EUROPEAN MIDDLEWARE INITIATIVE

DSA1.2 - SOFTWARE RELEASE PLAN

EU DELIVERABLE: D3.2

Document identifier:	**EMI-DSA1.2-CDSREFSoftware_Release_Plan-v0.14.doc**
Date:	**07/01/2010**
Activity:	**SA1 – Maintenance and Support**
Lead Partner:	**INFN**
Document status:	**Draft**
Document link:	

Abstract:
This document describes the EMI release management procedures and plans, and the initial release schedules prepared in collaboration with the PTB and the JRA1 Work Package. The release schedule will be updated every three months during the course of the project.

Copyright notice:

Copyright (c) Members of the EMI Collaboration. 2010.

See http://www.eu-emi.eu/about/Partners/ for details on the copyright holders.

EMI ("European Middleware Initiative") is a project partially funded by the European Commission. For more information on the project, its partners and contributors please see http://www.eu-emi.eu.

This document is released under the Open Access license. You are permitted to copy and distribute verbatim copies of this document containing this copyright notice, but modifying this document is not allowed. You are permitted to copy this document in whole or in part into other documents if you attach the following reference to the copied elements: "Copyright (C) 2010. Members of the EMI Collaboration. http://www.eu-emi.eu ".

The information contained in this document represents the views of EMI as of the date they are published. EMI does not guarantee that any information contained herein is error-free, or up to date.

EMI MAKES NO WARRANTIES, EXPRESS, IMPLIED, OR STATUTORY, BY PUBLISHING THIS DOCUMENT.

Delivery Slip

	Name	Partner / Activity	Date	Signature
From				
Reviewed by				
Approved by				

Document Log

Issue	Date	Comment	Author / Partner
1			
2	0.14	Modified Schedules Dates	C. A.
3			

Document Change Record

Issue	Item	Reason for Change
1		
2		
3		

TABLE OF CONTENT

1. INTRODUCTION

1.1. PURPOSE

This document describes the release procedures and policies for the EMI software components and the initial releases plans and schedules

1.2. DOCUMENT ORGANISATION

The document is organized in six chapters as follows:

Chapter 1: **Introduction**: This section explains the purpose, scope and organization of the document

Chapter 2: **Executive Summary**: This section contains a high-level description of the document. It gives a summary of the most important points described in each main section.

Chapter 3: **Summary of the initial situation:** This section presents a short overview of the release processes of the four middleware providers, ARC, dCache, gLite, UNICORE, constituting the basis on which the EMI Release process will be developed.

Chapter 4: **EMI Release Process:** This section describes the EMI release management process starting with the description of the roles and responsibilities, decision making bodies, EMI release categories, release criteria, EMI repositories and the technical management of releases. Last sections are on the release process, scheduled and emergency, and initial release schedules.

Chapter 5: **Internal Release EMI-0:** This section describes the first integration test release, the first test build of EMI components configurations in ETICS and the first release schedule attempt

Chapter 7: **Conclusions**: This section provides information on further work.

1.3. REFERENCES

R1	ARC Release Process http://wiki.nordugrid.org/index.php/Release_management
R2	ARC WIKI http://wiki.nordugrid.org/
R3	dCache SVN http://svn.dcache.org/dCache/
R4	dCache RT http://rt.dcache.org/Ticket/Display.htm
R5	dCache WIKI http://trac.dcache.org/projects/dcache/wiki
R6	dCache build and test system http://svn.dcache.org/build/
R7	Definition and documentation of the gLite revised software lifecycle process https://edms.cern.ch/document/973115
R8	gLite CVS http://cvs.web.cern.ch/cvs/
R9	gLite savannah https://savannah.cern.ch/bugs/?group=jra1mdw
R10	gLite WIKI https://twiki.cern.ch/twiki/bin/view/EGEE/EGEEgLite

R11	gLite EMT meetings https://twiki.cern.ch/twiki/bin/view/EGEE/ReleaseManager#EMT_agendas
R12	GGUS https://gus.fzk.de/pages/home.php
R13	ETICS https://twiki.cern.ch/twiki/bin/view/ETICS/WebHome
R14	EGEE-III gLite Product Team Release Process https://edms.cern.ch/document/1041827/1
R15	gLite Release Process https://twiki.cern.ch/twiki/bin/view/EGEE/ProdInt
R16	UNICORE SVN http://unicore.svn.sourceforge.net/viewvc/unicore
R17	UNICORE Bug tracking system http://sourceforge.net/tracker/?group_id=102081
R18	UNICORE WIKI http://sourceforge.net/apps/mediawiki/unicore/index.php?title=Main_Page
R19	UNICORE build and test system http://unicore-dev.zam.kfa-juelich.de/bamboo/start.action;jsessionid=1ththaa1402c1
R20	EMI Build and Configuration Guidelines https://twiki.cern.ch/twiki/pub/EMI/EmiSa2ConfigurationIntegrationGuidelines/EMI_EmiSa2ConfigurationIntegrationGuidelines_v_1_0.pdf
R21	EMI Production Release Criteria https://twiki.cern.ch/twiki/bin/view/EMI/ProductionReleaseCriteria
R22	EMI Change Management Guidelines https://twiki.cern.ch/twiki/pub/EMI/EmiSa2ChangeManagementGuidelines/EMI_SA2_Change_v_1_0.pdf
R23	Fedora Packaging Guidelines http://fedoraproject.org/wiki/PackagingGuidelines
R24	Debian Policy Manual http://www.debian.org/doc/debian-policy/
R25	EMI Certification and Testing Guidelines https://twiki.cern.ch/twiki/pub/EMI/EmiSa2CertTestGuidelines/EMI_SA2_CertificationAndTesting_v_1_0.pdf
R26	EMI Software Quality Assurance Plan https://twiki.cern.ch/twiki/pub/EMI/DeliverableDSA21/EMI-DSA2.1-1277599-QA_Plan-v1.2.pdf
R27	EMI Metrics Guidelines https://twiki.cern.ch/twiki/pub/EMI/EmiSa2MetricsGuidelines/EMI_SA2_Metrics_v_1_0.pdf
R28	EGI Software Provisioning Process https://documents.egi.eu/public/RetrieveFile?docid=68&version=7&filename=EGI-MS503-final.pdf

R29	EGI Quality Criteria https://documents.egi.eu/public/ShowDocument?docid=240
R30	EMI Release Schedule tracker https://savannah.cern.ch/projects/emi-rel-sched/
R31	Compute Area Work Plan https://twiki.cern.ch/twiki/pub/EMI/DeliverableDJRA111/EMI-DJRA1.1.1-1277608-Compute_Area_Work_Plan-v1.0.pdf
R32	Data Area Work Plan https://twiki.cern.ch/twiki/pub/EMI/DeliverableDJRA121/EMI_DJRA1.2.1-1277615-Data_Area_Work_Plan_v1.0.pdf
R33	Infrastructure Area Work Plan https://twiki.cern.ch/twiki/bin/view/EMI/DeliverableDJRA141
R34	Security Area Work Plan https://twiki.cern.ch/twiki/pub/EMI/DeliverableDJRA131/EMI-DJRA1.3.1-1277566-Security_Area_Work_Plan-v1.0.pdf
R35	EMI User Requirement Tracker https://savannah.cern.ch/projects/emi-req/
R36	EMI Technical Objectives Tracker https://savannah.cern.ch/task/?group=emi-tech
R37	EMI Technical Development Plan https://twiki.cern.ch/twiki/pub/EMI/DeliverableDNA131/EMI-DNA1.3.1-1277540-Technical_Plan-v1.0.pdf
R38	EVO – the Collaboration Network http://evo.caltech.edu/evoGate/
R39	EMT indico agendas http://indico.cern.ch/categoryDisplay.py?categId=3077
R40	EMT activity twiki https://twiki.cern.ch/twiki/bin/view/EMI/EMT
R41	EMI 1 Development and Test Plan https://twiki.cern.ch/twiki/bin/view/EMI/InternalDeliverableEmi1ReleaseDevPlans
R42	EMI Integration Testbed https://twiki.cern.ch/twiki/bin/view/EMI/TestBed#NagiosTestbed
R43	EMI Software Repository http://emisoft.web.cern.ch/emisoft/dist/EMI/
R44	EMI Releases Documentation http://www.eu-emi.eu/en/downloads
R45	Deploying Software into the EGI production infrastructure https://documents.egi.eu/secure/ShowDocument?docid=53
R46	EMI Technical Objectives https://twiki.cern.ch/twiki/bin/view/EMI/EmiJra1T1EMITechnicalObjectives
R47	LCG-ROLLOUT mailing list https://www.jiscmail.ac.uk/cgi-bin/webadmin?A0=LCG-ROLLOUT
R48	ETICS EMI-0 nightly-builds

	http://etics-repository.cern.ch/repository/reports/name/emi_R_0/-/reports/index.html
R49	EMI EMT Coordination – action list https://savannah.cern.ch/task/?group=emi-emt
R50	NorduGrid source repository http://svn.nordugrid.org/
R51	RPM list for the custom "sl5_x86_64_gcc412EPEL" ETICS platform http://eticssoft.web.cern.ch/eticssoft/internal/public/VMWareImages/EMI_SL5_x86_64_EPEL_rpmlist.txt

1.4. DOCUMENT AMENDMENT PROCEDURE

This document can be amended by the authors further to any feedback from other teams or people. Minor changes, such as spelling corrections, content formatting or minor text reorganization not affecting the content and meaning of the document can be applied by the authors without peer review. Other changes must be submitted to peer review and to the EMI PEB for approval.

