Abstract:
This deliverable contains the detailed work plan of the Compute Services technical area compliant with the overall EMI Technical Development Plan. The plan is released early in the project life and updated every year including a status report on the achievements of the past 12 months compared to the planned objectives. The status report at M03 will cover the state-of-the-art while the work plan at M36 will provide recommendations for further work.
Copyright notice:

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

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

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

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

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

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

<table>
<thead>
<tr>
<th>From</th>
<th>Partner / Activity</th>
<th>Date</th>
<th>Signature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Massimo Sgaravatto</td>
<td>INFN/JRA1</td>
<td>01/09/2010</td>
<td></td>
</tr>
<tr>
<td>Reviewed by</td>
<td>Michel Drescher</td>
<td>30/09/2010</td>
<td></td>
</tr>
<tr>
<td>Michele Carpene`</td>
<td>INFN/SA1</td>
<td>15/10/2010</td>
<td></td>
</tr>
<tr>
<td>Reviewed by</td>
<td>Alberto Aimar</td>
<td>26/10/2010</td>
<td></td>
</tr>
<tr>
<td>Maria Alandes Pradillo</td>
<td>CERN/SA2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jozef Cernak</td>
<td>CERN/SA2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>UPJS/SA2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Approved by</td>
<td>PEB</td>
<td>30/11/2010</td>
<td></td>
</tr>
</tbody>
</table>

Document Log

<table>
<thead>
<tr>
<th>Issue</th>
<th>Date</th>
<th>Comment</th>
<th>Author / Partner</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>01/07/2010</td>
<td>First version. Integrated all but Unicore state-of-the-art contributions</td>
<td>Massimo Sgaravatto</td>
</tr>
<tr>
<td>0.2</td>
<td>14/07/2010</td>
<td>Added state-of-the-art section also for Unicore</td>
<td>Massimo Sgaravatto</td>
</tr>
<tr>
<td>0.3</td>
<td>21/07/2010</td>
<td>Added workplan section.</td>
<td>Massimo Sgaravatto</td>
</tr>
<tr>
<td>0.4</td>
<td>22/07/2010</td>
<td>Added “executive summary” section. References finalized. Approved by relevant PT leaders</td>
<td>Massimo Sgaravatto</td>
</tr>
<tr>
<td>0.4.1</td>
<td>22/07/2010</td>
<td>Minor fix. Submitted to Compute Area mailing lists: no issues raised</td>
<td>Massimo Sgaravatto</td>
</tr>
<tr>
<td>0.5</td>
<td>03/08/2010</td>
<td>Integrated comments sent by Balazs Konya</td>
<td>Massimo Sgaravatto</td>
</tr>
<tr>
<td>0.6</td>
<td>31/08/2010</td>
<td>Removed stuff concerning integration with LB (as requested by Balazs Konya)</td>
<td>Massimo Sgaravatto</td>
</tr>
<tr>
<td>0.7</td>
<td>09/11/2010</td>
<td>Addressed most of the comments raised by M. Riedle, M. Carpene`, M. Drescher, A. Aimar, M. Alandes Pradillo, J. Cernak</td>
<td>Massimo Sgaravatto</td>
</tr>
<tr>
<td>0.8</td>
<td>11/11/2010</td>
<td>Executive summary rewritten. Some other minor changes</td>
<td>Massimo Sgaravatto</td>
</tr>
<tr>
<td>1.0</td>
<td>08/12/2010</td>
<td>Final version</td>
<td>Alberto Di Meglio</td>
</tr>
</tbody>
</table>
### Document Change Record

<table>
<thead>
<tr>
<th>Issue</th>
<th>Item</th>
<th>Reason for Change</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
TABLE OF CONTENTS

