Consolidation and development roadmap of the EMI middleware

B Kónya1, C Aifitimiei2, M Cecchi3, L Field4, P Fuhrmann5, J K Nilsen6 and J White7

1 Institute of Physics, Lund University, Box 118, 22100 Lund, Sweden
2 INFN, via Marzolo 8, I-35131 Padova, Italy
3 INFN-CNAF, viale Berti Pichat 6/2, Bologna, Italy
4 CERN, CH-1211 Genève 23, Switzerland
5 DESY, Notkestrasse 85, D-22607 Hamburg, Germany
6 Dept. of Physics, University of Oslo, P. b. 1048 Blindern, N-0316 Oslo, Norway
7 Helsinki Institute of Physics, Helsinki, Finland

E-mail: balazs.konya@hep.lu.se

Abstract. Scientific research communities have benefited recently from the increasing availability of computing and data infrastructures with unprecedented capabilities for large scale distributed initiatives. These infrastructures are largely defined and enabled by the middleware they deploy. One of the major issues in the current usage of research infrastructures is the need to use similar but often incompatible middleware solutions. The European Middleware Initiative (EMI) is a collaboration of the major European middleware providers ARC, dCache, gLite and UNICORE. EMI aims to: deliver a consolidated set of middleware components for deployment in EGI, PRACE and other Distributed Computing Infrastructures; extend the interoperability between grids and other computing infrastructures; strengthen the reliability of the services; establish a sustainable model to maintain and evolve the middleware; fulfil the requirements of the user communities. This paper presents the consolidation and development objectives of the EMI software stack covering the last two years. The EMI development roadmap is introduced along the four technical areas of compute, data, security and infrastructure. The compute area plan focuses on consolidation of standards and agreements through a unified interface for job submission and management, a common format for accounting, the wide adoption of GLUE schema version 2.0 and the provision of a common framework for the execution of parallel jobs. The security area is working towards a unified security model and lowering the barriers to Grid usage by allowing users to gain access with their own credentials. The data area is focusing on implementing standards to ensure interoperability with other grids and industry components and to reuse already existing clients in operating systems and open source distributions. One of the highlights of the infrastructure area is the consolidation of the information system services via the creation of a common information backbone.

1. Introduction
The European Middleware Initiative (EMI) [1] is a close collaboration of four major middleware providers, ARC [2], gLite [3], dCache [4] and UNICORE [5]. It aims to deliver a consolidated set of middleware components for deployment in EGI [6] and other distributed computing infrastructures, 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, harmonize and evolve the middleware, ensuring it responds effectively to the requirements of the scientific communities relying on it. The broader landscape in which EMI operates is shown in figure 1 below.

![Figure 1](image-url). The middleware layer capabilities as part of the broader distributed computing infrastructure landscape. EMI products are delivering solutions in the middleware layer.

The growing availability and use of compute and data infrastructures now requires their transformation into a professionally managed and standardised service. It is of strategic importance for the establishment of permanent sustainable research infrastructures to lower the technological barriers still preventing resource owners and researchers from using grids as a commodity tool in their daily activities. The EMI development roadmap will contribute to the realisation of this vision.

The EMI software development is organized around the following three pillars:

1) Support existing infrastructures by providing re-active and pro-active maintenance for software components used in production. Implement best-practice service-oriented procedures based on clear Service Level Agreements. Work out transition and phase out plans.
2) Harmonize and consolidate the software portfolio originating from the middleware consortia by removing duplications and simplifying usage and maintenance. The EMI software stack must be consolidated and streamlined by removing unnecessary duplication, replacing proprietary technologies with off-the-shelf and community supported technologies wherever possible. The harmonization should be carried out via the adoption of either standard interfaces from well-established international collaborations or interfaces defined via EMI agreements reflecting de-facto standards used by the majority of implementations.

3) Evolve the middleware by addressing the requirements of the growing infrastructures as they become more stable and pervasive. The focus is more on hardening the reliability of existing services, evolving their operational capabilities, implementing new requested features and addressing clear and present needs, rather than producing new prototypal technology to be deployed in a few years’ time. The development preferably should be based on existing code or off-the-shelf 3rd party solutions, this way avoiding the creation of yet another prototype-level solution.

As the ultimate result of the EMI software development activity, by the end of April 2013, EMI will deliver a high quality consolidated middleware distribution of modular inter-compatible components with unified interfaces offering advanced functionalities that can be swapped depending on what kind of feature set is needed. The EMI-Final software stack will consist of reliable interoperable solutions for the core capabilities needed to operate and manage a distributed computing infrastructure. In particular, EMI will provide services within the compute, data, security and infrastructure functionality areas. The EMI services will form an integrated ecosystem via the common security mechanisms and the information system backbone. On the user-side simplified management of security credentials will hide the complexity of Grid security and considerably lower the entry level barrier. The EMI products, many of them to be available as grid appliances deployable in cloud environments, will benefit from advanced technologies such as messaging and virtualization. An overview of the planned EMI product ecosystem is given in figure 2.

The EMI development roadmap is divided into three phases (years):

1) The first phase of the development is marked as EMI 1. This phase was completed with the Kebnekaise release delivered on 12 May 2011 [7]. During the first EMI development phase important technical agreements, component design and early implementations were delivered in addition to the enormous integration efforts that had been deployed for EMI 1 release preparation. Furthermore, most of the Kebnekaise products come with numerous improvements as a result of individual component evolution.

2) The second development phase, leading to the EMI 2 release [8], completed the work on the four area consolidation plans (data, security, compute and infrastructure) that already had started back in year 1. The second phase constituted the most development intensive period of the EMI project. This was the phase that delivered harmonized solutions based on the first year agreements, products such as the EMI Execution Service Interface implementations or the EMIR service and the CANL security library are highlights of the Mattehorn release. Some Year 2 product developments have not yet become ready for public release and their completion is due during the third year (STS and the EMI datalib are such products).

