EUROPEAN MIDDLEWARE INITIATIVE

EUROPEAN

MIDDLEWARE INITIATIVE

CHANGE MANAGEMENT POLICY

Document Version:	**3.1**
Date:	**24.08.2011**
Status:	**APPROVED**

This work is co-funded by the EC EMI project under the FP7 Collaborative Projects Grant Agreement Nr. INFSO-RI-261611.

Table of Contents

Change Management Policy

1. Introduction

This document describes the EMI policy to be followed to manage software changes in EMI software products.

There is a project deliverable describing the Software Maintenance and Support Plan, DSA1.1 [R1]. This policy is kept synchronised with DSA1.1.

2. Change Management Process

2.1. EMI Releases

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 products, 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 product 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 product 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.

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

- **Major Release**: A major release for a product 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 product 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 product includes significant interface or behavior changes that are backwards-compatible with those of the corresponding major release. New minor releases of a product 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.
- **Revision Releases**: A revision release of a product includes changes fixing specific defects found in production and represents the typical kind of release of a product during the lifetime of an EMI major release.
- **Emergency Releases**: An emergency release of a product includes changes fixing only Immediate-priority defects found in production, typically security-related.

The list of current EMI software products is in section 4 of DNA1.3.2 - Technical Development Plan [R2]

2.2. Release Tasks

Release tasks track new versions of EMI software products. A release task must be opened for each supported EMI major release and platform. Release tasks contain general information about the new version of the product and the list of changes that are introduced. Changes must be tracked in RfCs or development tasks. For more details about changes check section 2.3.

2.2.1. Release Task Tracking

Release Tasks are tracked in the EMI Release Tracker [R3] in Savannah and they are created by the Release Manager upon request from the PTs. PTs must prepare the list of changes they want to include in the release task so that they are approved in the EMT. See section 2.3 for more details.

Release tasks should contain the following information:

- `Unique identifier`: This is automatically created by Savannah when a new task is generated.
- `Should Start On`: Ignore this field within the EMI context. It's a field that always appears in a Savannah task template. PTs can use for it internal purposes if they want to.
- `Should be Finished On`: The date by which the new product version should be released. This field should be defined only once and can't be modified.
- `Postponed Until`: In case a task needs to rescheduled, it's the new date by which the new product version should be released.
- `Category`: Type of tracker entry. Values can be `Component` or `Release`. `Component` refers to tasks tracking a single product version. `Release` refers to a set of new product versions. `Release` tasks are only used by the release manager to help organise the release work.
- `Technical Area`: Lists one or more of the four EMI technical areas that are relevant to the product/release, that is: Compute Area, Data Area, Infrastructure Area and Security Area.
- `UMD Capability`: It specifies one or more of the UMD capabilities provided by the product/release. For more information please check the UMD Roadmap [R4].
- `Supported Platform`: Platform supported by the new product version. Note that one release task should be opened for each EMI major release and supported platform.
- `Priority`: `Normal` unless it's an emergency release. In that case it should be set to `High`.
- `Status`: see next section.
- `Testbed Result`: Result of the deployment of the new product version in the EMI Testbed. The values of this field are `PASS/FAIL` and it's defined by the Testbed Manager.
- `Assigned to`: savannah user name of the PT responsible person for the product. If it's a release, it's assigned to he release manager.
- `Open/Closed`: field that tracks whether the task is open and closed. It's automatically managed by Savannah.
- `Discussion Lock`: ignore this field. It's a field that is always set to `unlocked` and it's defined by the tracker manager.
- `Release`: EMI major release.
- `Name`: Name of the product or release.
- `Component Version`: version of the product/release.
- `List of elements`: A product/release can include several `elements` that are listed in the DNA1.3.2 - Technical Development Plan [R2].
- `List of RfCs`: links to the corresponding items in the different RfCs tracking systems describing which defects/new features are fixed in this release. For development tasks use the `Dependencies` section as explained below.
- `Package list`: list of packages affected by the change and developed by the PT who owns the product. All supported formats must be listed. (i.e. source and binary rpms and tarballs for SL5). In order to automate the QC verification, the following syntax has to be taken into account:
    - ♦ Use # to differentiate real packages from comments in the field.
    - ♦ Package names possible syntax:

◊ Complete package name: i.e.

```
sherpa-1.0.0-1.sl5.x86_64.rpm
sherpa-1.0.0-1.tar.gz
sherpa-1.0.0-1.sl5.src.rpm
sherpa-1.0.0-1.src.tar.gz
```

◊ Complete URL to the package.