1. INTRODUCTION ............................................................................................................................................. 6
   1.1. PURPOSE .................................................................................................................................................. 6
   1.2. DOCUMENT ORGANISATION .................................................................................................................. 6
   1.3. REFERENCES ........................................................................................................................................... 6
   1.4. DOCUMENT AMENDMENT PROCEDURE ............................................................................................... 8
   1.5. TERMINOLOGY ........................................................................................................................................ 8

2. EXECUTIVE SUMMARY ..................................................................................................................................... 10

3. STATE OF THE ART ........................................................................................................................................ 14
   3.1. ARC CE ................................................................................................................................................... 14
   3.2. ARC COMPUTE CLIENTS ......................................................................................................................... 16
   3.3. GLITE JOB MANAGEMENT SERVICES .................................................................................................. 17
   3.4. GLITE MPI ................................................................................................................................................ 21
   3.5. UNICORE JOB MANAGEMENT COMPONENTS .................................................................................... 22
       3.5.1 UNICORE TARGET SYSTEM ACCESS ............................................................................................... 23
       3.5.2 UNICORE WEB SERVICE INTERFACES ............................................................................................. 24
       3.5.3 UNICORE CLIENT & APIs ................................................................................................................... 25

4. WORKPLAN ..................................................................................................................................................... 27
   4.1. HARMONIZATION ACTIVITIES ................................................................................................................ 29
       4.1.1 Definition and adoption of job related standard interfaces ................................................................. 29
       4.1.2 Adoption of information service related standards ............................................................................. 31
       4.1.3 Harmonization of the authorization mechanisms .............................................................................. 32
       4.1.4 Harmonization in the parallel and MPI job support .......................................................................... 32
       4.1.5 Consolidation and harmonization of compute area clients/APIs ......................................................... 33
       4.1.6 Investigate possible use cases for a common standard messaging system in the computing area ....... 34
       4.1.7 GSI removal ......................................................................................................................................... 34
   4.2. EVOLUTION PLANS ................................................................................................................................. 35
       4.2.1 Evolution plans for the ARC CE ........................................................................................................... 35
       4.2.2 Evolution plans for the ARC compute clients ...................................................................................... 36
       4.2.3 Evolution plans for the gLite job management services ..................................................................... 36
       4.2.4 Evolution plans for gLite-MPI ............................................................................................................... 37
       4.2.5 Evolutions plans for UNICORE job management components ....................................................... 38

5. CONCLUSIONS .............................................................................................................................................. 39
1. INTRODUCTION

1.1. PURPOSE
This document covers the state-of-the-art of the EMI compute area components.

It then outlines the planned harmonization and evolution work in the context of these middleware components foreseen for the first project year. In particular, the document presents the first year development plans of the following compute area product teams consisting of numerous relevant components:

- ARC Compute Element
- ARC Compute Clients
- gLite Job Management
- gLite MPI
- UNICORE Target System Access
- UNICORE WS Interfaces
- UNICORE Client and APIs

1.2. DOCUMENT ORGANISATION
This document is basically organized in two parts. In the first one (Sect. 3) an overview of the current status of the EMI compute area software components is presented. The second part, Sect. 4, presents the planned workplan for the first project year. Sect. 5 concludes the document.

1.3. REFERENCES

| R1  | http://www.nordugrid.org/middleware |
| R2  | http://www.nordugrid.org/documents/GM.pdf |
| R5  | http://www.nordugrid.org/documents/ARCHED_article.pdf |
| R7  | http://www.nordugrid.org/documents/ui.pdf |
| R10 | http://www.gLite.org |
| R11 | http://grid.pd.infn.it/cream |
| R12 | http://web.infn.it/gLiteWMS/ |
1.4. DOCUMENT AMENDMENT PROCEDURE