3) During the third, final phase the work will focus on completing all the open development tasks thus bringing the three-year EMI developments to production level. Not yet released products such as STS and the EMI datalib will be the new product highlights of the third release. Apart from these, the final phase development objectives are mostly targeting hardening of existing EMI features, improving non-functional aspects such as reliability, usability and interoperability. Another important objective is the integration of common libraries and EMI products with the rest of the EMI software portfolio. The broad usage of the EMIR information index service and the migration to the CANL EMI security library are such planned activities. The phase will result with the Monte Bianco release due February 2013.
In what follows, after a brief overview of requirements, the planned and already ongoing development work of the final two phases is presented along the four technical areas of compute, data, security and infrastructure.

**Figure 2.** Overview of the EMI ecosystem with the major EMI products and technologies shown. Server-side products are grouped by the four technical areas (lower blue area) while the client-side offerings are presented together (top brown shape).

### 2. Requirements

The planning of EMI software development has been governed by the project’s main objective to support efficient, reliable operations of EGI and other infrastructures. The initial set of requirements influencing the EMI project setup and had been already provided during the project preparation phase. During the course of the first project year EMI received requirements from its collaborating partners. After necessary pre-filtering, assessment and prioritization the endorsed first year requirements were taken into account during the preparation of the 2\textsuperscript{nd} year technical plan and influenced the developments leading to EMI 2 release.

Naturally, EMI has been continuously receiving requirements during the second project year as well. Our major customers communicate their requests through well-established collaboration channels and procedures [9]. All the received requirements get submitted to the EMI requirement management tool, the EMI Requirement Tracker [10].
As part of the year 3 Roadmap adjustments the EMI technical management analysed all the open requirements available in the EMI requirement tracker. Requirements were assessed one by one to check and discuss the status, EMI 2 release progress and relevance of each of them. Relevant and not yet addressed requirements were grouped by areas and products, and translated into prioritised high level technical objectives. Several similar requirements may have resulted in one objective, while some complex single requirements may have led to a number of inter-connected objectives, such as e.g. implementation of a common information schema in different products.

As a very specific example, the request #26574 “Mandatory variables in configuration files should be clearly identified” received during year 2 can be used to show how it influenced the EMI workplan of year 3: this request got accepted, approved and set as a new high level top priority cross-area technical objective, the “X19 Clear identification of the mandatory configuration variables”, affecting all technical areas.

3. Compute Area developments

This section introduces the development highlights of the EMI compute area, a field that includes various computing elements, a workload manager and the corresponding client libraries and command line tools.

3.1. GLUE2 support

The EMI Computing Elements (ARC CE, gLite CREAM and UNICORE/X) already fully support the publication of local-level resource information expressed according to the GLUE2 OGF proposed recommendation standard [11]. The remaining activity for a complete compute area GLUE2 support is the development of GLUE2 support in the match-making modules and client tools. This concerns the implementation of a new module in the WMS [12], responsible for querying over LDAP a GLUE2 enabled BDII [13] and fetching information in the WMS internal cache. For the record, these modules are called ‘purchasers’ in the WMS architecture. A conversion from LDIF to Classad [14] is performed by the WMS engine and the result stored into an internal cache, the so called Information Supermarket. Both GLUE 1.3 and GLUE2 resources need to be fetched, paying attention that the same endpoint publishing both GLUE versions is not accounted twice.

For what concerns support in the clients through the JDL, the formalism is already capable to handle GLUE2 resource descriptions, for both CEs and SEs. For such definitions to be properly interpreted by the server, the aforementioned support in the match-making and purchaser modules must be finalised.

3.2. Common job submission and management method (EMI-ES)

This development objective refers to the implementation of the EMI Execution Service agreement [15] in all the EMI computing services and their clients, as by its reference specification. The common job management interface is one of the most distinguished developments of the project, and it will allow, also fostered by the migration to the common authentication library in the compute Area components, seamless execution of complex workflows to HPC and HTP environments through a single entry point, i.e. the gLite WMS.

Delegation and authorization aspects will still need to be finalised, especially for what concerns interoperability with ARC/gLite and UNICORE. The final goal is to have each client of each of the three middleware solutions able to send and manage jobs to each different computing service and, conversely, to have all computing services able to accept jobs sent by each different client. In some interesting use-cases that have been discussed with user communities EMI-ES can enable considerable simplification in their application level tool (see figure 3).

At the time of writing, the overall completion status of EMI-ES implementation for each middleware is estimated to be around 80%. Parallel to the ongoing iterative implementation of the EMI-ES specification, already interoperability tests have started, involving all the three middleware services in the possible combinations of clients and servers.
Figure 3. Job management interfaces offered and used by various EMI compute products. Before the EMI-ES agreement, compute services and their clients were relying on middleware-specific communication channels (top scenario). With the definition of EMI-ES, the Y2 release of the compute products come with support for the common EMI-ES job management protocol in addition to the pre-EMI possibilities. The common EMI job management client utilizes this interface (bottom scenario).

3.3. Compute Accounting Record (CAR)
A compute accounting record (CAR) is defined reflecting practical, financial and legal requirements of resource consumption, including CPU time, wall-clock time and memory usage. An agreement, in terms of XML schema definitions, over detailed and aggregated usage records was reached. This was done by addressing the previous Usage Record limitations and by extending accounting records to include VO-aware storage usage accounting. This activity resulted in a description document and two XML schemas, one for each record type. Both were based on existing OGF standards, that has been slightly modified both in syntactical and semantic aspects to allow for extended interoperability for the existing middleware layers and taking into consideration existing grid use cases.

Next step now is to modify the EMI computing elements to make those capable generating usage records in the EMI CAR format. The implementation of the common EMI CAR usage record will enable common presentation and aggregation systems for accounting data retrieved from all EMI components of the compute area.

3.4. Integrated solutions to interface with batch systems
An important compute area development task is to provide the ability for all the EMI computing elements to fully support a set of batch systems. These initial set of target batch systems includes PBS/Torque family, Sun/Oracle/Univa Grid Engine and LSF.

Among these, PBS and LSF are already supported by all the three CEs. SGE was supported in ARC CE and UNICORE computing services but not in CREAM. A year ago CREAM started working
on supporting SGE and by now SGE is officially enabled as one of the supported batch systems in the EMI distribution.

As a recent requirement the need to support of the more and more popular SLURM [16] batch system was received. As of writing SLURM is fully supported by ARC and UNICORE while CREAM will need to add support for it.