- `Documentation`: links to the relevant documentation. Please, check the Documentation Policy [R5] to know which documents are mandatory and must be provided.
- `Component Release Notes`: Check the Documentation Policy [R5] to know what must be provided.
- `License`: link to the file describing the license under which the product is released.
- `Extended Release Notes`: link to the web page, if any, where an extended and more detailed version of the release notes can be found.
- `Test Plan Link`: link to the test plan used to test the product.
- `Dependencies`: Use the `Dependencies` section of the release task to attach any development tasks [R7] implemented in the new product version.

Although release Tasks are created in Savannah by the release manager, PTs must fill in the remaining fields as described in this section.

2.2.2. Release Task state transition diagram

The following diagram represents the set of states in the life of a release task:

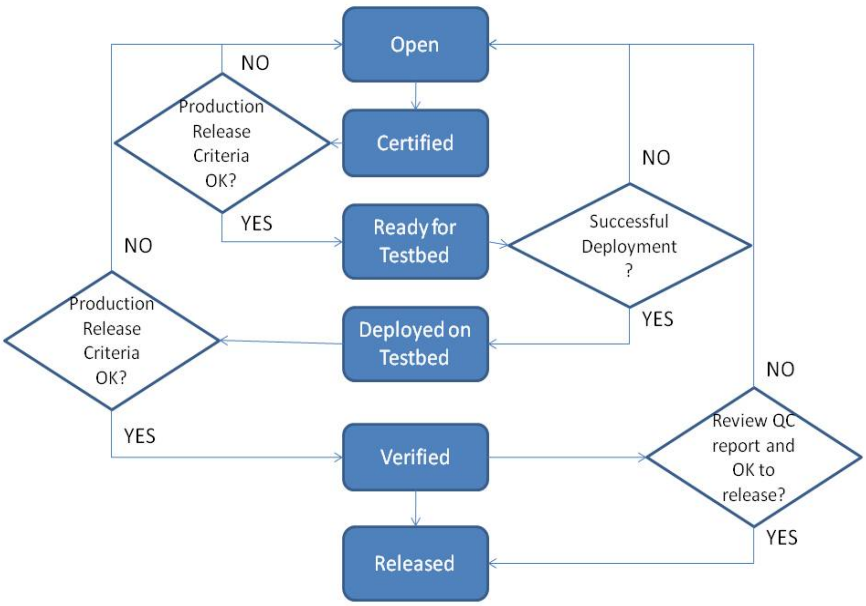

Figure 1 - Release Task States

- **Open**: the Release manager is responsible for creating the release tasks in Savannah to track the different scheduled releases. This should be done with the assistance of the JRA1 leader making sure the different development plans are fully covered in the release schedule. Check the Release Management Policy for more details on this. All release tasks 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 release tasks will be created in Savannah at any time by the Release manager. At this moment, the Release manager together with the PTs, defines the `Should be Finished on` field

and some other mandatory fields.