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

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

1.5. TERMINOLOGY

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>A-REX</td>
<td>ARC Resource-coupled EXecution</td>
</tr>
<tr>
<td>AGU</td>
<td>ARC gLite UNICORE</td>
</tr>
<tr>
<td>ARC</td>
<td>Advanced Resource Connector</td>
</tr>
<tr>
<td>BES</td>
<td>Basic Execution Services</td>
</tr>
<tr>
<td>BLAH</td>
<td>Batch Local Ascii Helper</td>
</tr>
<tr>
<td>CLI</td>
<td>Command Line Interface</td>
</tr>
<tr>
<td>CREAM</td>
<td>Computing Resource Execution And Management</td>
</tr>
<tr>
<td>DEISA</td>
<td>Distributed European Infrastructure for Supercomputing Applications</td>
</tr>
<tr>
<td>DoW</td>
<td>Description of Work</td>
</tr>
<tr>
<td>FTS</td>
<td>File Transfer Service</td>
</tr>
<tr>
<td>HED</td>
<td>Hosting Environment Daemon</td>
</tr>
<tr>
<td>HEP</td>
<td>High Energy Physics</td>
</tr>
<tr>
<td>HILA</td>
<td>High Level API</td>
</tr>
<tr>
<td>HPC</td>
<td>High Performance Computing</td>
</tr>
<tr>
<td>HPC-BP</td>
<td>HPC-Basic Profile</td>
</tr>
<tr>
<td>ICE</td>
<td>Interface to CREAM Environment</td>
</tr>
<tr>
<td>JC</td>
<td>Job Controller</td>
</tr>
<tr>
<td>JDL</td>
<td>Job Description Language</td>
</tr>
<tr>
<td>JMS</td>
<td>Job Management Service</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>JSDL</td>
<td>Job Submission Description Language</td>
</tr>
<tr>
<td>JURA</td>
<td>Job Usage Reporter of ARC</td>
</tr>
<tr>
<td>LAM-MPI</td>
<td>Local Area Multicomputer-MPI</td>
</tr>
<tr>
<td>LB</td>
<td>Logging and Bookkeeping</td>
</tr>
<tr>
<td>LRMS</td>
<td>Local Resource Management System</td>
</tr>
<tr>
<td>MCT</td>
<td>Minimum Completion Time</td>
</tr>
<tr>
<td>MPI</td>
<td>Message Passing Interface</td>
</tr>
<tr>
<td>NGI</td>
<td>National Grid Initiative</td>
</tr>
<tr>
<td>OGF</td>
<td>Open Grid Forum</td>
</tr>
<tr>
<td>PACX-MPI</td>
<td>PArallel Computer eXtension-MPI</td>
</tr>
<tr>
<td>PGI</td>
<td>Production Grid Infrastructure</td>
</tr>
<tr>
<td>PBS</td>
<td>Portable Batch System</td>
</tr>
<tr>
<td>PT</td>
<td>Product Team</td>
</tr>
<tr>
<td>RTE</td>
<td>Run Time Environment</td>
</tr>
<tr>
<td>SE</td>
<td>Storage Element</td>
</tr>
<tr>
<td>SGAS</td>
<td>SweGrid Accounting System</td>
</tr>
<tr>
<td>SGE</td>
<td>Sun Grid Engine</td>
</tr>
<tr>
<td>SLURM</td>
<td>Simple Linux Utility for Resource Management</td>
</tr>
<tr>
<td>SMS</td>
<td>Storage Management Service</td>
</tr>
<tr>
<td>TSF</td>
<td>Target System Factory</td>
</tr>
<tr>
<td>TSS</td>
<td>Target System Service</td>
</tr>
<tr>
<td>UCC</td>
<td>UNICORE Commandline Client</td>
</tr>
<tr>
<td>UNICORE</td>
<td>UNiform Interface to COmputing REsources</td>
</tr>
<tr>
<td>XACML</td>
<td>eXtensible Access Control Markup Language</td>
</tr>
<tr>
<td>XNJS</td>
<td>eXtensible Networked Job Supervisor</td>
</tr>
<tr>
<td>WG</td>
<td>Working Group</td>
</tr>
<tr>
<td>WLCG</td>
<td>Worldwide LHC Computing Grid</td>
</tr>
<tr>
<td>WMS</td>
<td>Workload Management System</td>
</tr>
<tr>
<td>WS</td>
<td>Web Service</td>
</tr>
</tbody>
</table>
2. EXECUTIVE SUMMARY

The job management services relevant for the EMI project have been implemented in the past years in the context of different middleware stacks: ARC, gLite and UNICORE. These job management services are now used by different user communities with interests as diverse as Earth Sciences, Computer Sciences and Mathematics, Fusion, Life Sciences or High-Energy Physics.