When the document is modified for any reason, its version number shall be incremented accordingly. The document version number shall follow the standard EMI conventions for document versioning. The document shall be maintained in the CERN CDS repository and be made accessible through the OpenAIRE portal.

1.5. TERMINOLOGY

API	Application Programming Interface
APT	Advanced Packaging Tool
ARC	The **Advanced Resource Connector** is general purpose, Open Source, lightweight, portable middleware solution (http://www.knowarc.eu/middleware.html)
BDII	Berkley Database Information Index
CR	Component Release
dCache	System for storing and retrieving huge amounts of data, distributed among a large number of heterogeneous server nodes, under a single virtual filesystem tree with a variety of standard access methods (http://www.dcache.org/)
DCI	Distributed Computing Infrastructure
EGEE	Enabling Grids for E-sciencE (http://www.eu-egee.org/)
EGI	European Grid Infrastructure – http://www.egi.eu
EGI-InSPIRE	Integrated Sustainable Pan-European Infrastructure for Researchers in Europe (http://www.egi.eu/projects/egi-inspire/)
EMI	European Middleware Initiative – http://www.eu-emi.eu
EMT	Engineering Management Team
GA	General Availability Release

GGUS	Global Grid User Support (https://gus.fzk.de)
gLite	The next generation middleware for grid computing born from the collaborative efforts of more than 80 people in 12 different academic and industrial research centers as part of the EGEE Project (http://glite.web.cern.ch/glite/)
PEB	Project Executive Board
PRACE	Partnership for Advanced Computing in Europe, a unique persistent pan-European Research Infrastructure for High Performance Computing (HPC) - http://www.prace-project.eu/
PT	Product Team
PTB	Project Technical Board
QA	Quality Assurance
QC	Quality Control
RT	Request Tracker
SA1	Maintenance and Support Work Package
SA2	Quality Assurance Work Package
SQA	Software Quality Assurance
SQAP	Software Quality Assurance Plan
SVN	Subversion, version control system
TAC	Technical Area Coordinator
TD	Technical Director
UMD	Unified Middleware Distribution, the EGI middleware distribution
UNICORE	The Uniform Interface to Computing Resources offers a ready-to-run Grid system including client and server software. UNICORE makes distributed computing and data resources available in a seamless and secure way in intranets and the internet (http://www.unicore.eu/)
WP	Work Package
YUM	Yellowdog Updater Modified

2. EXECUTIVE SUMMARY

The European Middleware Initiative (EMI) is a close collaboration of four major European middleware providers, ARC, gLite, UNICORE and dCache. Its main objectives are to deliver a consolidated set of middleware components for deployment in EGI (as part of the Unified Middleware Distribution or UMD), PRACE and other DCIs, extend the interoperability and integration between grids and other computing infrastructures, strengthen the reliability and manageability of the services and establish a sustainable model to support, harmonise and evolve the middleware, ensuring it responds effectively to the requirements of the scientific communities relying on it.

During the first year in parallel with the preparation of the first EMI release, the release processes of the four middlewares stacks is maintained in order to ensure a smooth transition from many middleware distributions to one, so that the production infrastructures stay functional without noticeable discontinuity. The document begins with a short overview of these processes.

The release management task is responsible to coordinate and maintain the package repositories, defining policies and release cycles, to ensure customers receive certified software releases of middleware services and components according to agreed release policies and quality of service attributes.

The objective of release planning is to complete a release plan that will identify:

- release objectives;
- roles, responsibilities;
- tasks, activities;
- communication and
- release schedule

All these objectives are presented in detail in this document.

The Release Plan is established by the Maintenance and Support Work Package (SA1) with the details of the timelines to be applied to each release cycle (e.g. code freeze date, release date, any technical preview release date) and the outline of the set of acceptance criteria to be fulfilled by the components (like documentation and specific categories of tests)

During the three year duration of the EMI project, the EMI developers and engineers will work together to consolidate, harmonize and support the existing software products, evolving and extending them based on existing and new requirements.

Once a year, the project will produce a major release of the EMI distribution, characterized by a well-defined interface and behaviour for each of its components. Interface and behaviour are allowed to change within a major release of the distribution only in a backwards-compatible way. Component Releases are classified in major, minor, revision and emergency, based on the impact of the changes on the component interface and behavior.

The five-steps yearly cycle of the development and maintenance activities is presented in Section 4. After the presentation of the roles and responsibilities, as well as the decision bodies, that will play an important role in the definition of each release objectives, the release management is detailed, with an emphasize on the Components Releases tracking and different states of the Components Releases tasks until its deployment.

The section also contains an overview of the software components build process and the management of the external dependencies, and details on the structure of the EMI repositories.

Scheduled and emergency release procedures are explained and the initial schedules for the three major releases are presented.

3. SUMMARY OF INITIAL SITUATION

This section presents a short overview of the release processes of the four middleware providers, ARC, dCache, gLite, UNICORE, constituting the basis on which the EMI release process will be developed.

3.1. ARC RELEASE PROCESS

The **Advanced Resource Connector (ARC)** middleware, developed by the NorduGrid collaboration, is a software solution that uses Grid technologies to enable sharing and federation of computing and storage resources distributed across different administrative and application domains. ARC is used to create Grid infrastructures of various scope and complexity, from campus to national Grids [R1].

3.1.1 Background

An ***ARC release*** is defined as a set of source packages and the corresponding binaries packages on the supported platforms. The source release is defined by an SVN tag [R50]. *ARC releases* have been created with the strong emphasis on multiplatform support, Linux distribution independence and recently also extended OS independence (Linux, Windows, Mac, Solaris), non-intrusiveness and ease of operation

3.1.2 Development Environment

- Code is kept in SVN spread over several SVN trees [R50];
- Bugzilla is used to report software defects and to act upon them and to report feature requests. They are periodically reviewed and prioritized;
- WIKI [R2] is used as a workarea to collect development plans, technical drafts, release coordination, configuration templates, quick guides, HOWTOs;
- The nordugrid-discuss mailing-list is used for technical discussions;
- An issue tracking system is used to provide user support;
- Currently there are two production systems in use to perform automatic nightly builds of several SVN areas and to produce binary packages for tagged releases.

3.1.3 Release Categories

Major release needs longer term planning; 3-6 months of preparation; alpha, beta, rcx test releases; introduces new components, features; obsoletes components; may break backward compatibility.

Minor release is a planned bug-fix release that needs maximum one month preparation; no new features; no new components; only fixes. It's a scheduled release.

Emergency release represents an unplanned urgent release to fix a security issue or a critical bug; needs maximum two weeks preparation.

Binary update of a release: Sometimes, due to a change in the external dependencies ARC releases have to be rebuilt. This does not affect the ARC source code, no re-taging takes place, only new binaries are build and it can happen for both major, minor or emergency releases.

3.1.4 Workflow

Scheduled release procedure is characterized by:

- Major or minor releases follow a scheduled release procedure;

- Planning phase consists of prioritizing open bugs to be fixed; collecting requests for new features (only for major release);
- Decision phase takes into account the status (maturity) of release candidate components. The release content is defined in terms of components and bugs to be fixed. The supported platforms are decided;
- Preparation of the release consists of creating the release branch, updating the documentation and release notes, and creating the release candidate(s) from the branch;
- Testing phase: consists of deploying the Release Candidate on the test infrastructure where the test suits are executed;
- Release is ready if all the critical bugs recorded in Bugzilla are closed; functionality decided in planning phase is successfully tested on the release candidate; binary packages are available for all supported platforms and documentation reflecting changes from the previous release is in place.

Emergency Release Procedure is characterized by**:**

- It is applied when running into a major issue like a software defect considered critical or a security hole;
- Reporting is done through Bugzilla, mailing list or direct contact to core coordination group members;
- The process cannot take more than two weeks.

3.2. DCACHE RELEASE PROCESS

dCache is a system for storing and retrieving huge amounts of data, distributed among a large number of heterogeneous server nodes, under a single virtual filesystem tree with a variety of standard access methods. Depending on the persistency model, dCache provides methods for exchanging data with backend (tertiary) storage systems as well as space management, pool attraction, dataset replication, hot spot determination and recovery from disk or node failures.

3.2.1 Background

dCache is moving away from feature based releases towards Time Based Releases. As the name already implies, this essentially means that it will be better in predicting when a release is being published. The drawback of this approach is that features which don't make it into the repository till some time before the release is due, will not be part of that release. The advantage is that sites can do better planning in upgrading and that the synchronization with distributors is eased.

3.2.2 Development Environment

- For version control code is kept in SVN [R3];
- RT is used to keep track of software defects and to act upon them; feature requests are also collected in RT bugs [R4];
- WIKI [R5] is used as a workarea: to collect development plans, technical meetings, release coordination and documentation links;
- Currently the Hudson build and test system [R6] is used to perform automatic and on-demand builds and to produce binary (i.e. rpm) and source (i.e. tgz, pkg) packages for tagged releases.

3.2.3 Release Categories

dCache follows a two dimensional release system:

- **Recommended Releases:** At some point in time, dCache.org recommends a particular release that should be used on production systems. Other releases are considered stable, however care should be taken as such releases have only received limited amount of public testing or are outdated.
- **Golden Release:** One release is declared a "golden release" that means it is supported for a long period, in the order of a year or more. For golden releases there are strict rules concerning the patches which are allowed (e.g. no new features).