- **Open to Certified**: the PT moves the release task to `Certified` once the development, testing and certification of the new product version has finished. Before this transition happens, the PT has to make sure all the fields in the release task described above are properly filled in and a complete certification report has been attched to the tracker entry.
- **Certified**: the work of the PT has finished at this stage. This stage prompts the QC task force to evaluate the release task from the QA point of view.
- **Certified to Ready for Testbed**: The QC task force evaluates the release task. The QC task force includes the result of its evaluation in a verification report attached to the release task. At this stage, the Production Release Criteria [R6] is checked. If all mandatory checks are OK or only minor things are missing, the QC task force informs the PT to provide them and moves the task to `Ready for Testbed`.
- **Certified to Open**: The QC task force evaluates the release task. The QC task force includes the result of its evaluation in a verification report attached to the release task. At this stage, the Production Release Criteria [R6] is checked. If some of the mandatory checks fail, the QC task force puts the task back to `Open` informing the PT.
- **Ready for Testbed** : This stage prompts the Release Manager to prepare the repositories so that the Testbed Manager can deploy the new product version in the EMI Testbed. The communication between the Release Manager and the Testbed Manager is done using GGUS.
- **Ready for Testbed to Deployed on Testbed**: The Testbed Manager moves the task to `Deployed on Testbed` once he finishes deploying and testing the new packages in the EMI Testbed. If the testing is successsful he also changes the field `Testbed Result` to `PASS`; Otherwise he changes the `Testbed Result` to `FAIL`.
- **Ready for Testbed to Open**: The Testbed Manager moves the task to `Open` if the deployment of the new product version fails.
- **Deployed on Testbed**: This stage triggers once more the QC task force to verify that everything is correct from the QA point of view. If minor things were missing, they are checked at this step.
- **Deployed on Testbed to Open**: Even if mandatory verification criteria is checked by QC before, it could be the case that at this point some of the mandatory checks still fail. In this case, the QC task force puts the task back to `Open` informing the PT.
- **Deployed on Testbed to Verified**: The QC task force moves the release task to `Verified` after doing the evaluation of the information contained in the release task. If needed, because minor things were missing, the QC task force includes once more the result of its evaluation in another verification report attached to the release task.
- **Verified**: the work of the QC task force has finished at this stage. This stage prompts the release manager to continue the preparation of the release.
- **Verified** to **Released**: The release manager copies the packages in the EMI production repository and prepares the release pages.
- **Released**: The new product version is available in the EMI production repository and the release pages are now online. This state automatically closes the Savannah task. **Note that at this point no modifications should be added in the tasks, not even comments.**

2.3 Changes

Changes can be tracked in both RfCs and development tasks. In both cases, changes should be linked to the corresponding release task as explained in the previous section.

2.3.1. Request for Change (RfC)

RfCs are tracked in different internal trackers selected by PTs. RfCs are motivated by:

- GGUS tickets where users report incidents or make requests.
- PTs when detecting defects that have been found internally or when introducing minor unplanned improvements.

In order to have a common view of the status of all RfCs, XML files exporting information from the different trackers are generated and processed by the SA2.3 task in order to provide a unique report. This report summarises the status of Immediate and High priority RfCs and it is produced every week for the EMT as explained in the Release Management Policy [R8]. For more information about the different trackers and the XML files, please check the Appendix on EMI Tracker Mappings.

An RfC should be evaluated as soon as possible by the PTs to accept it or reject it. A period of two weeks has been defined to carry out this evaluation. RfCs must be then approved by the EMT before they can be included in a release task.

If the same RfC is going to be applied to different EMI major releases, then an RfC is created for each EMI major release.

2.3.1.1. RfC Tracking

A set of fields have been defined to provide information about an RfC. PTs must define these fields in the corresponding tracking tools. When this is not possible, a XML file can be generated containing the needed fields. This is described in the Metrics twiki page.

The mandatory fields are:

- **Unique identifier/URL**: a URL pointing to the RfC. The URL acts as a unique identifier for the RfC.
- **Associated GGUS ticket**: If applicable, one or more URLs pointing to GGUS tickets that have caused the opening of this RfC.
- **Affected Product**: According to the list of products defined in DNA1.3.2 - Technical Development Plan [R2].
- **Affected EMI major release**: EMI major release affected by the RfC
- **Affected Platforms**: Whether the RfC is platform-specific and, if so, which are the affected platforms. Possible values are `SL 5, SL 6, Debian 6, All Linux, All, Noarch, Other`.
- **Priority** of the change. The priority of the change is based on severity, impact, urgency and cost.
    - **Immediate**: The RfC needs to be addressed as soon as possible, in all affected EMI major releases. A release containing immediate- priority changes can contain only immediate-priority changes. Multiple immediate-priority changes can be included in the same release, provided that any change does not delay the release significantly.
    - **High**: The RfC will be addressed in a next planned release of the affected product, in all affected EMI major releases.
    - **Medium**: The RfC will be addressed in the release of the affected product that will be shipped with the next EMI major release. If Medium priority RfCs are ready by the time High priority RfCs are going to be released, they can also be released together as part of an update to an existing EMI major release.
    - **Low**: There is no target date for addressing the RfC.
- **Defect vs Feature**. To differenciate between software bugs and new features.
- **Status** of the change. It shows in which stage of the Change Management Process the RfC is. See the Change Status section below.
- **Detection Area**: The context in which the version of the product affected by the change is available. Possible values are:
    - `Production`: The RfC is opened after feedback received from the users who have used the product in a production environment.
    - `Testing`: The RfC is opened after feedback received from the team of people testing the product.
    - `Development`: The RfC is opened after feedback received from the team of people developing the product.
- **Target EMI major release**: The target EMI major release where the RfC will be available.

2.3.1.2. RfC state transition diagram

The following diagram represents the minimum set of states that should be present in any of the tracking tools chosen by the PTs:

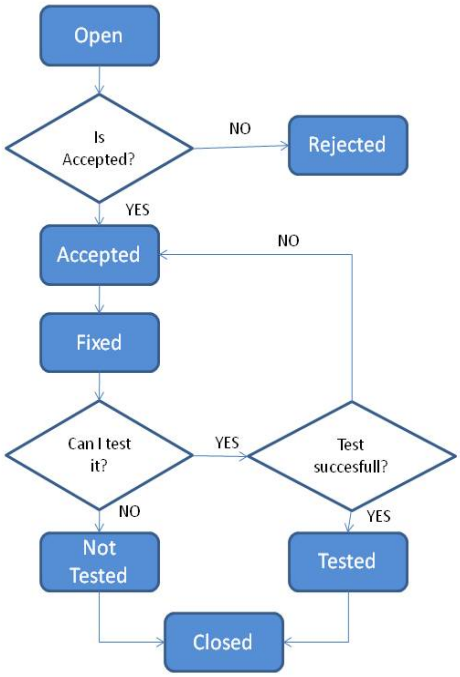

Figure 2 - RfC states

- **Open**: The RfC is opened after all the necessary clarifications have been made. This may include discussions in a GGUS ticket to understand if there's an actual problem, internal discussions within the PT after a defect has been found or after an improvement has been proposed; etc.
- **Open to Accepted**: The clarification is complete and the RfC is accepted to be implemented.
- **Open to Rejected**: The clarification is complete and it's been decided in the end not to implement the RfC.
- **Accepted**: The RfC is ready to be implemented by the PT. This state triggers the implementation of the RfC within the PT.
- **Rejected**: The RfC won't be implemented and the PT should explain why it has been rejected.
- **Accepted to Fixed**: The implementation of the RfC has finished, that is, the code has been committed.
- **Fixed**: The code is committed. Once the packages are ready within the PT, testing of the new release can start, including the testing of all RfC within the release.
- **Fixed to Not Tested**: The PT is not able to test the RfC.
- **Not Tested**: The RfC can't be tested and the PT should explain why.
- **Fixed to Tested**: The PT tests the RfC by running the relevant tests. If the tests are succesful, the RfC can be moved to `Tested`.
- **Tested**: The RfC has been succesfully tested.
- **Fixed to Accepted**: The PT tests the RfC but the tests fail. This means the new feature/bug hasn't been properly fixed.
- **Tested to Closed** and **Not Tested to Closed**: Once the corresponding task where the RfC is included is released to the EMI production repository, the PT closes the RfC.
- **Closed**: The RfC is now available in the EMI production repository.

2.3.2. Development Tasks

Development tasks are tracked in the Development Tracker [R7]. They are maintained by the PTB and assigned to the different PTs. They track:

- High level user requirements.
- Technical objectives defined in the technical area work plans.