For the record, full support of a batch system indicates the ability to implement missing functionality and to perform proactive maintenance on existing functionality in terms of job submission, management and information retrieval - keeping up with new product releases, updates and so on. To be more specific, full support does not only mean the ability to interact with the batch system for job specific operations, but also the ability to query for specific and overall information about all the grid jobs delivered to a batch system (number of queued/running jobs, status, etc.).

3.5. Compute client harmonization

There is a community demand for compute area consolidation plan for the compute clients and APIs. Several agreements have been reached concerning the shape of a high-level compute APIs in C and particularly in Java. A taxonomy on the command line options of the three compute clients has been also produced. What is still being discussed is the need for a unified client, a CLI interface. To start the missing part, an extended survey of existing client products, not only limited to CREAM, ARC, UNICORE and not restricted to EMI products will be performed. This will be used as the basis for identifying potential overlaps. At least an agreement on a common API in the major language bindings is foreseen and encouraged to be feasible.

3.6. Common parallel execution framework

Another convergence task within the EMI compute area is the identification of a common parallel execution framework. In particular, more than simply trying to identify a common back-end, a somewhat difficult and even not necessary operation, it was decided to find an agreement on a common definition of parallel jobs across the three middleware. The idea was to adopt the ParallelEnvironment as defined by the EMI-ES. This also leaves the implementation of the back-end up to each single service, leaving the expected behavior defined by the interface. This proposal was greeted with enthusiastic approval by the involved developers. It required slight adaptations to the definition of the EMI-ES that were promptly done in order to accommodate for different kind of requests of parallel applications, coming from the long experience of MPI-Start [17], that were not initially considered. For what concerns the implementation of this proposal when using MPI-start as a common back-end, some adaptations for ARC RunTimeEnvironments and UNICORE ParallelEnvironment were already done. The rest will automatically come with the implementation of the the EMI-ES ParallelEnvironment with the adaptations that were requested in the meantime to support for hybrid MPI-openMP applications and memory/CPU affinity.

The final realization of the common parallel execution framework depends on the availability of the EMI-ES implementations. Once in each stack the EMI-ES interface is implemented, MPI-Start will be adapted for each stack to be used as ParallelEnvironment. The use of MPI-Start as the back-end will not be mandatory, but it will be ready for any site administrator to use if it fits their resources. In the meantime, MPI-Start was modified in order to make it possible to use it as a UNICORE Execution Environment or ARC Runtime Environment.

3.7. Argus-based common authorization

As an important EMI agreement Argus [18] was selected as the common authorization service for EMI components. From the computing area already every service were integrated with Argus. The last missing computing service to be made interoperable with Argus was the WMS. This integration work was accomplished through the PEP client C APIs. For what concerns as the future work proactive maintenance and intensive and comprehensive integration tests are critical to reach the required production level, especially taking into account that it is an off-important security feature.
3.8. Exclusive node or multi-core allocations

Traditionally, most high energy physics applications have been running their workloads using one core per job. This normally applies to both early- and late-binding jobs. However, these applications have for some time required the possibility to run their jobs on more than one core. This has led in some cases to issues related to memory consumption. These issues can be mitigated if memory is shared across several cores on worker nodes.

Currently few sites are running prototypes to support multi-core jobs through dedicated “whole-node” queues, i.e. entry points where experiments would send jobs to, with the assurance that the jobs would run on hardware exclusively allocated to them. This solution has a few advantages: once a job has landed on a given node, it can simply look up how many cores there are, how much memory, etc.; the job will be able to manage the resources simply using the standard OS tools, without having to care about other jobs being running on the same node. However, from a site perspective, this solution often leads to resource partitioning: some resources are allocated to one or more “whole-node” queues, and cannot be used (without greatly sacrificing resource exploitation) by traditional single-core jobs. This partitioning may also be aggravated by the fact that systems have an increasingly number of CPU cores; therefore, a single system accounts for a relatively larger share of resources.

Several sites in general, and sites serving more than one experiment in particular, rather prefer dynamic resource sharing making use of the appropriate scheduling mechanisms provided by the site batch systems. It will be anyway a site decision whether to offer a “strict whole node” solution, or rather generic multi-core access, but the middleware must be able to support it. With the rise of multi-/many-core architectures and the experiments’ needs to limit the increased memory requirements that cannot be solved by running single core jobs on such CPUs, the risk with not achieving this task is that a growing part of the computing resources will be misused.

For the above reasons, being far from representing a technical obstacle, it is important that this activity is well tracked and embraced as a coordinated development task for all the EMI computing services.

3.9. Jobs with special characteristics

Sites have used general purpose batch systems (LRMS) for many years to provide highest throughput and fairest allocation of resources between the communities that they serve. Many sites have also expressed interest with scheduling based on resource constraints, particularly storage. More options should be given to schedule resources as they assert that higher throughput could be achieved if jobs requirements are known better to the LRMS scheduler. I/O throughput on sites storage systems has an optimum load, and global throughput can be reduced when a load above a certain threshold is reached. Excessive Storage load could also lead to storage system instability. For example a site may decide that by spreading users’ jobs on different nodes, the start up time is more staggered and so preventing resource contention on a single file. The introduction of pilot jobs has prevented scheduling decisions based on user name. Although user name only gave an indirect interpretation of job load, I/O-bound and CPU-bound tagging intends to better scheduling based on user name, by giving more useful information to the LRMS. These resource constraint tags should be honoured by the CE and passed to the LRMS in an agreed way, allowing sites to customize scheduling if they so desire. Sites that do not want to make use of resource constraint tags will not need to support them; furthermore, sites are encouraged to publish in the information system the tags that apply to a given queue, so that this mechanism will serve user needs other than being only at the advantage of the sites themselves. This will require some synchronisation among producers and consumers about the way the bits of information will have to be advertised and retrieved.

4. Security area developments

This section introduces the development highlights of the EMI security area, a field that includes various security services and libraries.
4.1. Lowering the security credential handling barrier

A key security development in EMI is to make the security credential management more accessible to ordinary users. This is to be achieved by introducing simplified management of security credentials via reducing the complexity of handling certificates and integrating different security mechanisms like Shibboleth [19] and Kerberos [20] across the EMI stack. This development will allow users to use their own authentication system to access a Grid. The goal of this activity is to lower the barrier of accessing distributed computing infrastructures using institutional or federated institutional authentication systems and to enable the usage of EMI components and services with other security infrastructures such as Kerberos or Shibboleth. In order to enable this access, a new security service, the Security Token Service (STS) is needed to translate these external credentials into the X.509 credentials needed by most Grid infrastructures.