Every couple of months it is produced a feature release. Feature releases have version numbers 1.9.x-1, where x is a positive integer. Each feature release is followed up by a number of maintenance releases containing bug fixes or high priority feature tweaks. Maintenance releases for release 1.9.x-1 have version numbers 1.9.x-y, for y being an integer larger than 1.

3.2.4 Workflow

Two weeks before the release is declared to be cut, the repository is closed and no check-in is allowed anymore. At the same time, testing starts.

Once a developer has done unit testing the code will be submitted to a Review Board. Here the other developers can and should have a look at this code. Code has to be reviewed by at least one of the other developers and can be committed to the trunk branch as soon as all the developers who reviewed it mark it to be shipped. When code is committed, commit hooks automatically build dCache, dcap and srm client. On error an email is sent to the developer who made the last commit. With the last successful build the yum repositories are updated and the rpms automatically installed on some machines.

If the code is intended to go into the production branch, the developer has to send a merge request to the Release Manager. Only the Release Manager can submit code to stable branches. A stable branch is any SVN branch from which releases are made.

3.3. GLITE RELEASE PROCESS

gLite is the next generation middleware for grid computing. Born from the collaborative efforts of more than 80 people in 12 different academic and industrial research centers as part of the EGEE Project, gLite provides a framework for building grid applications tapping into the power of distributed computing and storage resources across the Internet.

3.3.1 Background

gLite uses a *'continuous release process*' that was defined and optimized during EGEE-II and EGEE-III projects. The guiding principle of this approach is to manage independent updates in isolation from one another, allowing each to progress through the full software life-cycle without interference. This has proven to bring numerous benefits, particularly in accelerating the adoption of important changes which can be properly prioritized and which are not held back by other unrelated updates. In this model there is a baseline release to which updates are continuously added [R7]

3.3.2 Development Environment

- Code is kept in CVS [R8];
- Savannah [R9] is used to report software defects, to act upon them, and as release tracking tool;

- WIKI [R10] is used as a workarea, to collect development plans, technical drafts, release coordination, configuration templates, quick guides, HOWTOs;
- The project-eu-egee-midlleware-emt mailing-list is used for developers discussions and announcements. Weekly EMT [R11] telephone conferences are held for discussions on gLite Release Schedule and to assess bug priorities;
- The GGUS ticketing system is used to provide user support [R12];
- The ETICS system is used to manage every aspect of the software development workflow, i.e. build, release and testing [R13];

3.3.3 Release Categories

gLite follows a two dimensional release system.

- **Internal Release** is an update to individual, certified components that are made available to other Product Teams (PTs). Instead of being released to production, it will instead serve as input to production releases.
- **Production Release** is an update to a self-contained product, typically node-type based, which is deployed onto the production infrastructure. The product will typically contain a number of components that have been released internally by other PTs; however, it is the product team releasing the complete product to production that is ultimately responsible and accountable for that release.

3.3.4 Workflow

An overall description of the new gLite release process can be found in the EGEE-III gLite Product Team Release Process [R14] document. The process is based on product teams being entirely responsible for their products (meaning: development, testing, integration and release). The Product Team Release Process [R15] consists on the following steps:

- creating patches and metapackages;
- certification and tests;
- verification;
- staged roll-out;
- production deployment.

3.4. UNICORE RELEASE PROCESS

UNICORE (Uniform Interface to Computing Resources) offers a ready-to-run Grid system including client and server software. UNICORE makes distributed computing and data resources available in a seamless and secure way in intranets and the internet.

3.4.1 Background

UNICORE in EMI is a set of Client (i.e. UCC and HiLA) and Server (i.e. core and UVOS) packages, all available for Windows, Linux/UNIX and MacOS.

3.4.2 Development Environment

- Code is kept in SVN, hosted by SourceForge [R16]

- As tracker system the one offered by SourceForge [R17] is used to report bugs, patches and feature requests;
- WIKI [R18] is used as a workarea, to collect development plans, quick guides for users, admins, developers, FAQs.
- Two mailing-lists are used, one for developers discussions (*unicore-devel*) and one for release announcements (*unicore-announce*).
- The Atlassian Bamboo [R19] is used to manage the build & test process.

3.4.3 Release Categories

Regular releases of UNICORE core components (e.g. core servers, workflow, ucc, Eclipse client) are performed at a schedule of approximately three months. The release process combines a t*ime-based release* method with *continuous release* one, trying to release approximately every three months. If there are no changes, like bugs solved or improvements there is no release. The benefit of having a new release has to outweigh the work associated with it.

3.4.4 Workflow:

- A new version of a component is built and tested thoroughly by the developers and any additional "in-house" testers;
- The "release candidate" of the component is made available for download on public repository. It is announced on development mailing list (unicore-devel);
- There is a 10 working days “voting period“;
- In case of bug reports, these are fixed, and the release candidate is updated;
- If no objections are raised and no (further) critical bugs have been reported, the software is released on SourceForge repository.

4. EMI RELEASE PROCESS

4.1. DESCRIPTION OF THE LIFE-CYCLE

During the three year duration of the EMI project, the EMI developers and engineers will work together to consolidate, harmonize and support the existing software products, evolving and extending them based on existing and new requirements. Redundant or duplicate services resulting from the merging are deprecated; new services can be added to satisfy user requirements or specific consolidation needs. Input for the development activities is taken from users, infrastructures projects, standardization initiatives or changing technological innovations. The software components are adapted as necessary to comply with standard open source guidelines to favour the integration in mainstream operating system distributions.

The maintenance and development of the EMI middleware services is based on a 5-step yearly cycle (Fig. 1) composed of:

1. **The Requirements Analysis phase**: input collected from the EMI collaboration activities or from direct user submission in the EMI User Support system is analysed and prioritized based on the Severity assigned by the users, the urgency, impact, cost and available effort. The result of the analysis is compiled in the form of the EMI Technical Plans defining the project technical objectives (EMI Deliverables DNA1.3.x, DJRA1.1.x, DJRA1.2.x, DJRA1.3.x, DJRA1.4.x, DJRA1.5.x and DJRA1.6.x). The plans are defined at the beginning of the project and refined at every cycle based on the new input. The requirements analysis and the overall technical plan are coordinated by the PTB and the specific Technical Areas and Standardization Plan are coordinated by the Technical Area leaders and the Standardization Task leader within JRA1.
2. **The Development and Test Plans phase**: based on the latest version of the project Technical Plans, the Development and Test Plans for the current cycle are defined. The plans outline which of the technical objectives can be included in the cycle, which components are involved, which platforms and operating systems can be targeted, external and internal integration constraints, development, deployment and testing timelines, etc. The plans are centrally coordinated by the JRA1 Work Package leader and distributed to the EMI PTs for implementation. The Release Plan is established by SA1 with the details of the timelines to be applied to each release cycle (like code freeze date, release date, any technical preview release date) and the outline of the set of acceptance criteria to be fulfilled by the components (like documentation and specific categories of tests).
3. **The Development, Testing, Certification phase**: based on the Development and Test Plans, the various PTs develop the new functionality and perform initial unit, integration, deployment and functional tests under the overall coordination of JRA1 and monitoring of the PTB. Once a piece of functionality has passed the foreseen set of acceptance criteria and has been certified, it is released as Release Candidate to SA1.
4. **The Release Certification and Validation phase**: as Component Releases are transitioned from JRA1 to SA1, the final certification phase starts. During this phase, the software components are validated against the set of acceptance criteria defined by the customers. Technical Previews are made available to users and projects taking part in the "Works with EMI" technical collaboration program and the final packaging and signing is performed. Components that do not pass the criteria are rejected back to JRA1 for revision.
5. **The Release and Maintenance phase**: once the software components have passed all acceptance criteria, they are released and uploaded to the official EMI Software Repository from where they can be picked up by users and infrastructure operators.

The continuous maintenance phase starts at this point, any defect found by users in production environments and submitted to the EMI User Support system (GGUS) are analysed, prioritised and addressed as revision or minor releases. Any component that has not passed all criteria by the time the final release is due it is rejected and reschedule for another release cycle.

Figure 1: EMI Release Cycle

4.2. PROCESS OVERVIEW

4.2.1 Roles and responsibilities

Developers are responsible for the maintenance of a particular part of the codebase, which includes fixing bugs and adding new features to software components. They can belong to one or more Product Teams, depending on the components they are developing

Product Teams (PT) are small teams of software developers, fully responsible for the successful release of a particular software product (or group of tightly related products) compliant with agreed sets of technical specification and acceptance criteria. The individual PTs are requested to execute the necessary processes and procedures and respect established requirements and priorities as globally defined and endorsed by the EMI project executive and technical leadership.

Product Team Leader (PTL) is responsible for managing a specific PT, for the design, development, support and maintenance of specific products

Software Quality Control Leader (SQCL) is responsible for the Software Quality Control layer, which ensures both SQA and SQAP are being followed by the development teams. All the releases of EMI components need to satisfy well-defined certification and validation criteria before being

included in a stable EMI distribution, sufficient to guarantee to a high degree of confidence that all EMI products meet or exceed the requirements set by EGI and that no regression are introduced.

Technical Area Coordinators (TACs) are the members of the Project Technical Board responsible for the overall coordination of each of the four technical areas: compute, data, infrastructure and security areas. They are responsible to define together with the relevant Product Team Leaders the technical strategy of their area. They are also responsible to define together with the TD and the other TACs the overall project technical plans. The TAC are proposed by the PD and the TD and approved by the CB.