For what concerns ARC, the ARC computing element implements job execution capability over a large variety of computational resources. The ARC CE relies on the core functionality of the Grid Manager (GM) exposed via the GridFTP jobplugin interface. The purpose of GM is to take necessary actions on a computing cluster front-end, needed to execute a job. GM processes job descriptions written in ARC's Extended Resource Specification Language (XRSL) or JSDL (with some ARC extensions). Since ARC release 0.8, transition from the gridftp interfaced GM to the new A-REX service has started. A-REX is the next generation computing element of ARC offering WS-interfaces and advanced security solutions. It is built around GM and inherits all its conceptual design and functionality.

The ARC Compute Clients PT is responsible in ARC for the client-side job execution components that are used for resource discovery and brokering, job submission and management operations. It consists of basically three components: the pre WS client, the WS client and the libarcclient (which is a powerful modular library, supporting information retrieval, job submission and management, job description translator and resource broker modules).

In gLite, job submission and management is instead implemented by the following components: WMS, CREAM, CEMon, BLAH.

CREAM is a simple, lightweight service for job management operation at the Computing Element level. Its main functionality is job management: users submit jobs described as JDL expressions, and CREAM executes them on an underlying LRMS (batch system). CREAM is a Java application which runs as an Axis servlet inside the Tomcat application server, and exposes a Web service interface.

The gLite WMS is instead a higher level service responsible for distributing and managing tasks across computing and storage resources available on a Grid. WMS assigns user jobs to CEs and SEs belonging to a Grid environment in a convenient fashion, so that they are always executed on resources that match the job requirements and Grid-wide load balance is maintained.

BLAHPD is a light component used in the CREAM CE, accepting commands to manage jobs on different Local Resources Management Systems. The BLAH service provides a minimal, pragmatically designed common interface for job submission and control to a set of batch systems. BLAH, with its Blparser component, is also responsible to efficiently obtain the current status of BLAH active jobs and notify CREAM accordingly.

CEMon is a generic framework for information gathering and provisioning. It provides an asynchronous event notification framework, which can be coupled with CREAM to notify the users when job status changes occur.

In gLite, execution of parallel jobs is managed through the MPI-Start software component. Once the resources needed for the execution of a parallel job have been allocated, MPI-Start provides a unified layer for starting the parallel application. It hides the underlying details of the resources and MPI frameworks to the upper middleware layers and to the users.

The job execution functionality of UNICORE is covered by the UNICORE Target System Access, the UNICORE web service interfaces and the UNICORE client and APIs.
The Unicore Target System Access comprises two components: the UNICORE XNJS and the UNICORE TSI. The XNJS component is the job management and execution engine of UNICORE. Its basic job is to accept tasks from the upper layers (i.e. the web service interfaces), parses the task description and generates site-specific scripts for performing specific tasks like creating a directory or submitting a batch job. These scripts are then sent to the TSI component for execution. The TSI is a daemon written in Perl which is running on a frontend node of the target resource.

The UNICORE web service interfaces provide the actual web services that are deployed in the UNICORE/X container, and which use the XNJS/TSI for performing the needed work on the backend. The functionality of the XNJS is accessible via two services interfaces, a UNICORE's proprietary interface, called UAS and a standardized OGSA-BES interface.