The simplified management of credentials is to be fulfilled by the development and deployment of the EMI Security Token Service. The STS implements the service defined by the WS-Trust specification. STS is a Web service that issues security tokens, a collection of claims, for the authenticated clients. As the clients can authenticate to the service using different security token formats, the service can be seen as converting a security token from one format into another. As such the STS is used to bridge different trust domains.

The current plan is that the STS will be implemented on top of the upcoming Shibboleth IdP version 3 and OpenSAML3 implementations. The IdP version 3 will provide support for SOAP binding and delegation: the SAML assertion will be targeted at a service other than the requester. The advantage of reusing the Shibboleth / OpenSAML3 code base is twofold: (1) It allows the reuse of non-trivial Web service libraries; (2) it allows to "leverage" the STS from the beginning against the code base of the most used AAI system in Europe. In addition, this service will exploit the EMI common authentication library. Unfortunately, the release schedule of the Shibboleth/OpenSAML version 3 has slipped by some months.

As the current state of the STS implementation is already capable of issuing the X.509 certificates, the first use-case has been prototyped in PM25. It consists of an issuance of an X.509 certificate based on a security token from another security domain: username and password in this case.

The EMI STS development plan specifically mentions obtaining X.509 based security token from Shibboleth-based AAI federations and Kerberos. The support for incoming and outgoing SAML assertions is scheduled first that will be followed by the support for incoming Kerberos, X.509 and X.509 proxies and outgoing X.509 certificate.

The client-side toolkit used to make the actual integration to clients such as a command line, GUI or third-party portal is scheduled as follows. Each incoming credential format can be supported on the client-side approximately one month after the server-side support is finished.

4.2. Common authentication library (CANL)

The security area consolidation of the EMI products is driven by the definition, implementation and migration over to the common EMI authentication library (CANL). EMI wants to provide common authentication libraries supporting X.509 and optionally, for the future, SAML. The EMI authentication library has been defined. API definition is available for Java, C and C++ [21]. The implementations of the libraries have almost completed and prototype versions of CANL for all the three languages were included into the EMI Matterhorn release [8].

The final step of the consolidation is to migrate EMI products over to the new common libraries. Adoption of the library in UNICORE already started, at first the UNICORE Gateway and UNICORE security libraries are updated to use the new library. The adoption of the C library by other EMI components is expected during the EMI 3 phase. The adoption of the C++ version of the common authentication libraries in the ARC security components will happen as a continuous process along with the implementation. According to preliminary survey all major gLite, UNICORE, ARC and dCache products plan to adopt the common authentication library.
4.3. Proxy handling features

EMI security is built around X509 technology where proxies play a central role. Therefore a considerable part of the security area development is targeting proxies. In particular proxy handling features to address SHA2, default key-size and OCSP requests.

Soon sites will receive certificates and proxies from Certificate Authorities that will be signed with a SHA2 hash rather than the current SHA1 or even MD5. Support for the older hash algorithms will be retained. Default key size for generated proxies is required as this is not universally enforceable now. The Online Certificate Status Protocol (OCSP) is an Internet protocol [22], alternative to static revocation lists, used for obtaining the revocation status of an X.509 certificate.

In order to provide a common solution, all these proxy handling features will be implemented in the Common Authentication libraries (CANL). Subsequent usage of these common libraries by other EMI components should automatically provide the proxy features:

C library: the primary focus of the common authentication library is X.509 and PKI however, there are use cases for which other security mechanisms would be useful. Therefore, the C implementation will be extended to internally support the GSS-API, a standardized API to develop an application independent of any particular security protocol. In order to facilitate application deployment using this library, support will also be provided for the PKCS#11 API. Using this API it will be possible for the library to access cryptographic artifacts (like keys and certificates) that are stored on smart cards and/or specialized repositories. The default key size and SHA2 key hashing algorithm features are already supported in the C version of CANL. The basic support for OCSP is also available as of EMI 2. As mentioned above, a precise and concrete definition of the configuration and usage of OCSP must be provided by our clients before this can be truly considered released and supported.

C++ library: The support for OCSP is almost completed in the C++ version. As OCSP is merely a protocol, there is some work needed on profiling, i.e. how it should work and be configured. The libraries on which the C++ CANL version is based do support the SHA2 hashing algorithm. Once the SHA2 proxies are supported in the C++ version, a proxy generator supporting this feature will be needed. The support in the C++ version of CANL for a default key size for any generated proxy needs to be implemented.

Java library: The Java version already supports a configurable default key size and keys signed with the SHA2 hash algorithm. Logically, the library supports the generation of proxies containing these features. The support for Online Certificate Status Protocol is implemented and details of operation being agreed.

4.4. Encrypted storage layer

Security area is involved in providing a transparent solution for encrypted storage utilizing ordinary EMI Storage Elements. The realization of an encrypted storage within EMI is relatively simple: necessary services to protect data and user identities on a Grid have to be provisioned. These services are requested by user communities that have stringent data protection requirements. EMI offers the pseudo-anonymity (pseudonymity) service and the key storage service (Hydra) as a solution.

While pseudonymity is a certified EMI product, unfortunately Hydra is not yet ready for release in EMI 2 on SL5/64. Necessary development work is related to the understanding the changed behaviour of some external libraries. In particular, on the SL6 platform the changes in the gsoap version are significant and will cause quite some work on the Web Service level.

4.5. Standard CRL handling

This development task is about checking that each service is able to properly import Certificate Revocation Lists from the EMI distribution and about checking the present behaviour with respect to ‘live’ re-loading without service interruption. The task is defined as “verify the standard support of CRL handling of EMI services”.

The Certificate Revocation List (CRL) is a method used to verify that a credential (user or other) received at the Authentication stage has not been revoked by the issuing authority. These CRLs are
generated by the issuing authorities and distributed via a “fetch-crl” tool. The ability for a service to register a CRL update without needing to be restarted is acknowledged. This ability can be easily verified through the use of a functionality test. The Security area services that may be affected by this technical objective are: Argus, Hydra, Pseudonymity and VOMS. All the other area services will also be revised with respect to handling of CRLs. It has to be noted that the migration to the common authentication library will automatically solve this issue most of the cases, as it will be able to transparently deal with CRLs.