Release Manager is responsible for overall coordination. Main duties include assuring the good functioning of the release process, managing release delivery and scheduling, chairing the EMT and ensuring that its agenda reflects all relevant issues of the day.

Technical Director (TD) is responsible for defining the Technical Development Plan and is also responsible for deciding on specific technical issues.

JRA1 activity leader is responsible for collecting the technical objectives of the project from the different technical area work plans, making sure they are aligned with the Technical Development plan and together with the Release Manager, define the release schedule.

4.2.2 Decision-Making bodies

Engineering Management Team (EMT) is lead by the Release Manager and composed of the PTLs (or duly appointed representatives), a QA representative, a Security representative, representatives of the operations teams of the major infrastructures (like EGI and PRACE.) and invited experts as necessary. It manages the release process and the "standard' changes, that is all the changes that are pre-approved based on established policies and do not need to be individually approved by the PEB or the PTB, like: software defects triaging and prioritization, technical management of releases, integration issues, and user support request analysis.

Project Technical Board (PTB) is lead by the TD and composed of the Technical Area Leaders and a representative and is responsible to assist the TD in defining the technical vision of the project and deciding on specific technical issues. The PTB members are the coordinators of the project technological areas.

Project Executive Board (PEB) is responsible to assist the Project Director in the execution of the project plans and in monitoring the milestones, achievements, risks and conflicts within the project. It is led by the PD and is composed of the Work Package Leaders, the Technical Director and the Deputy Technical Director.

4.2.3 Release Categories

The EMI distribution will be organized in periodic major releases tentatively delivered once a year, providing a good balance between the conflicting requirements of stability and innovation.

An EMI major release is characterized by well-defined interfaces, behavior and dependencies for all included components, available on a predefined set of platforms. What is included in a new EMI major release is defined by the PTB and the implementation of the plan is coordinated by JRA1.

Backward-incompatible changes to the interface or to the behavior of a component that is part of the EMI distribution can be introduced only in a new EMI major release. Changes to interfaces that are visible outside the node where the component runs (e.g. a WSDL) need to be preserved even across major releases, according to end-of-life policies to be defined on a case-by-case basis.

The availability of a new major release of EMI does not automatically obsolete the previous ones and multiple major releases may be supported at the same time according to their negotiated end-of-life policies.

4.2.3.1 Components Releases

An EMI distribution includes all the components that are developed within the project and that have reached production quality. Within an EMI major release, only one version of a given component is maintained. Four types of releases have been identified for a given component:

- **Major Release**: A major release for a component is characterized by a well-defined interface and behavior, potentially incompatible with the interface or behavior of a previous release. New major releases of a component can be introduced only in a new major release of EMI. The contents of a new major release are endorsed by the PTB and included in the project technical plan. The implementation is coordinated by JRA1.
- **Minor Release**: A minor release of a component includes significant interface or behavior changes that are backwards-compatible with those of the corresponding major release, but it can include new functionality or roll-ups of sets of software fixes not previously released as dedicated revisions. New minor releases of a component can be introduced in an existing major release of EMI. The contents of a new minor release are endorsed by the PTB and included in the project technical plan. The implementation is coordinated by JRA1. If the release is going to be introduced in an existing major release of EMI, the implementation is also supervised by SA1 in order to guarantee that the production quality and the backwards-compatibility are preserved. It is expected that minor releases will be made available a few times per year.
- **Revision Releases**: A revision release of a component includes changes fixing specific defects found in production and represents the typical kind of release of a component during the lifetime of an EMI major release. It cannot include new functionality or change the nominal expected behaviour of the components. Revision releases will be made available very frequently and user will be able to decide whether to apply the revision or not based on the existing impact of the defects on their activities.
- **Emergency Releases**: An emergency release of a component includes changes fixing only high-impact, high-severity bugs like security issues and can follow special shortened release procedures. Emergency releases are made available as needed.

The type of release is reflected in the version of the corresponding package(s), as described bellow.

4.2.3.2 Versioning Schema

For components releases, EMI is adopting, as VERSION-RELEASE numbering convention, the typical major.minor.revision(-[age/release]) schema, where:

- increment in major# reflects a change in the component interface or behavior, that can be backward incompatible;
- increment in minor# reflects a change in the component interface or behavior backward compatible;
- increment in revision# reflects bugs fixes, with no new features;
- increment in release# reflects a rebuild due to change in the external dependencies; changes in the ABI that implies the rebuild of all the clients or an emergency release for component-x.y.z, when the version component-x.y.z+1 is already available..

4.2.3.3 Types of Releases

All EMI major releases will be **full releases**, meaning that all software components are build, tested, distributed together, including components that were not changed.

All minor, revision or emergency releases will be partial, **delta**, releases, including only those software components that are changed or are new, since the last full or delta release;

4.2.4 Build Process

All the EMI software components are built together in a single unit. This build produces QA metrics, binary and source packages. Binary packages can be used for testing or release. Source packages will be used in the future for builds for OS distributions.

A single tool is used to perform integration builds from source. The system of choice is ETICS [R13]. All build tools will be supported and integrated via this unified integration system.

Build Environment:

- EMI is released on two sets of platforms, **mandatory** and **optional**.
- **Mandatory** platforms must be supported by all components in EMI, **optional** platforms can be supported by individual PTs depending on needs of technical capabilities.
- For every major release the details on what platforms and operating systems can be targeted are contained in the Development and Test Plans prepared by the JRA1 Work Package
- All components in EMI must be provided in the standard packaging formats typical of the supported mandatory platforms. Currently this means rpms for RHEL5 and compatible platforms and debs for Debian 5 and compatible platforms.

Build and Configuration Guidelines [R20] are prepared by the SA2 team.

4.2.5 External Dependencies

One of the issues of having to integrate four middleware stacks is that the external dependencies used by different middleware distributions are of different versions and are taken from different sources All EMI software components external dependencies will be compatible with the ones provided by the OS or the supported YUM or APT repository. Globus packages for EMI shall be taken from existing standard repository such as EPEL whenever available. All components in EMI will adopt this strategy.

Exceptions will be discussed at the EMT meetings:

- If a package not available in OS or EPEL is required and right package is available from a third party, then it will be added to an EMI third-party repository that will be used as source of external dependencies as well.
- If a package is not available in the right package format (like RPM or DEB) and it is required as dependency, the requester PT will be responsible of creating the proper package, building the package from source and making it available as an ETICS external.

4.2.6 Release Criteria

EMI's main objectives are to deliver a consolidated set of middleware components for deployment in EGI (as part of the Unified Middleware Distribution or UMD), PRACE and other DCIs, extend the interoperability and integration between grids and other computing infrastructures, strengthen the reliability and manageability of the services and establish a sustainable model to support, harmonise and evolve the middleware, ensuring it responds effectively to the requirements of the scientific communities relying on it.

In order to be able to decide when the release is ready to be deployed a set of release criteria; Production Release Criteria [R21], are defined by the SA1 and SA2 teams, and verified during the different phases of the software components life-cycle.

The Production Release Criteria defines the minimum set of criteria that a component release needs to respect in order to be approved by QC. The approval takes place once the component release has been certified by the PT as described in the Change Management Guidelines [R22]. The criteria are defined after the instructions given in the different guidelines provided by the SA2 -QA team. The instructions that are defined as mandatory are part of the Production Release Criteria:

- Integration and Configuration criteria:
  - All components to be released as part of EMI must have configurations in ETICS.
  - EMI is released on two sets of platforms, mandatory and optional; Mandatory platforms must be supported by all components in EMI.
  - Externals dependencies must be compatible with the ones provided by the OS or the supported YUM/APT repository.
- Packaging criteria:
  - EMI software components must conform the Fedora [R23] and Debian [R24] packaging policies.
- Testing and Certification criteria (guidelines on how to certify the EMI software components so that they can be included in the EMI release are given in the Certification and Testing Guidelines [R25]):
  - All software components to be released should provide Software Verification and Validation Reports showing successful results of unit, deployment, functionality, regression, standard compliance/conformance, performance, scalability tests, written following the Certification and Testing guidelines.
- Metrics criteria:
  - Are defined in the SQAP [R26] and Metrics Guidelines [R27].
- Component Release criteria:
  - Defined taking in consideration the Customers acceptance criteria, contained in documents like the EGI Software Provisioning Process [R28] and EGI Quality Criteria [R29].
  - All code must compile and build for all mandatory platforms in the integrated environment provided by ETICS.
  - Successful configurations are provided for each component release registered in the Release Schedule tracker [R30].
  - Zero high priority bugs affecting the component to be released.
  - Minimum documentation is available, like release notes, installation guide, user documentation, troubleshooting guide.
  - All Components Releases fields described bellow should be filled.

4.2.7 EMI Releases Repository

EMI will provide a basic YUM/APT repository based on AFS and HTTPD services provided by CERN.
The repository will contain the software components developed during the life-time of the project, with the following structure:

- for each EMI major release, EMI-[1,2,3], there will be

- EMI-production (stable), called emi-prod, contains stable and signed, well tested software components, recommended to be installed on production-sites;
- EMI-testing, called emi-test, contains packages that will become part of the next stable distribution, passed the certification & validation phase, available for technical-previews;
- EMI-development (unstable), called emi-dev, contains packages in development phase; not publicly available;
- EMI-third-party, called emi-third-party, contains packages not available from the OS/EPEL repositories, needed by the EMI software components;

- Each of the EMI major releases will be split by platform, mandatory and optional, defined in the Development & Test Plans document [R41], and by architecture.