The functionality offered by UNICORE is fully accessible through a set of basic client classes. On top of these client classes, the actual UNICORE end user clients are built. The UNICORE command-line client (UCC) is a universal, easy to use tool which offers an extensible set of commands for interacting with UNICORE sites, and even non-UNICORE OGSA-BES services. For integration of Grid functionality into other applications and for the simple development of new Grid clients or Grid portals, the high-level HiLA client API is available.

Different approaches were used when designing and implementing such components, very often leading to proprietary solutions. This was done mainly because of the unavailability of consolidated and usable standards in the area, at least at the time when these services were designed and implemented.

The same issue applies also to the services used by these job management components: authentication, authorization, accounting, etc. are often managed in different ways in the different implementations. Therefore, even if most of the considered services (the job management services and the ones used by these job management components) provide the same core functionality; they are implemented in different and often incompatible ways.

In this document the plans to address some of these issues (foreseen to be implemented in the first year of the EMI project) are presented. In particular the plan is:

- **To define and adopt job execution standard/agreed interfaces.**
  A specification for an EMI Execution Service is being defined by a specific task force composed by ARC, gLite and UNICORE representatives. As soon as this EMI specification is ready, it will be implemented in the EMI job management products.

- **To adopt information service related standards.**
  The well established relevant standard for the specification of the characteristics and status of Grid entities is GLUE. The latest available specification is GLUE v. 2.0, which has been published as OGF Proposed Recommendation, and which will be supported in the EMI components of the compute area.

- **To harmonize the used authorization mechanisms.**
  A single authorization service, namely the gLite Argus service, will be considered in EMI. The EMI job management components will therefore need to be integrated with such Argus.
• To harmonize the parallel and MPI job support.
The description for parallel and MPI jobs (along with the description of the needed resources) will be standardized in the context of the job related standardization activities. The standardization of the “back-end”, that is the provision of a standard EMI unified layer, targeted to the end user and responsible to manage the parallel application hiding the underlying details will be assessed.

• To consolidate and harmonize the compute area clients and APIs
It will be investigated which kind of harmonization and consolidation can be done for the clients, and concrete plans (to be then implemented) will be settled. The main objectives are to improve: usability, maintainability and portability to different architectures and operating systems.

• To assess the use of a messaging system for the EMI job management components
The infrastructure area of the EMI project currently investigates a solution for messaging to be used within the greater EMI architectural framework. It will be assessed how such messaging infrastructure can be useful for the services of the compute area.

• To remove GSI
An overall strategy of the EMI project is to remove the Globus proprietary GSI protocol and replace it with standard SSL/TLS. This will have to be done also for the services of the compute area.

The document also presents the planned evolution activities for the EMI compute area services, that is the new foreseen developments targeted to address the requirements of users and to adapt to the changes in the underlying used technologies. These planned activities include:

• Improvements of the ARC CE Janitor component for the dynamic management of Run Time Environments.
• Provision of accounting hooks in the ARC CE.
• Planning for a new data staging system in the ARC CE.
• Provision of libarcclient language bindings on Windows.
• Improvement of the libarcclient data broker.
• Implementation in the gLite WMS of general feedback mechanisms, to migrate jobs from busy to idle computing resources.
• Improvement of MPI job execution on gLite middleware.
• More flexible generation of site-specific scripts in UNICORE Target System Access.
The workplan presented in this document will be updated at the end of the first project year and at the end of the second year of the project.
3. STATE OF THE ART

Different services providing compute and job related functionality have been implemented in the last years and are being used by several user communities for real production activities in several Grid infrastructures.

However these services, which have been designed and implemented in the context of different Grid projects, have been realized leading to proprietary solutions. This was mainly because of the unavailability (at least when these services have been designed) of consolidated and usable standard solutions. Therefore most of these services provide the same core functionality (e.g. job submission and management) but in different and incompatible ways.

Also the security mechanisms (e.g. for authentication, authorization, e.g.) used in these job management services are usually different and in some way even incompatible. The lack of common or at least compatible solutions applies also to other areas: accounting, service monitoring and management, etc.