5. Infrastructure area developments

This section presents the development highlights from the EMI infrastructure area, a field that covers information system components, service monitoring solutions and accounting probes and publishers.

5.1. EMI Service Registry (EMIR)

Prior to the EMI project there was no common solution to discover ARC, gLite, UNICORE or dCache service instances from a common information source. The middleware stacks were confined into their own information systems. A common information system backbone, a service registry shared by the middleware stacks therefore was identified as the most important missing service, a critical missing component needed for the harmonization and convergence of the EMI products.

Development of a new service, the EMI Registry was proposed. A design document for the EMI Registry was produced [23,24]. The EMI Registry, which enables service discovery, represents a major new product for EMI and as such good progress with this is encouraging. The Service Discovery use case has been identified by the general user community as an important aspect of the Grid information system for which the existing solutions are falling short. A specific system for Service Discovery, the EMI Registry, aims to address the limitations.

Following on from the design document, a development plan was defined with the objective of providing an implementation of the EMIR in time for the EMI 2 release. As of Today a prototype EMIR is available as part of the EMI Matterhorn release. The implementation should be considered experimental. The aim of the release is to gain initial experience for users in order to provide a production quality release in time for EMI 3.

Now that the EMIR prototype is available the next step is the adoption and roll-out of the registry. This phase requires two subtasks to be carried out during the last year of EMI: all EMI service should be provided with a registration module that takes care of sending service information to the EMIR registry. The EMIR development team will ensure that all EMI services will be able to publish their existence to the EMI Registry. The second subtask concerns the modification of the information consumers, the different service discovery clients. Here WMS, ARC and UNICORE products will be adjusted to be able to use EMIR as a service discovery information source.

5.2. Harmonized resource level information (ERIS)

The introduction of the EMI Resource Information Services (ERIS) into the EMI infrastructure area as a common component for obtaining information directly from services is an important major step for harmonization. This development objective consolidates the existing resource-level products and ensures interoperability through the agreement on a common information model and interface.

An expert task force was created to provide a technical proposal on resource-level information service for EMI. The task force considered the options and on balance recommended that the ERIS should provide an LDAPv3 interface to GLUE 2.0 information. Information providers, in the classic sense, will extract information from the under laying Grid service and produce GLUE2 information in the LDIF format. These two together represent the external and internal interfaces for obtaining local information from EMI services.

It is envisioned that this approach will be a minimal-cost solution for existing EMI products and will have a low-impact on existing infrastructures. As of writing all gLite, ARC and dCache services already covered by ERIS since those are capable publishing GLUE2 information via LDAPv3.
interface. The missing part to be developed during the last year is a solution for UNICORE services that currently are not covered by ERIS.

5.3. Accounting consolidation
When it comes to accounting EMI is concerned about producing and transporting accounting records to accounting servers such as APEL [24]. The maintenance and development of the server-side of the accounting ecosystem is not within the scope of EMI.

Both the Compute Accounting Record (CAR) [25] and Storage Accounting Record (StAR) [26] have been defined and agreed within EMI. As a result of the collaboration with EGI the central accounting repository was updated in March 2012 to accept the new CAR messages from clients and summaries from other systems. The existing EMI Accounting Client is being updated to send these new CARs using messaging and it is expected that this will be released as an update to the EMI 2 release. Storage accounting record publishers is to be developed.

An important outcome of the EMI Vision Meeting held in December 2011 was the removal of the DGAS accounting client from the EMI stack. APEL client is now the sole accounting client in the EMI stack. It will be ensured that DGAS accounting server (not an EMI product) will be able to accept information from the EMI Accounting Client via interoperability testing. This decision completes the harmonization in this area.

Now that the majority of the work has already been done it just remains to integrate the different available modules so that the development objective “to provide or adapt the accounting publishers for compute and data Area services to use the common messaging system” can be considered completed.

5.4. Service monitoring
In the initial EMI plans it was envisioned that EMI infrastructure area would investigate the use of messaging technology for service monitoring and even propose a common interface for service monitoring and management that could be adopted by all EMI services. A survey of Grid sites was jointly conducted with EGI to investigate the requirements in this field. The feedback from system administrators is that while there is general agreement that such a solution is on everyone’s wish list, this would only make sense within the wider context of standards in data centers, which is out-of-scope for EMI. In practice smaller more concrete objectives would have greater impact for service monitoring.

The result of consultations with EGI has re-focused the EMI plans with respect to Service Monitoring and Management. In particular, for service monitoring, EGI requested that each service should provide a Nagios probe which can be used to measure the availability. During 2nd EMI year 90% of the service Nagios probes were delivered and released with Matterhorn. The next step is to integrate the new probes into the EGI monitoring infrastructure, a task that requires further close collaboration between EMI and EGI.

6. Data area developments
This section contains the main development tasks from the EMI data area, a field covers storage elements, data movement service, data catalogue and client-side data libraries and utilities.

6.1. EMI datalib: consolidation of the data access libraries
The ARC and gLite middleware are providing data access libraries individually, which have sufficient functionality in common to justify a merge of those libraries. This includes, but is not limited, to storage control (SRM), storage access (e.g. gsiFTP) and information protocols. EMI data has been working on an agreement to merge this functionality in a common EMI data library and a design was agreed upon in January 2012 [27].

Prior to the EMI_datalib, gLite provided two sets of data libraries, the lcg_util library and the GFAL library. The lcg_util library supported high-level file operations with an API closely matching the lcg_util CLI commands (e.g., lcg-cp, lcg-cr and lcg-ls). The GFAL library provided a low-level
POSIX-like interface and an SRM-like interface to support storage resource management operations. Transfer protocols were supported through special GFAL plug-ins. Similarly, ARC offers a single file-based, high-level API for moving data between a local and a remote site, with automatic handling and registering check summing of data. As for the GFAL library, transfer protocols were supported through plug-ins.