The EMI Repository will contain always non conflicting packages and only packages that are not commonly available on standard OS repositories or other mainstream repositories such as Fedora or EPEL.

A web interface for browsing the repository is available at http://emisoft.web.cern.ch/emisoft.

4.2.8 Technical management of Releases

Next figure represents how a release is defined.

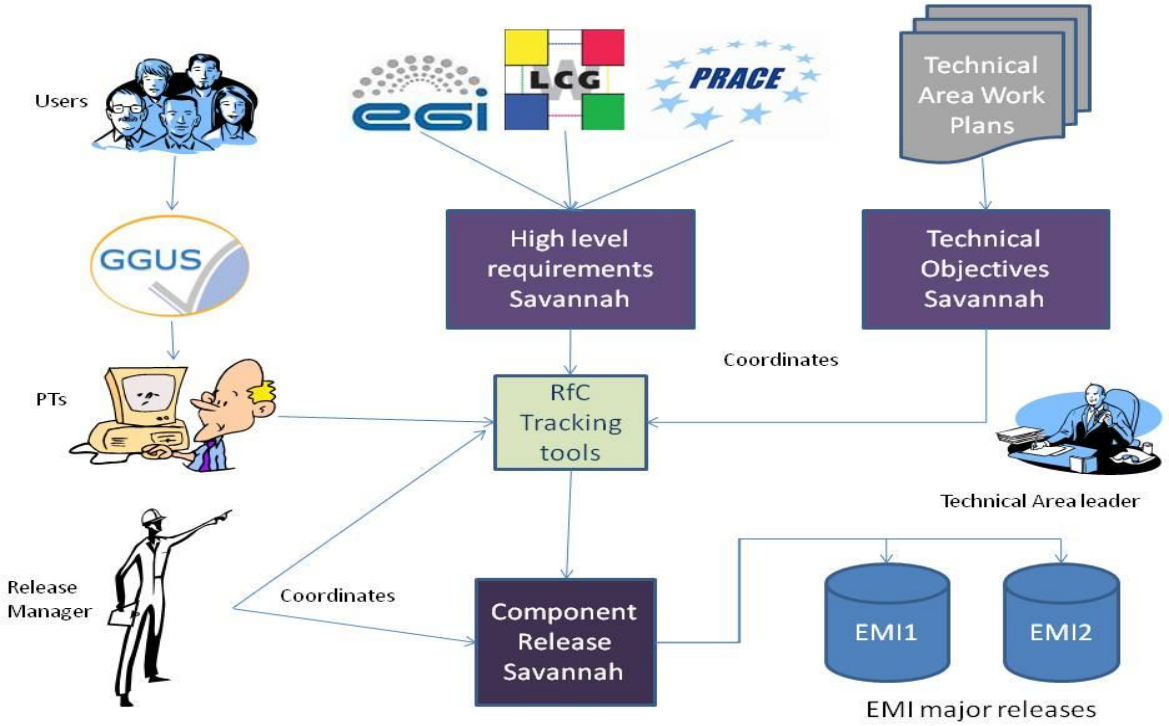


Figure 2: Release Management Process

The following items are key components of a release:

- High level requirements: represent high level descriptions of requirements that imply changes in the software that need to be discussed and approved first by the PEB/PTB. They are coming from the major DCIs using the EMI middleware.

- Technical Objectives: The technical objectives are the milestones defined in the technical area work plans.
  - Compute Area Work Plan [R31];
  - Data Area Work Plan [R32];
  - Infrastructure Area Work Plan [R33];
  - Security Area Work Plan [R34].
- GGUS tickets: Users should always report incidents or requests through the GGUS portal. GGUS incidents/requests will be transformed into RfCs when accepted by the developers;
- Changes: Changes are described in a Request for Change (RfC). The Software Maintenance and Support Plan Deliverable [Ref] contains more details;
- Component Releases: Component releases are the collections of RfCs to be released.

A component release is the basic release unit. It's comprised of RfCs that may be motivated by GGUS, high level user requests or technical objectives. RfCs can also be changes originated within the PT itself.

4.2.8.1 Release Trackers

For keeping track of the Release Process the items described above will be recorded using:

1. **User Requirement Tracker**:
   - It can be found in Savannah under: EMI Requirements Tracker [R35];
   - It tracks user requirements coming from the DCIs using the EMI software;
   - The PTB is responsible for making sure that an entry is created in this tracker for each user requirement, where the corresponding Component Releases will be attached;
   - The PTB is responsible for tracking the open entries and closing them when the user requirements have been implemented.
2. **Technical Objectives Tracker**:
   - It can be found in Savannah under: EMI Technical Objectives Tracker [R36];
   - It tracks technical objectives defined in the different technical area work plans;
   - The JRA1 activity leader is responsible for making sure that an entry is created in this tracker for each technical objective per technical area, where the corresponding Component Releases for that technical area will be attached;
   - The JRA1 activity leader is responsible for tracking the open entries and closing them when the objective has been addressed.
3. **Component Release Tracker**:
   - It can be found in Savannah under: EMI Release Schedule Tracker [R30];
   - It tracks the component releases. The set of open entries forms the release schedule;
   - The release manager is responsible for making sure that an entry is created in this tracker for each component release, where the corresponding RfCs of that component will be attached taking into account their priority. This has to be done in coordination with the JRA1 activity leader and the different PTs, making sure all the technical objectives and user requirements are addressed within the expected deadlines. The component releases are described in detail in the Change Management Guidelines. [R22];

- The Release manager is responsible for monitoring the progress of the release candidates, making sure with the help of QC, that they meet the Production Release Criteria [R21], defined by SA2, and the release schedule.

4. **RfC Tracker**:
   - It tracks RfCs for the different middleware components. RfCs are described in detail in the Change Management Guidelines [R22];
   - The technical area leaders are responsible to make sure that the technical objectives and user requirements associated to the software components in their area are mapped into entries in the RfC tracking tool;
   - PTs are responsible for creating entries in the RfC tracking tool. In this case, RfCs can be motivated by those GGUS tickets describing a problem or after an internal decision within the PT;
   - Each RfC should have a priority defined. The assignment of priorities is described in the Change Management Guidelines.[R22].

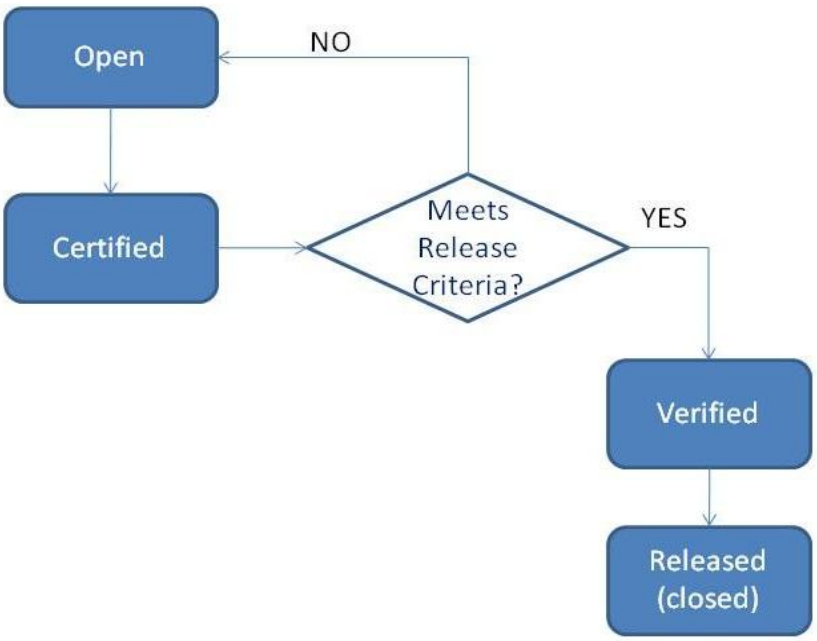
4.2.8.2 Components Releases tracking

Component Releases should contain the following information:

- **Unique identifier**;
- **Component name**: The name of the component should be one of the list of EMI software components defined in Technical Development Plan DNA1.3.1 [R37];
- **Component version**: The version of the component should be defined according to the versioning-schema;
- **Due Date**: deadline for the component release to be certified by the PT;
- **EMI major release**;
- **Packages**: list of packages affected by the change and the URL from where they can be downloaded;
- **Release Notes**: non-technical text written in good english giving an overview of the change introduced by the packages. The text should be prepared by the PT responsible for the component. It should contain the following structure:
  - **What's new**: Brief description of the main changes, both new features and bug fixes;
  - **Installation and configuration**: More details on installation and configuration stating very clearly if the service must be reconfigured and/or restarted;
  - **Know issues**: Known issues present in the release, possibly with a workaround;
- **List of RfCs**: Links to the corresponding items in the different tracking systems describing which bugs/feature requests are fixed in this release;
- **Test report**: Link to the test report where the test results of the executed tests are included. This should prove that all mandatory tests relevant to the released packages have been executed successfully;
- **Documentation**: Link to the relevant documentation (user guides, troubleshooting guides);
- **Known Issues**: List of known issues associated with the release;
- **Changelog**: For each of the released packages, a changelog containing the developer's comments associated with each change should be included.

Component releases are created in Release Schedule tracker by the release manager although PTs will have to fill in the requested information as describe in the next section.

