning process https://documents.egi.eu/public/ShowDocument?docid=505
MS512	MS512 Software provisioning process https://documents.egi.eu/document/1135
D59	D5.9 Annual report on the software provisioning activity https://documents.egi.eu/document/1657
MS514	MS514 EGI Platform Roadmap https://documents.egi.eu/document/1624