The proposed architecture will keep both interfaces, the POSIX-like one (provided by GFAL-2) and the higher-level file-based one (provided by the libarcdata2 library). In addition, a transfer interface will be made available in GFAL-2 to handle initiation and monitoring of 3rd party transfers. In the proposed architecture, libarcdata2 will use the POSIX-like interface of GFAL-2 through a GFAL-plug-in. For platforms where this plug-in is available no other plug-ins will be needed for libarcdata2, while 3rd party clients, requiring POSIX-based byte-wise data access, can use the GFAL-2 library directly, file based, data moving clients such as the lcg_util and ARC CLIs and ARC CE will use the libarcdata2 file-based interface. Additionally, FTS3 and parts of the lcg_utils and ARC CLIs will use the 3rd party transfer library. Hence, the lcg_util library can be removed. A Python library will be created to replace the needed functionality of the lcg_util Python API. The main benefits of the proposed design is that there will be only one set of transfer and control plug-ins instead of two and the clear division of components, with one file-based layer and one underlying POSIX layer with the transfer plug-ins, which significantly reduces support efforts.

As of writing the EMI datalib is being implemented with a scheduled release time of September 2012. The main work required, GFAL-2 and the ARC data library plug-ins for GFAL-2 are already implemented at a prototype level, and the GFAL-2 library is released in beta state in the EMI 2 Matterhorn release. Once the prototype is available the affected EMI data area products will be adopted to use the new library. Due to the nature of the design, the ARC CLI will naturally support the EMI datalib. However, the lcg_util CLI needs to be reimplementation to use libarcdata2.

6.2. FTS3: next generation file transfer service

FTS3 was conceived to address a particular set of FTS [28] shortcomings. It was named FTS3 to emphasize the continuity with existing production File Transfer Service (FTS). The next generation FTS addresses:

- **Configuration model**
  - Relax the requirement to configure channels
  - Instead use endpoints, good defaults and adaptation

- **Allow adaptation to the state of the infrastructure.** This information could come from a number of sources including monitoring, history and interrogation of endpoints

- **Protocol support**
  - Make it easier to add support for new transfer protocols by basing the transfer agents on GFAL-2
  - http and xroot are targeted

- **Database support**
  - Decouple from Oracle and add Mysql

- **Transfer monitoring (as in FTS2)**

- **Simplified deployment**

- **Development of FTS3 has followed a process, which schedules a demonstration every 3 weeks of newly added functionality.** The following features have been demonstrated during recent development periods CLI submitting a job
  - New (C++) web-service accepting transfer
  - Transfer added to db via db-independent interface and oracle plug-in

- **CLI can check for “Transfer initiated”**
  - CLI can check for completion

- **CLI can check for “Transfer completed”**
  - Basic configuration of an endpoint
Channel-less configuration actions
 Execution of multiple transfers

The implementation of the FTS3 is currently on-going, the developer are working towards “Prototype One” which will be the first version of the prototype made available to external testing. The main additional functionality with is full support for the SE-centric configuration model and scheduling based on this information. It underlines “message passing” as the main communication backbone and the focus on standard protocols e.g. SRM and http. Prototype 2, expected by the end of 2012, will add the following features:

- Adaptation of transfer parameters based on recent performance
  - An area of research will be evaluating automatically applied incremental changes in order to optimize parameters
- Full authorization model
  - The model is still to be determined with the stakeholders

6.3. Storage accounting

While there are storage systems that have been doing accounting for some years, there has so far been no standardized way to collect data on storage usage in the Grid. Within EMI, none of the storage elements had any formalized accounting methods prior to EMI and deciding on and implementing a common set of accounting records was seen as highly needed. The storage accounting record (StAR) [26] reached an internal agreement in February 2012 and will be implemented for the EMI 3 release. Additionally, it has been taken up in OGF as an informational document and taken as input for the next generation OGF usage record format.

Where as the StAR definition was agreed upon in February 2012, this is a general definition that only provides the allowed fields in a storage record. As StAR in its nature targets storage accounting in general, the main costumers of EMI (WLCG and EGI) need to account for storage usage in the Grid environment. To provide the wanted information, work has been ongoing in collaboration with EGI to define an EGI profile to StAR, which clearly states what information is required and in which format the information should be represented within the EGI community.

Another major question is how the records should be published. While in the case of computing records publishing was already implemented in some way or another prior to EMI, storage accounting is a new undertaking for the EMI storage elements. While for compute records, information is provided by the compute elements and then converted to compute records with accounting sensors, for storage records it made more sense that the storage elements directly create records following the StAR definition and send these to the accounting publishers. The storage elements (dCache, DPM and StoRM) will generate StAR records and publish these to the APEL accounting server using the APEL Secure Stomp Messenger (SSM). Then, if time allows, publishing to DGAS and other accounting servers may be considered.

The design of how to generate and send StAR records into the common messaging infrastructure in order to be processed by external storage accounting service has been completed, architecture and work plan were discussed, approved. The implementation work for DPM and StoRM has just started and is due with EMI 3, while dCache already provides a module generating the Storage Accounting Record and ready to integrate with the APEL-SSM.

6.4. Supporting WebDAV

Recently WebDAV [29] has become a more and more requested protocol. Therefore data area development tasks have been created to support the protocol in the EMI Storage Elements and the LFC file catalogue.

6.4.1. WebDAV protocol for DPM: DPM had already offered an HTTP(s) front-end as an add-on daemon since EMI 1, but it had known issues that limited both its functionality and performance. This module has been rewritten adding support of the additional WebDAV standard (RFC 2518) and basing...
the implementation in two widely used open source tools: Apache2 and mod_dav. In addition, some areas were identified where HTTP as an alternative to GridFTP would fail either to have similar functionality or show similar performance. These are listed below along with the solution followed to make sure HTTP is a complete alternative to GridFTP:

- **Multiple Stream Transfers:** Especially important over the WAN, these are natively supported by the FTP protocol. We rely on the usage of the Content-Range HTTP header to achieve similar functionality.
- **Third party copies:** Also natively supported in the FTP protocol. We rely on the WebDAV COPY method to achieve similar functionality.
- **Credential delegation:** This is the main feature of GridFTP over pure FTP. We rely on the deployment of the gridsite delegation service along our WebDAV endpoint to achieve similar functionality. Especially relevant to achieve third party copy functionality.