4.2.8.3 Component Release state transition



- **Open**: the Release manager is responsible for creating CR items in Savannah to track the different scheduled releases. This should be done with the assistance of the JRA1 activity leader making sure the different development plans are fully covered in the release schedule. All CRs should be either planned according to the different development plans, or in case they are needed because an unplanned change needs to be introduced, new CR items will be created in Savannah at any time by the Release manager. At this moment, the Release manager together with the PTs, defines a Due Date and some other mandatory fields.
- **Open to Certified**: the PT moves the CR to Certified once the development and the certification of the CR has finished. Before this transition happens, the PT has to make sure all the fields in the CR described above are properly filled in according to the EMI Production Release Criteria [R21].
- **Certified**: the work of the PT has finished at this stage. This stage triggers the SA1 QC team to verify that the CR complies with the EMI Production Release Criteria [R21].
- **Certified to Open**: The SA1 QC team moves the CR to Open if the evaluation of the information contained in the CR fails to meet the EMI Production Release Criteria. The SA1 QC team should include the result of its evaluation as an attachment to the CR.
- **Certified to Verified**: The SA1 QC team moves the CR to Verified if the evaluation of the information contained in the CR succeds to meet the EMI Production Release Criteria. The SA1 QC team should include the result of its evaluation as an attachment to the CR.
- **Verified**: the work of the SA1 QC team has finished at this stage. This stage triggers the SA1 team responsible for maintaining the EMI production repository.
- **Verified** to **Released**: The SA1 team includes the new packages in the EMI production repository. They also announce the new release in the EMI web pages and through any other communication channel agreed with the DCIs.
- **Released**: the component release is available in the EMI production repository. This state automatically closes the savannah ticket.

4.2.8.4 Coordination Meetings, Communication Channels

The release manager is responsible for organizing EMT coordination meetings where outstanding issues are discussed with the relevant parties. The agenda for the meetings should contain:

- Report on the status of releases by the release manager;
- Report on continuous integration by the release manager;
- Report about the progress of component releases not meeting the release schedule. This will be done by the relevant PT. A new deadline for the component release should be defined;
- Report about problems blocking a Component Release. This will be done by the PTs affected in the problem;
- Report about new RfCs with Priority High. The release manager will inform the EMT about unplanned RfC that have priority High, which means that the RfC needs to be included in the next release of the affected component;
- QA announcements. An SA2 representative should always be present at the EMT meetings to answer any question related to guidelines, tools or testbeds.

Meetings are held using the EVO system [R38] and the agendas are available under Indico [R39]. Developers are encouraged to subscribe to the mailing-list emt@eu-emi.eu, used for the EMT discussions, communications of meetings. All details of EMT activity are included a dedicated twiki page [R40].

The use of the Savannah project management tool [R49] was started for tracking EMT actions and EMI-0 tasks. The use of this tool will increase once the tracking activity for the EMI-1 release starts

4.3. EMI RELEASE PROCEDURES

4.3.1 Scheduled Release Procedure

Major, minor and revision releases will follow a scheduled release procedure, consisting of

1. Release Planning – summary of activities
   - identify requests for new features to be developed & bug fixes to be implemented;
   - prioritize, plan and schedule the development & maintenance activities;
   - document & track release planning activities in the Development & Test Plans and by creating items in the User Requirements & Technical Objective Trackers & RfCs in the PTs trackers with approved priorities;
   - define the Release Schedule by creating the Components Releases items in the corresponding tracker;
2. Release Building – summary of activities
   - develop new features, implement required bug fixes;
   - test & certify developed components;
   - Guidelines are provided by the SA2 regarding:
     - Build Integration & Configuration [R20], defining how the four middleware distributions will manage to have a successful shared single build using standard external and internal dependencies and how to configure the selected build/integration tool to properly build each software component;

- Packaging[R23] - EMI components will follow the Fedora & Debian packaging guidelines;
- Certification & Testing [R25], containing details on what kind of tests to perform and when; how to write a Software Verification and Validation Plan, how to perform certification on a component release; how to write a Software Verification and Validation Report;

3. Certification & Validation (acceptance testing) – summary of activities
   - Component Releases are validated by the SA1 QC against the set of acceptance criteria defined by the Customers [R28] and the EMI Production Release criteria [R21];
   - components are made available for technical-preview;
   - components are deployed on the EMI-testbed for a (6 days) grace period under an automatic monitoring tool [R42];
4. Release Preparation & Deployment – summary of activities
   - final packaging & signing of components;
   - components uploaded in the official EMI Software Repository [R43];
   - prepare & publish release documentation [R44]: Release Notes, Known Issues;
   - announce the Release to user communities/production infrastructures/customers, using the following delivery strategy:
     - general announcements:
       - EMI RSS Feed [Ref];
       - emi-announce mailing-list;
       - general user-communities mailing-lists (lcg-rollout) [R47];
     - specific announcements:
       - following EGI Staged-Rollout procedures [R45]: a GGUS ticket will be used to announce the availability of a new release; it's status will be followed in order to declare the Component Releases closed and the release deployment successful;
       - other DCIs procedures.

4.3.2 Emergency Release Procedure

When a major incident, a defect considered critical or a security hole, is discovered during everyday usage of a software component or service in the production-infrastructure, or during bug fixing, debugging, regular testing activities or is triggered by external dependencies, an emergency release procedure will be followed. The entire process cannot take more than few days.

1. Release Planning:
   - An RfC and a corresponding CR should be created in the respective trackers;
   - The incident is analyzed and a priority is assigned to the RfC reflected afterwards in the type of release that will be prepared:
     - emergency release for the affected component/s, or
     - the solution will be provided in the next scheduled (revision/minor/major) release.

- Decision is taken by PTB & TACs if the emergency release of the component should be prepared on all mandatory platforms, or only on the affected ones;
- Decision is taken and communicated to developers, PT, about the deadline for providing the solution;

2. Release Building:
   - find the solution, implement & test it by the PT/s who's component/s are affected;
   - successfully build the package;
   - update CR with the necessary information (release notes, changelog, test teports)
3. Certification & Validation (acceptance testing):
   - SA1-QC team verifies test reports;
   - component/s is/are deployed on the testbed, continuously monitored;
4. Release Preparation & Deployment:
   - component/s are build, signed, uploaded to the EMI Release repository;
   - release notes are updated reflecting changes (fix or workaround) ;
   - release is announced using the general & specific channels described above.

4.4. RELEASE SCHEDULE

The EMI Roadmap is structured on four major releases, called EMI 0 (Zugspitze), EMI 1 (Kebnekaise), EMI 2 (Matterhorn) and EMI 3 (Monte Bianco). The sequence of releases is shown in Figure 3.

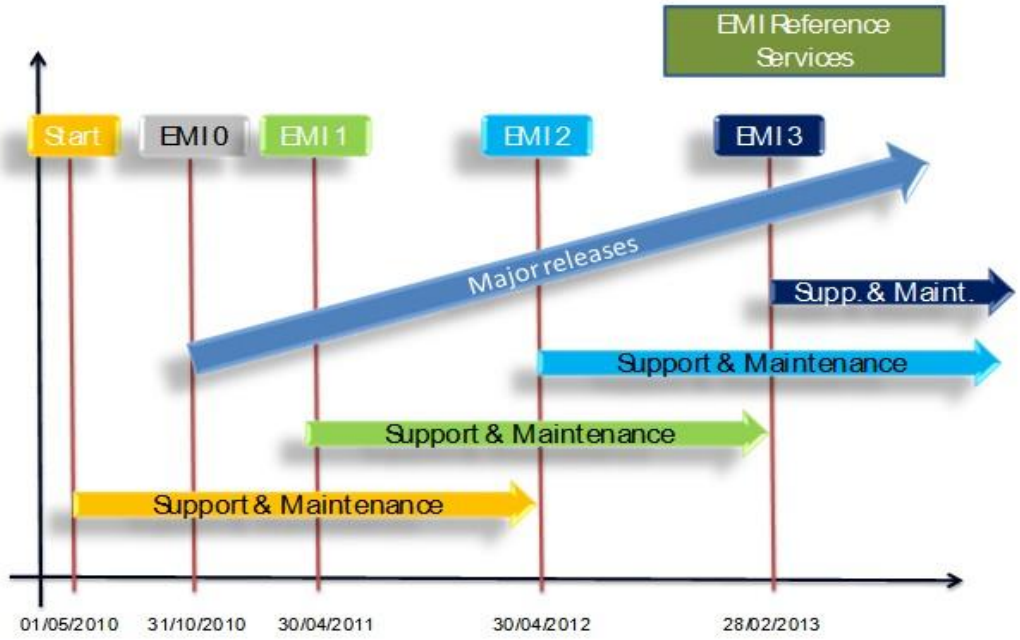


Figure3: The EMI Release Roadmap

The EMI releases are time-based. They are scheduled in advance for a certain date and a proposed content described in the Development Plans [Ref]. As the release date approaches, components that

are found not to meet the release criteria in time for the release are dropped and rescheduled for another release.

The release called EMI 0 is an internal release meant to be used as internal benchmark for validating the software engineering processes, the actual skills of EMI in managing the release, and specifically to construct a solid baseline for the public releases after solving the expected integration issues coming from merging together products from four middleware collaborations (ARC, gLite, UNICORE and dCache). The EMI 0 release follows an abbreviated cycle, where essentially only a subset of the integration and deployment testing is done as part of the Certification phase. Apart from EMI 0, all other releases follow the standard cycle and are followed by a period of support and maintenance according to the policies defined in the Software Maintenance and Support Plan [R4].

The major releases are checkpoints used to make sure that all technical objectives planned for that release are indeed implemented and working as expected, that all interface contracts among components are respected and that the acceptance criteria defined by the customers are all satisfied for all released components.