3.1. ARC CE

The ARC [R1] middleware (which includes the ARC CE and the ARC compute clients) is used by a number of national and international Grid infrastructures. Most notably, it is the basis (together with dCache) for the Nordic DataGrid Facility, and is deployed at all Nordic computing centers. Apart of that, it supports national Grid initiatives in Sweden, Norway, Finland, Ukraine and Slovenia, and is used by research projects in Switzerland, UK, Hungary, Slovakia, Germany and Russia. The key user community is High Energy Physics (mostly ATLAS experiment at CERN, as well as a smaller scale usage by CMS and ALICE), followed by a variety of bioinformatics and other life science communities, material sciences, computational chemistry, Earth sciences, and even humanities. Each of these communities deploys own application tools and environments, either integrated with ARC (like HEP production frameworks), or used as runtime environments. The first production-level deployment of ARC took place in summer 2002.

The ARC computing element (CE) implements job execution capability over a large variety of computational resources. The ARC CE PT is in charge of delivering all components necessary for it to function. This includes the core job management component Grid Manager (GM) [R2] and its replacement re-implemented inside A-REX [R4], and the necessary for their functioning components: the GridFTP-based job submission and data transfer tool (jobplugin), data stage in and stage out utilities (downloaders and uploaders), cache management utilities, LRMS modules, RTE management service (Janitor) and accounting hooks. The architecture of the ARC-CE is shown in Figure 1 (dotted lines indicate communications via file system, and white blocks indicate components that are not covered by the EMI project).

Description of this system architecture, provided functionality and usage is given below.

The ARC CE relies on the core functionality of the Grid Manager exposed via the GridFTP jobplugin interface. The purpose of GM is to take necessary actions on a computing cluster front-end, needed to execute a job. A Grid job is typically defined as a set of input files, a main executable, the requested resources (including the needed application software) and a set of produced files. The entire process of gathering the input files, executing the main payload in the LRMS, and transferring/storing the output files is handled by GM. GM processes job descriptions written in ARC's Extended Resource Specification Language (XRSL) [R3] or JSDL [R32] (with some ARC extensions).
Each job gets a directory on the execution cluster, the session directory, which is created by GM as a part of the job preparation process. The job is supposed to use this directory for all input and output data, and possibly other auxiliary files.

Input files are staged in the session directory by the downloader module of GM, or pushed by a client. Downloader is a data management client application capable of fetching data from various sources, including those registered in indexing services. Support for replicas of data and download retries makes data transfer more robust. To do that, downloader uses the ARC datamove library.

![Image of ARC Compute Element](image)

**Figure 1: The ARC job submission**

GM handles a cache of input files. Every URL requested for download is first checked against the contents of the cache. If a file associated with a given URL is already cached, the remote storage is contacted to check the file's validity and the access permissions based on the user's credentials. If a file stored in cache is found to be valid, it is either copied to session directory, or a soft-link is used, depending on the configuration of GM.

After the contents of the session directory has been prepared, GM communicates with the LRMS (a batch system) to run the main payload of the job. GM uses a set of back-end shell scripts in order to perform such LRMS operations as job submission, detection of the exit state of the main executable, etc. Currently supported batch system types include various PBS [R33] flavors, Condor [R34], SGE
[R35], LoadLeveler [R36] and SLURM [R37]. There is also support for job execution on the front-end, meant for special testing purposes only.

The output files generated by the job have to be listed in the job description together with their final destinations, or indicated to be kept on the computing resource for eventual retrieval. After the job's main executable has stopped running, they are staged out to the specified destinations by the uploader module. The uploader is similar to the downloader described above.

Support for software runtime environments (RTE) is also a part of GM. RTEs are meant to provide a unified access to pre-installed application software packages. They provide predictable and functionality-wise identical environments for jobs.