The current DPM WebDAV component is production ready and deployed at several sites. It is fully functional, but we are considering additional improvements:

- **PULL instead of PUSH for third party copies:** 3rd party copies are always done in a PUSH mode as of today. To be consistent with the GridFTP features, we will add the additional possibility of doing PULL transfers.
- **Transfer markers in separate connection:** We currently support transfer markers on 3rd party copies, periodic responses containing information on transfer completion. One downside of the current implementation is that the client connection has to be kept alive during the duration of the transfer. We’ll add the possibility of clients requesting this information asynchronously.
- **Improved replica retrieval:** One additional feature of the current implementation is the retrieval on failed file replica access (to a different replica). Currently it relies on semantics fully managed by the server, but that do not cover inaccessible Disk Nodes. We will investigate the usage of the Multiple Choices HTTP redirection reply to overcome this issue.
- **Add ACL management:** ACL management functionality is currently not available via the HTTP/DAV frontend. This is mostly due to the lack of support in clients for RFC 3744, which adds this feature to the DAV standard. We will put effort into evaluation potential clients, selecting the best ones and patching them if necessary to offer this functionality.
- **DPM metadata clients talking to WebDAV backend:** For file metadata management today, DPM users use the custom name server clients, which use an internal protocol to talk to the name server daemon. We will develop a new set of clients offering the same functionality but contacting the WebDAV backend instead - we will rely on deb/rpm alternatives to allow deployment of both flavors in the same setup.

In addition to the items above, we will put a lot of effort into validating our backend against the other storage services of the EMI middleware, to ensure that standards based HTTP access to storage within EMI is fully functional and consistent between implementations.

### 6.4.2. WebDAV support in dCache

The dCache team decided to use the “Milton” libraries [30] as to support the protocol. The Milton library manages the protocol stack and does call-outs to the dCache core system for the various client requests. If the requested file is found in dCache, the Milton door redirects the client to an appropriate data pool. The pools subsequently talk http with the client. All dCache supported protocols, including NFSv4.1 and WebDAV are sharing all dCache core components to provide consistent name space, permission and space management. For WebDAV, gPlazma2 offers X509 certificate and user/password authentication. The WebDAV implementation in dCache is finished, no further work except for possible BUG fixing is planned.

### 6.4.3. WebDAV support in StoRM

At this point in time, StoRM doesn’t provide a WebDAV service yet. However, any WebDAV 3rd party server with sufficient flexibility in configuration can be
integrated with StoRM. A feasibility study and a survey of WebDAV servers that can be integrated into the StoRM system has identified two possible candidates:

- Milton [30] as used by dCache.
- 101tec [31].

The StoRM developers didn’t decide yet which way to go.

6.4.4. WebDAV for LFC. Traditionally access to the LFC has been done using a custom protocol, preventing the usage of any standards based client to browse the catalog. As part of our work to move towards standard protocols and building blocks, we have added a new component exposing the name server metadata via WebDAV. Standard operations like browsing directory contents, renaming files or directories, removing files or any other expected file system operation are now available to LFC users using standard clients - browser or command line tools.

In addition, we also expose the base HTTP standard, allowing GET redirections from the catalog to the different storage services with corresponding replicas registered in the catalog. This offers the user a global access service to all the data registered on a given LFC catalog, using standard HTTP clients and without having to know any details regarding available file replicas.

The sequence of client requests when accessing the LFC using a logical file name (LFN) is the following:

- Client requests access to a file using its logical name
- Client gets (transparently) redirected to a storage server hosting a replica of the file, without having to know about the storage file name being used
- Client gets (again transparently) redirected to a disk server hosting the physical replica of the file referred to in previous step.

The current implementation is production ready, and is available with the EMI 2 release. In addition to the base functionality described above, we have also implemented support for transparent failover over multiple replicas, again relying solely on standard clients. This mechanism is the following:

- Client requests the file directly from the LFC
- Client gets redirected to a first replica of the file, but the HTTP URI includes information on additional replicas to be tried in case of failure
- First replica is unavailable, and client gets redirected to a second replica
- Second replica is unavailable, and client gets redirected to a third replica
- All replicas failed, client gets redirected back to the LFC

The LFC can then provide the client with even more replicas to be tried, or simply fail if there are no more replicas available. Of particular note here is the fact that any standard client will work, as we rely only on the standard HTTP redirections - the knowledge on how to handle the replica list is only on the server side.

As future development these are the areas where we will put additional work:

- Limited number of replicas for retrial: We rely on the URI used in the redirections to keep a list of replicas for retrial. Given the limitation on the length of a URI in some clients (mostly browsers), this creates a limitation on the number of replicas available for retrial and mandates a periodic request to the LFC (to ask for more replicas). We will investigate alternatives to avoid this.
- Multiple Choices in HTTP redirection: The ability to use the Multiple Choices header in a HTTP redirection would remove both the need to keep the retrial replicas list in the URI, and cover the issue with offline storage services. We will investigate both its potential usage and client support.
- Optimize the access to replicas: Currently the clients are redirected to replicas in a random order, without any knowledge on the overall system load being taken into account. We will
investigate the available options (if any) to easily retrieved up to date storage load to introduce a more intelligent replica selection.

- Extend the concepts used here with the LFC (a centralized catalog) to distributed catalogs or federations.
- Continue validation of the transparent redirection from catalog to storage service against all the storage systems that are currently part of the EMI middleware.

6.5. Transparent HTTP access to distributed storage: storage federation

This development objective responds to the need for transparent world-wide data federations composed of heterogeneous storage systems based on standard access. The approach is to enable designing geographically distributed data storage and management systems based on the HTTP and WebDAV protocols, in the context of EMI. This includes specifications that allow applications to access transparently aggregated global views made of multiple File Catalogues and/or Storage Elements. Policies like failover across sites and tolerance to lost files are also taken into account.

The idea of a global storage federation is the following: clients, which can be either an unmodified browser, a WebDAV driver of the operating system or special applications, e.g. the ROOT framework, send their file request to the “federation service”. The service tries to find the most appropriate copy of the requested dataset and returns an http-redirect command to the selected endpoint. The data is finally transferred directly between the client and the endpoint. The federation service is not involved in the data transfer.