PTs addressing a development task should just include a reference to it in the corresponding release task.

2.4. Roles

- Change Advisory Board: Although the priority of an RfC can be suggested by the responsible PT, oficially it is the PTB with the support of the SA1 activity leader who should assess the RfCs and determine their priority and their association with the EMI major release(s) where they will be implemented. The PTB can also delegate the decisions concerning corrective and adaptive maintenance to the EMT.
- Change Manager: i.e. following the process of controlling the lifecycle of approved changes, is taken either by the SA1 Maintenance task leader or by the JRA1 leader, depending on whether that RfC is going to be applied to a component release to be delivered within an existing EMI major release or within the next one.

3. Contacts

EMI SA2

4. Table of References

Reference	URL
R1	DSA1.1. Software Maintenance and Support Plan https://twiki.cern.ch/twiki/bin/view/EMI/DeliverableDSA11
R2	DNA1.3.2 - Technical Development Plan https://twiki.cern.ch/twiki/pub/EMI/DeliverableDNA132
R3	EMI Release Tracker https://savannah.cern.ch/task/?group=emi-releases
R4	UMD Roadmap https://documents.egi.eu/public/ShowDocument?docid=272
R5	EMI Documentation Policy https://twiki.cern.ch/twiki/bin/view/EMI/EMISa2DocumentationPolicy
R6	EMI Production Release criteria https://twiki.cern.ch/twiki/bin/view/EMI/ProductionReleaseCriteria
R7	EMI Development tracker https://savannah.cern.ch/task/?group=emi-dev
R8	EMI Release Management Policy https://twiki.cern.ch/twiki/bin/view/EMI/EmiSa2ReleaseManagementPolicy
R9	EMI Metrics https://twiki.cern.ch/twiki/bin/view/EMI/EmiSa2MetricsGuidelines
R10	EMI Persistent Bug Tracking Collection https://twiki.cern.ch/twiki/bin/view/EMI/EmiSa2MetricsGuidelines#Persistent_Bug_Tracking_Collecti

5. Logbook

v3.1 (Approved on 24.08.2011)

- *23.08.2011*:
  - Differentiate between two type of changes: RfCs and development tasks. Added more details about the development tasks.
  - Added more information about the XML files.
  - Added information about RfCs need to be approved by the EMT.
  - Reorganise `EMI releases` section to include there all the overview on EMI major releases and EMI product releases.
- *22.08.2011*:
  - Rename `Testbed Deployed` to `Deployed on Testbed`. Update release task diagram and add transition from `Deployed on Testbed` to `Open`.
  - Added information about the Development tasks and that they should be attached to the release task.
- *08.08.2011*:
  - Changes agreed with Release Manager, Testbed and QC teams:
    - New state `Ready for Testbed`. Definition and Diagram updated.
    - New field `Supported Platform`.
    - Update `Package List` definition. Indicate that all formats are needed. Give examples of the required syntax needed by the QA Dashboard.
- *13.07.2011*
  - Changes agreed with Release Manager, Testbed and QC teams:
    - New Field `Postponed Until`.
    - Specified that `Should be Finished On` can't be modified once it's defined.
    - New Field `Testbed Result`.
    - New State `Testbed Deployed`.
    - Clarify that no modifications can be done to Released/Closed tasks.
    - Change `component release` (CR) with `release task` which is a more clear and appropriate term.
  - Added example values for `Affected Platforms` as requested by SA2.4.
  - Added Table of References.

v3.0 (Approved on 28.06.2011)

- *28th June 2001*:
  - Applied more feedback from ARC and feedback from Technical Director.
- *27th June 2011*:
  - Applied feedback from ARC.
- *14th June 2011*:
  - Replace `component` with `product` where applicable to be aligned with DNA1.3.2.
  - Remove the need to specify the URL in the package list of a release task.
- *10th June 2011*:
  - Align RfC fields with DSA1.1 deliverable: added GGUS tickets, Affected platforms, Affected products to point to DNA1.3.2, target EMI major release.
  - Clarity one RfC per major release.
- *9th June 2011*: Added new twiki to include tracker mapping tables.