Between any two releases, each EMI Product can follow an internal release cycle similar to the overall release cycle, but with shorter timelines. The only constraint that each Product has to respect, apart from the overall acceptance criteria, is that its interfaces have to be backward compatible with the release currently in production. Any non backward compatible change must be released only as part of a major EMI Release at the end of the main release cycle. Using this release cycle, specific products versions can be released at any time during the year as minor releases or bug fix releases, following the plan defined in the development and release plans at the very beginning of the overall release cycle.

The foundation of the technical objectives scheduled bellow can be found in the EMI Technical Objectives [R46]

4.4.1 3-years Release Schedule

EMI-Releases are time-based, tentatively delivered once per year, offering good balance between the conflicting requirements of stability and innovation, allowing in this way customers to plan in advance organize their updates, to have a preview of what will be available.

Release	CodeName	ReleaseDate	EndOfSupport
EMI-1	Kebnekaise	April 30, 2011	April 30, 2013
EMI-2	Matterhorn	April 30, 2012	April 30, 2014
EMI-3	MonteBianco	February 28, 2013	ToBeDefined

Table 1: EMI 3-year Major Release Schedule

4.4.2 Schedule Methodology

EMI Release Schedules are proposed by the Release Manger to the EMT for discussion, the final version gets approved by the PEB/PTB.

Task/Milestone	Start Day	Length
Planning & Development	Day after the end of the Planning phase for the previous GA Release	More than 5 months
Feature Submission Deadline	5-8 months before the Feature Freeze	n/a
Feature Freeze	2 month before the RC (Dic 20xx)	Until GA

Testing & Certification	1 day after the Feature Freeze	2 months
Feature Complete	1 month after the Testing & Certification begin	1 month
Final Change Deadline (code freeze)	End of Testing & Certification	Until GA
Release Candidate	End of Testing & Certification Start Acceptance Testing 2 months before GA-release day (Feb 20xx) 3 days after the Final Change Deadline	Until GA
Compose Final RC	Same as Release Candidate	1 week
Technical-Previews	1 month before the GA	Should last 2 weeks
Test Final RC	2 weeks before GA	6 days
GA Release	April 30, 20xx	n/a
Maintenance	GA release day	~2 years
End of Life	2 years after the GA of current release	n/a

where:

Feature Submission Deadline:

- final date that new features can be submitted to the release;
- new features must be submitted to next release.

Feature Freeze:

- feature development is complete or in a significantly complete state so that the feature can be adequately tested;
- after the Feature Freeze: new enhancements should not be added; changes to components that require other (dependent) software components to make changes as well, should be avoided;
- for introducing exceptions an exception process and evaluation will be defined.

Feature Complete:

- software components contain all intended functionality of the final version;
- still have to undergo testing and bug fixing, performance or stability enhancement before they can become Release Candidates.

Final Change Deadline:

- code freeze date meaning that no changes whatsoever are permitted to components source code. Critical components, components on which others depend, will have a different code freeze calendar;
- no more updates are made to the emi-dev repository; packages are moved to the emi-test repository.

GA Release – General Availability Release:

- software components are delivered to Customers

4.4.2.1 Maintenance Schedule & End-of-Life

EMI major releases will be maintained for approximately 2 years, after the corresponding release date, as mentioned in the Table 1: EMI 3-year Major Release Schedule. When a release reaches the point where it is no longer supported, updates are no longer created for it and it is considered *End of Life* (EOL). A specific procedure will be created, containing the tasks to be followed in case of EOL.

4.4.3 EMI-1 (Kebnekaise) Schedule

4.4.3.1 Key Features

Component	Summary	Status
A-REX	support of GLUE2 information providers.	75%
CREAM	support of GLUE2 information providers.	50%
U.TSI, U.XNJS, U.UAS, WSRFLite	support of GLUE2 information providers.	
WMS	matchmaking module of the WMS will be enhanced to be compliant with GLUE2.	
Libarcclient/ arc*	enhanced to work with GLUE2-compliant information.	
UCC	enhanced to work with GLUE2-compliant information.	
HILA	enhanced to work with GLUE2-compliant information.	
DPM	enable GLUE2.0 support support of the HTTP(S) protocol support NFS4.1 (experimental version) prototype-level support for the "file://" access protocol.	
dCache	enable GLUE2.0 support enable the use of HTTP(S) protocol prototype-level support for the "file://" access protocol.	
UAS-D	expose GLUE2.0 compliant information about storages.	
StoRM	enable GLUE2.0 support support of the HTTP(S) protocol prototype-level support for the "file://" access protocol.	

Detailed Feature List is contained in the Development & Test Plan [Ref] and tracked in the Release Schedule tracker [R30]

All software components to be part of EMI-1 will go through the Testing & Certification process, even if no developments are planned with respect to the production versions, to ensure that acceptance and release criteria are met.

4.4.3.2 Key Milestones

Start Date	Task/Milestone
May 1, 2010	EMI project starts Planning & Development Begins
October 1, 2010	Feature Submission Deadline
December 31, 2010	Feature freeze (Development ends)
January 1, 2011	Testing & Certification
February 1, 2011	Feature Complete
February 28, 2011	Final Change Deadline (code freeze)
March 1, 2011	RC/Compose Final RC
April 1, 2011	Technical -Preview
April 15, 2011	Test Final RC
April 30, 2011	GA Release

4.4.4 EMI-2 (Matterhorn) Schedule

Year-2 of the project development activity will be mostly focused on integration and deployment of messaging services, integration of virtualization technology and standardization of interfaces. Details will be contained in the Development And Test Plans document to be prepared by the JRA1.

4.4.4.1 Key Features

Technical Area/Components	Summary
Compute Area (2): * A-REX * CREAM * UNICORE - EMS * libarcclient * arc* * CREAM Client * UCC * HILA	Implementation of the agreed common job submission and management methods
Compute Area (3): * A-REX * CREAM	Provide limited interactive access for at least one EMI Computing element
Compute Area (4): * A-REX * CREAM * UNICORE XNJS	Support for the agreed compute accounting record (UR)

* UAS-Compute * UNICORE TSI	
Compute Area (5): * libarcclient * arc* * CREAM client * WMS client * UCC * HILA	Consolidation and harmonization of compute area clients/APIs
Compute Area (6): * A-REX * Janitor * WMS * CREAM * UNICORE EMI-XS * EMI Compute Clients	EMI compute clients are able to request access to virtualized resource managers and appliances
Data Area (5): * dCache * StoRM * DPM * UAS-Data * GFAL * libarcdata2	All storage elements publishing full set of GLUE 2.0 storage information
Data Area (6): * dCache * StoRM * DPM	All storage elements offering support for the WebDav protocol
Data Area (7): * dCache * StoRM * DPM * All data clients	Using https instead of httpg for the SRM protocol as a production implementation in all the storage elements and clients
Data Area (8): * dCache * StoRM * DPM * FTS * GFAL * libarcdata2	Overall consolidation of data area by adopting a consistent interpretation of SRM
Data Area (9): * GFAL * libarcdata2 * emi_datalib (new)	Providing a common set of data access libraries at least between gLite and ARC
Data Area (10): * LFC * dCache * StoRM * DPM	Solve the synchronization problem of the storage elements and the file catalogue

Infrastructure Area (6): * EMI Registry (new)	Implement the common EMI Registry
Infrastructure Area (7): * All EMI services * all infosys clients (WMS included)	Fully utilize and support the GLUE2 information model
Infrastructure Area (10): * DGAS HLR sensors * APEL-publisher * JURA * new-publisher for data area services	Implement or adapt the accounting record publishers of compute and data area services to use the common messaging system
Security Area (2): * STS * SLCS * VOMS	Simplified management of security credentials by reducing the complexities of handling certificates
Security Area (3): * Emi_authNlib (new component)	Provide common authentication libraries supporting X.509 and optionally SAML
Security Area (4): * EMI security clients and utilities	Consolidation and reduction in the number of security CLIs
Security Area (5): * VOMS * UVOS	Agreement and full support for a common single X.509 and SAML based Attribute Authority Service
Cross Area (5): * A-REX * CREAM * U.EMIEX (new)	An EMI-blessed delegation solution for at least the computing area
Cross Area (6): * A-REX * CREAM * CEMON * UNICORE Compute and Data * UNICORE/X * ARGUS * VOMS * STS * SLCS * Hydra	Definition and implementation of initial support for the common SAML profile all over the middleware stacks
Cross Area (7): * CREAM * CEMON * WMS * A-REX * HED * UNICORE/X * UAS-Compute * ARGUS	Integration of the compute area services with the ARGUS authorization framework
Cross Area (8): * DPM * dCache	Initial integration of the storage elements with the ARGUS authorization framework

* StoRM * FTS * ARGUS	
Cross Area (9): * DPM * dCache * StoRM * gfal * WMS * CREAM * A-REX * HED * libarcclient * libarcdata2 * VOMS	The legacy Globus security infrastructure (GSI) will be replaced with a common security solution
Cross Area (10): * all EMI services	Adapt or implement monitoring interfaces, sensors, providers for compute, data, security and infrastructure services
Cross Area (11): * all EMI services	Investigate service instrumentation interface for compute, data, security and infrastructure services

Component Releases will be created in the Release Scheduler tracker for all software components mentioned in the table above. Final list will be available once the Feature Submission Deadline for EMI-2 will be reached.