The initial design of the federation service comprises of the following services: beside the front-end, providing the protocol engine responsible for communicating with the clients, at least 3 more services need to be implemented. Those services allow plug-ins to be inserted, doing the actual work:

- A mapping service: As the name space at the local sites, providing the data, may differ from the global namespace, a mapping has to be provided. Initially a plug-in to query the experiment catalogues and a simple pattern matching plug-in are envisioned.
- An endpoint matching services: Not all endpoints might be suitable for all type or locations of clients. So there must be a service, being able to select the most appropriate ones. Initially we are planning to only use proximity information for matching. As GeoIP plug-in is already in place. Others, like a plug-in describing the IP topology are planned.
- A name space collection and merging service: This service is in charge of querying the endpoints for the existence of requested data as well as getting listings of requested directories and merging the content, providing the federated view.

The global picture comprehends file catalogues like LFC, their interface to applications and newer concepts like the Dynamic Federations, which operates on the fly, without needing a central persistent catalogue. This kind of system:

- Makes different storage clusters be seen as one.
- Makes global file-based data access seamless.
- Offers mechanisms for user defined mappings between global and local name spaces.
- Is able to optimize the data access using proximity information.
- Is easy to setup/maintain.
- Exhibits high performance.

Regarding implementation status, as of writing (April 2012), an alpha-level component, called UGR (Uniform Generic Redirector) is available, together with a basic set of plug-ins. This was successfully integrated into an Apache2 server, through the DMLite and dav-DMLite libraries, developed by CERN. A demo of the Dynamic Federations that exploits the Global Access features is available. The demo aggregates a DPM instance in Taipei, a dCache instance in Hamburg and an international WebDAV-based Cloud storage provider. Given the wide scope of the development objective, the system’s design is agnostic with respect to the particular storage technologies and
protocols involved. The plan is to develop the missing plug-ins and features and continue with enhancing the core subsystems.

7. Conclusion and outlook

This paper lays down the software development directions for the final years of the EMI project. The roadmap is presented along the four technical areas. The last year of EMI is dedicated to bring to completion development and integration activities. With the final release EMI aims to achieve its technical objective which at a high level can be outlined as delivering a production-level consolidated middleware distribution, consisting of modular components with unified interfaces. The components will cover a wide range of functionalities, providing universal building blocks for infrastructures of various complexity and specialization.

In short, the final year activities will be focused in hardening of existing products, finalizing new ones, especially those common components that were newly developed during the project, and executing the necessary integration, functionality and usability tests. To summarize, the objectives that are considered to be of the top priority in the final year are: completion and delivery of the EMI datalib, FTS3 and the STS product, broad integration with the common EMI products and libraries such as EMIR indexing service and CANL authentication library, and focus on non-functional developments to improve performance, usability and reliability of EMI products.

Acknowledgements

This work was partially funded by the EMI project under European Commission Grant Agreement INFSO-RI-261611.

References

[1] European Middleware Initiative Web site URL http://www.eu-emi.eu
[2] Ellert M et al 2007 Future Gener. Comput. Syst. 23 219-240 ISSN 0167-739X
[3] Laure E et al 2005 Middleware for the next generation Grid infrastructure Proceedings of Computing in High Energy Physics and Nuclear Physics 2004 (publised version from CERN) pp826
[4] dCache Web site URL http://www.dacache.org
[5] Unicore Web site URL http://www.unicore.eu
[6] EGI Web site URL http://www.egi.eu
[7] EMI 1 Kebnekaise Release Web site URL http://www.eu-emi.eu/emi-1-kebnekaise
[8] EMI 2 Matterhorn Release Web site URL http://www.eu-emi.eu/emi-2-matterhorn
[9] EGI-TCB Requirements Management Web site URL https://documents.egi.eu/public/ShowDocument?docid=440
[10] EMI Requirement tracker Web site URL https://savannah.cern.ch/task/?group=emi-req
[11] GFD.147 GLUE 2.0 Specification Web site URL http://ogf.org/documents/GFD.147.pdf
[12] Cecchi M et al 2010 The gLite Workload Management System J. Phys.: Conf. Ser. 219 062039
[13] BDII, the Berkeley Database Information Index Web site URL https://twiki.cern.ch/twiki/bin/view/EGEE/BDII
[14] Classad Web site URL http://research.cs.wisc.edu/condor/classad/
[15] EMI Execution Service (EMI-ES) Specification, Web site URL, http://cdsweb.cern.ch/record/1452918
[16] SLURM batch system Web site URL https://computing.llnl.gov/linux/slurm/
[17] Dichev K et al 2008 MPI Support on the Grid Computing and Informatics, 27, pp. 213–222
[18] Argus authorization system Web site URL https://twiki.cern.ch/twiki/bin/view/EGEE/AuthorizationFramework
[19] Shibboleth Web site URL http://shibboleth.internet2.edu/
[20] Kerberos, The Network Authentication Protocol. Web site URL http://web.mit.edu/kerberos/
[21] EMI Common Authentication Library (CANL) Web site URL,
https://twiki.cern.ch/twiki/bin/view/EMI/EmiJraIT4SecurityCommonAuthNLib
[22] OCSP Online Certificate Status Protocol Web site URL
http://tools.ietf.org/html/rfc2560

[23] EMIR design Web site URL http://cdsweb.cern.ch/record/1449141?ln=en

[24] Memon S et al 2012 EMIR: An EMI Service Registry for Federated Grid Infrastructures
Proceedings of 1st European Grid Infrastructure (EGI) Community Forum/ 2nd European
Middleware Initiative (EMI) Technical Conference Muenich, Germany 2012

[25] Definition of the Compute Accounting Record (CAR), Web site URL
http://cdsweb.cern.ch/record/1449764

[26] EMI StAR – Definition of a Storage Accounting Record Web site URL
http://cdsweb.cern.ch/record/1452920

[27] Nilsen J K et al 2012 EMI datalib, EMI datalib - joining the best of ARC and gLite data
libraries submitted to Proceedings of CHEP 2012

[28] The gLite File Transfer Service Web site URL
https://wiki.chipp.ch/twiki/pub/LCGTier2/FTSlinks/transfer.pdf

[29] WebDAV protocol Web site URL http://tools.ietf.org/html/rfc4918

[30] Milton libraries Web site URL http://http://milton.io

[31] 101tec Web site URL http://webdav.101tec.com