v2.0 (Approved on 28.03.2011)

- *16th March 2011*: Added more specific information on the Documentation field of the CR task.
- *10th March 2011*: After discussing with QC, some improvements are done in the RfC state and transition definitions. Use `tested` instead of `certified` to be aligned with
- *3rd March 2011*: All definitions are now clear.
- *1st March 2011*: Added changes provided by Technical director (definitions). Some clarifications still pending. Mail sent to Technical Director.
- *23rd Feb 2011*: Changes requested by Technical Director:
  - Create new fields in the CR tracker: `UMD capability`, `Technical Area`, `List of elements`.
  - Rename the following fields: `Component version` to `Version` and `Component name` to `Name`.
  - Change the meaning of `Category` to only `release` and `component`.
- *18th Feb 2011*: Changes requested by Technical Director:
  - Create new fields in CR tracker: `Test Plan Link`, `License`, `Extended Release Notes`.

the meaning of this terms within the project.

v1.0 (Approved on 13.12.2010)

- Version 1.0 ready on 17.12.2010. To be announced on EMT 20.12.2010.

Appendix: EMI Tracker Mappings

This is an appendix to the Change Management Policy. It presents the mappings of the different tracking tools used by EMI product teams to track RfCs. It shows the mapping of the required field according to the Change Management Policy to the corresponding field in the PT tracker tool.

1. RfC field mapping

The following table presents the mapping of the different EMI trackers to the mandatory RfC fields:

RfC Field	Definition	ARC Bugzilla	dCache RT	gLite Savannah	UNICORE SourceForge	StoRM Redmine
Unique Identifier/URL	a URL pointing to a description of the RfC. If the RfC concerns a security vulnerability the URL should point to a private page. The URL acts also as a unique identifier for the RfC.	✅	✅	✅	✅	✅
Associated GGUS tickets	If applicable, one or more URLs pointing to GGUS tickets that have caused the opening of this RfC	✅ `URL`	✅ `GGUS_ID`	✅ `GGUS reference URL`	⚠ GGUS URL added in the `Details` field as free text.	✅ `Associated GGUS ticket`
Affected Product	EMI Product List as specified in DNA1.3.2	✅ `Product`	✅ `Product`	⚠ `Category` field exists but not sure it matches DNA1.3.2	⚠ `Category` field exists but doesn't match DNA1.3.2	✅ `Affected Product`
Affected EMI Major release	EMI major release affected by the RfC	✅ It can be mapped to an ARC release since there's 1:1 correspondence to EMI releases	✅ `EMI`	✅ `Baseline Release`	⚠	✅ `Affected EMI major release`
Affected Platforms	Whether the RfC is platform-specific and, if so, which are the affected platforms	⚠ `Platform` field exists but no multiple values although it can be generic like `Linux`	⚠ Since this is a Java project all platforms are supported	⚠ `Architecture` and `OS` fields exists but only one value can be specified	⚠	⚠ Affected Platforms= field exists but only one value can be specified
Priority	Immediate, High, Medium or Low	✅ 1 (Immediate) 2 (High) 3 (Medium) 4 5 (Low)	✅ `Priority` numerical field	✅ Immediate, High, Medium, Low	✅ 1 2 (Low) 3 4 5 (Medium) 6 7 8 (High) 9 (immediate)	✅ Immediate, High, Normal, Low

Defect vs Feature	Whether the RfC is to fix a defect or to introduce a new feature.	☑ One severity level is `feature request`	☑ Two different `Queues`, one for defect and one for feature.	☑ `Severity` field exist but not sure PTs use it properly to differentiate defect and features.	☑ Defects are in the `Bug` tracker, New features are in the `Enhancements and Improvements` and `Feature Requests` tracker	☑ `Bug` vs `Feature`
Detection Area	The context in which the version of the component affected by the change is available. Possible values are `production`, `testing` and `development`	☑ Implied by release version. `Development` has a special release tag CVS, and `Testing` are release candidate versions. Everything else is `Production`	⚠ One possible value only: `Development`	☑ `Bug Detection Area`	⚠	☑
Target EMI major release	The target EMI major release where the RfC will be available.	☑ `ARC release` field it can be mapped to EMI release 1:1	☑ field can be mapped to EMI release 1:1	☑ `Release`	⚠	☑

⚠ For the mandatory fields that are not present in the trackers by default, middlewares must provide the missing information in the XML file needed by SA2.3 to calculate metrics.