4.4.4.2 Key Milestones

Tentative Year-2 Schedule is presented in the table bellow (Table #). Definitive schedule will be adjusted after the EMI-1 release

Start Date	Task/Milestone
October 2, 2010	Planning
March 1, 2011	New Developments start
May 1, 2011	Maintenance of EMI-1 start
May 1, 2011	Feature Submission Deadline
December 31, 2011	Feature freeze (Development ends)
January 1, 2011	Testing & Certification
February 1, 2012	Feature Complete
February 28, 2012	Final Change Deadline (code freeze)
March 1, 2012	RC/Compose Final RC
April 1, 2012	Technical -Preview
April 15, 2012	Test Final RC
April 30, 2012	GA Release

4.4.5 EMI-3 (MonteBianco) Schedule

Year-3 workplan will consist on addressing new requirements, revision of business and exploitation plans.

4.4.5.1 Tentative Key Features

Technical Area/Components	Summary
Compute Area (7): * A-REX * CREAM * UNICORE TSI * UNICORE XNJS * UNCORE EMI-XS	Successful computational usage of emerging computing models i.e. clouds with EMI components (scaling out to clouds)
Compute Area (8): * MPI-Start * MPI-Utils * BLAH * CREAM * WMS * A-REX * UNICORE TSI * UNICORE XNJS * UAS - Compute	Provision of a common MPI execution framework, a "backend" across the different computing services
Compute Area (9): * MPI-Start * MPI-Utils * BLAH * CREAM * WMS * A-REX * UNICORE TSI * UNICORE XNJS * UAS - Compute	Extend the parallel computing capabilities to better address multi-core jobs on all emerging architectures resources
Data Area (11): * EMI data access clients	Completed migration to the common set of data access libraries
Data Area (12): * dCache * StoRM * DPM * UAS-Data * FTS	Add support for storage space usage accounting on the SE/FTS side
Infrastructure Area (11): * lcg-info * lcg-infosite * gstat-validation * GFAL	Consolidation and reduction in the number of information system discovery APIs and CLIs

* libarcclient * HILA * U. client libs * SAGA-SD * SAGA-ISN	
Security Area (6): * all security area components	Substantial simplification and reduction in the number of security area libraries, internal components and services
Security Area (7): * Pseudonymity * Hydra	Provide a transparent solution for encrypted storage utilizing ordinary EMI SEs
Cross Area (12): * all EMI services	Complete migration to the new AuthN libraries

4.4.5.2 Tentative Key Milestones

Start Date	Task/Milestone
May 2, 2011	Planning
March 1, 2012	New Developments start
May 1, 2012	Maintenance of EMI-2 starts Maintenance of EMI-1 continues
May 1, 2012	Feature Submission Deadline
October 31, 2012	Feature freeze (Development ends)
November 1, 2012	Testing & Certification
December 1, 2012	Feature Complete
December 31, 2012	Final Change Deadline (code freeze)
January 1, 2013	RC/Compose Final RC
February 1, 2013	Technical -Preview
February 15, 2013	Test Final RC
February 28, 2013	GA release EOL EMI-1 Maintenance EMI-2 continues Maintenance EMI-3 starts

5. INTERNAL RELEASE EMI-0 (ZUGSPITZE)

EMI-0 is an 'exercise' release designed to understand how to apply the agreed procedures, find any problem about tools and processes and in general fine tune the EMI software engineering process before the EMI 1 release. The outcome of EMI 0 is not expected to be released to external users. The goal of EMI 0 is to prepare a consistent, coherent repository of non-conflicting packages by the end of October 2010 without any specific commitment on functionality.

The specific goal of EMI-0 is to construct a solid baseline for the public releases after solving the expected integration issues coming from merging together products from four middleware collaborations (ARC, gLite, UNICORE and dCache). In order to achieve it's goals, general and specific procedures were agreed and implemented and followed through a schedule consisting mainly in build configuration and integration steps, presented in the MSA1.2.1 - EMI Reference Releases [Ref], and summarized below.

5.1. GENERAL PROCEDURES

- A single tool will be used to perform integration builds from source (nightly builds). All build tools will be supported and integrated via this unified integration system. Even though ETICS cannot be considered the optimal tool for each environment, it is the only one able to handle an integration build from source of all the four middleware distributions. Therefore it will be used to produce a first unified integration system.
- Globus packages for EMI shall be taken from existing standard repository like EPEL whenever available. All components in EMI shall adopt this strategy.
- EMI is to use a common build, test and QA system as much as practically possible. Currently the system of choice is ETICS. All components to be released as part of EMI must have configurations in ETICS.
- EMI is released on two sets of platforms, mandatory and optional. Mandatory platforms must be supported by all components in EMI, optional platforms can be supported by individual PTs depending on needs of technical capabilities.
- All components in EMI must be provided in the standard packaging formats typical of the supported mandatory platforms. Currently this means rpms for RHEL5 and compatible platforms and debs for Debian 5 and compatible platforms.
- The EMI Repository must contain always non conflicting packages and only packages that are not commonly available on standard OS repositories or other mainstream repositories such as Fedora or EPEL.

5.2. EMI-0 SPECIFIC PROCEDURES

- The starting platform will be Scientific Linux 5 x86_64 with the following enabled YUM repositories: SL5 X86_64, EPEL 5 x86_64, glite 3.2 X86_64.
- A new ETICS platform and Worker Node is available: "sl5_x86_64_gcc412EPEL" which will have all the RPMs required to build the whole EMI distribution already installed in the VM image. Full RPM list: EMI_SL5_x86_64_EPEL_rpmlist.txt [R51].
- A new EMI project is available where all the EMI software will be built. All the EMI configurations required to build will be part of this project.
- Configurations already existing in other parts of the system are cloned and moved in the EMI project, upon request.

- PTs which are interested in re-organizing subsystems or components are given the opportunity to change their software structure in ETICS.
- ETICS dependencies are currently not explicitly required for EMI-0 because all the dependencies are already installed in the VM. They will be needed for EMI-1. Adding those dependencies also in EMI-0, means one thing less to do for EMI-1.
- It is not necessary to remove dependencies if a configuration already has them since they will be soon required again.
- No .DEFAULT properties are allowed in component or subsystem configurations. .DEFAULT properties are only present in the project configuration.
- No STATIC dependencies allowed unless requested to the EMT upon emergency and approved.
- The process of continuous integration of the software components part of EMI-0 is monitored by analyzing the results ETICS nightly-builds [R48].

Details and definitions are present in the Build Configuration & Integration Guidelines [R20].

5.3. ETICS-CONFIGURATION SCHEDULE

Taking in account dependencies between different software components, the following schedule was created:

1. VOMS:
   - glite-security-voms-[api|api-c|api-cpp|api-clients]
   - "old" subsystem org.glite.voms
2. CESNETSecurity:
   - gridsite-[shared|apache], glite-security-gsoap-plugin, glite-security-gss.
   - "old" subsystems: org.glite.security, org.gridsite
3. gLiteSecurity:
   - glite-security-[lcas|lcmaps] glite-security-[trustmanager|util-java]
   - "old" subsystem org.glite.security
4. gLiteInformationSystem:
   - bdii, glue-schema, glite-info-[templates,generic], glite-info-provider-[release|service|ldap]
   - "old" subsystems org.glite.bdii, org.glite.info
5. DataManagement:
   - "old" subsystem org.glite.lcg, org.glite.data, org.glite.lcgdm
6. gLiteL&B, DGAS, APEL, ARGUS
7. gLiteJobManagement (CREAM+WMS), StoRM

A dedicated EMT meeting was held (27.09.2010) for discussing the timeline of the schedule:

Task	Sumamry	Due Date

1), 2), 3), 4)	in parallel + UNICORE	Sept 28, 2010
dCache	start as soon as 4) is ready	October 1, 2010
5), 6)	in parallel, start as soon as (1),2), 3), 4)) are ready	Sept 30, 2010
ARC	Start as soon as 1)& 5) are ready	October 1, 2010
7)	start as soon as (5), 6)) are ready	October 1, 2010

Other weekly EMT phone-conferences are organized, where issues regarding EMI releases are discussed, with emphasis on the EMI-0 release preparation. The Savannah project management tool [R49] is used for tracking EMT actions and EMI-0 tasks.
The process of continuous integration of the software components part of EMI-0 is monitored by analyzing the results of ETICS nightly-builds [R48]

EMI-0 milestone was declared closed at the end of December, 2010, with a delay caused mainly by the new way of managing the external dependencies, the need to adapt certain components to use packages with the versions available from the OS/EPEL repositories, like globus and gsoap, to improve conformance with EMI (Fedora/Debian) packaging guidelines.

6. CONCLUSIONS

This document presents the EMI release process and the initial EMI Release plans designed taking in consideration the initial set of requirements that has been inherited by EMI from past or present projects, like EGEE, KnowARC, NorduGrid, DEISA, OSG and others, and further refined during the project proposal preparation through extensive contacts with user communities and other project proposals coordinators, in particular EGI-InSPIRE and WLCG..

The main goal of the release plans is to ensure that users communities receive certified software releases of middleware services and components according to agreed release policies and quality of service attributes.

Release Plan and Schedules may undergo modifications as during the EMI life-time requirements can change, as well as their priorities. Periodic review of both the Release Plan and Release Schedule are intended every 3 months in order to assure that:

- the Release Plan is up to date and that it describes the actual release process;
- the priorities of the project are taken into account and reflected in the scheduled releases.

Once the first EMI public release, EMI-1, will be available, detailed Maintenance Schedules will be defined for minor and revision software components releases, as well as detailed schedules for phasing out existing functionalities.