2. RfC states mapping

The following table presents the mapping of the different EMI trackers to the mandatory RfC states.

State	ARC State Description	dCache State Description	gLite State Description	UNICORE	StoRM State Description
Open	☑ Unconfirmed New Reopened	☑ EMI Open	☑ Open	☑ Open	☑ New
Accepted	☑ Assigned	☑ Accepted	☑ Accepted	☑ Accepted	☑ In progress
Fixed	☑ Resolved/Fixed	☑ EMI Resolved	☑ Integration Candidate	☑ Fixed	☑ Resolved
Tested/Not Tested	☑ Verified/Fixed	☑ EMI Certified/Not Certified	☑ Fix Certified/Fix not Certified	⚠	☑ Tested/Not Tested
Closed	☑ Closed (after released to production)	☑ EMI Closed	☑ Closed (after released to production)	⚠ Closed (after fix commited to VCS)	☑ Closed
Resolution state	☑ Resolved/Fixed Resolved/Invalid Resolved/Wontfix Resolved/Duplicate	☑ Rejected	☑ Duplicate Won't fix Invalid Unreproducible	☑ Fixed Invalid Rejected	☑ Rejected Closed (Positive State)

	Resolved/Worksforme		Obsolete		

For the states that are not present in the trackers by default, middlewares must provide the missing information in the XML file needed by SA2.3 to calculate metrics.

3. SA2.3 XML mapping

In order to calculate metrics, SA2.3 needs to gather information from the different trackers. This is done exporting the tracker information to an XML file that follows a schema defined by SA2.3. The schema represents the required RfC fields defined in the Change Management Policy.

The following table presents the mapping of the different XML files with the existing schema defined by SA2.3 in order to properly calculate metrics:

RfC field	XML Element	ARC	dCache	gLite	UNICORE	StoRM	Comments
Unique Identifier/URL	`UID`	☑ URL	☑ URL	☑ URL	☑ URL	NA	It should be a URL not only an identifier
Associated GGUS tickets	`GGUS ticket`	UNSET	UNSET	☑	UNSET	NA	
Affected Product	`Component`	⚠ Product names do not follow DNA1.3.2	☑	☑	⚠ Product names do not follow DNA1.3.2	NA	
Affected EMI Major release	`AffectedEMIMajorRelease`	UNSET	☑	☑	☑	NA	
Affected Platforms	`AffectedPlatforms`	☑	☑	☑	☑	NA	
Priority	`Priority`	☑	☑	☑	☑	NA	
Defect vs Feature	`Severity`	☑	☑	☑	☑	NA	
Detection Area	`DetectionArea`	☑	☑	☑	☑	NA	
Target EMI major release	`TargetEMIMajorRelease`	UNSET	☑	☑	-	-	
Status	`Status` and `StatusChangeTimestamps`	☑	☑	☑	☑	NA	

To be followed up:

- UNICORE reports that it will be difficult for them to differentiate between `Target EMI major release` and `Affected EMI Major release`. If this is really needed we have to understand with them how they can provide this information.

4. Fields under discussion

Proposed RfC Field	Definition	ARC Bugzilla	dCache RT	gLite Savannah	UNICORE SourceForge	StoRM Redmine
Regression Test	Field indicating whether the RfC has an associated Regression Test	☑ OK to add in XML file	☑ OK to add in tracker	☑ OK to add in tracker	☑ OK to add in XML file	☑ OK to add in tracker

Comment from dCache: Regression tests will be there for all major features that EMI releases will have starting from EMI 2. For EMI 1 it won't be provided.