A job passes through several states in GM, and each state can be assigned to an external plugin. This way, the functionality of GM can be significantly extended with external components to include e.g. logging, fine-grained authorization, accounting, filtering, etc.

GM itself does not have a network port. Instead, it reads the information about job requests from the local file system. This information is delivered through the GridFTP channel by using a customized plugin for the ARC GridFTP server, named jobplugin. Information about the jobs served by GM and content of their session directories is represented by the same plugin through the virtual file system. Session directories are mapped to FTP directories. The files which are not accessible from a machine running downloader (like those stored at the user's computer) can be provided by the client through direct upload to the session directory via jobplugin GridFTP interface.

GM has several other features. Those include E-mail notification of job status changes, support for site local credentials through customizable plugins, authorization and accounting plugins, and a possibility to report jobs to a centralized logging service. All the features are highly configurable and can be turned on and off by either user or administrator.

Since ARC release 0.8 [R46], transition from the gridftp interfaced GM to the new execution service has started. The ARC Resource-coupled Execution service (A-REX) [R4] is the next generation computing element of ARC offering WS-interfaces and advanced security solutions [R47]. A-REX is built around GM and inherits all its conceptual design and functionality. A-REX interprets standard job description, offers OGF-compliant job execution and management interface (BES, with some ARC extensions), features transparent, automatic and integrated data staging and caching capability, support for large number of batch systems, session directory management, comes with logging capability and support for RTEs. A-REX offers Web Service-based local information interface serving Glue2 information. A-REX is also capable of working together with community approved security frameworks. A-REX does not require a GridFTP server and is hosted by the ARC HED [R5], which provides all the necessary external interfaces and the hosting environment for the services.

Being part of a modular middleware stack, A-REX works together with other components in order to provide the full CE functionality. Contrary to the rather monolithic GM, A-REX can utilize Janitor service for dynamic management of RTEs, and JURA [R6] accounting hooks in order to report usage data to SGAS [R38]. However, these components are not yet distributed with ARC and require further hardening.

### 3.2. ARC COMPUTE CLIENTS

The ARC Compute Clients PT is responsible for the client-side job execution components that are used for resource discovery and brokering, job submission and management operations. It consists of basically three components:
• the pre WS client
• the WS client
• the libarcclient

The pre WS client [R7] and the WS client [R8], which currently co-exist in the ARC production release, provides commands for:

• job submission (ngsub/arcsub), also with built-in brokering mechanisms;
• resubmission (ngresub/arcreresub);
• job migration (arcmigrate);
• job and cluster querying (ngstat/arccstat/arcinfo);
• retrieval of job results/output (ngget/arccget);
• view of stdout/stderr or server logs (ngcat/arccat);
• removal of jobs from target (ngclean/arcclean);
• kill running jobs (ngkill/arcckill);
• proxy renewal (ngrenew/arcrenew);
• resuming of failed jobs (ngresume/arcresume);
• synchronization of job list (ngsync/arcsync);
• testing client and server setup (ngtest).

All these clients build on the libarcclient [R9] general purpose library.

The libarcclient is a powerful modular library, supporting information retrieval, job submission and management, job description translator and resource broker modules. Additionally the library has a simple interface, which allows to easily creating custom plugins of any of the mentioned modules. The library is written in C++, but that does not limit third party applications from interfacing the library since language bindings for both Python and Java are provided. It is supported on all the major platforms, including Windows, Mac OS X and Linux. Packages for these platforms are provided. The libarcclient component, just as the ARC CLIs, is on the way to be distributed via the standard Debian, Ubuntu, Fedora and RedHat distributions.

3.3. GLITE JOB MANAGEMENT SERVICES

The gLite Job Management PT includes the following components: WMS, CREAM, CEMon, BLAH. Figure 2 show the components involved in the gLite [R10] job submission chain (in the figure the LCG-CE and OSG-CE are in red because they are not covered by the EMI project, while the ARC CE
is in yellow because it is not a component implemented and supported by the gLite Job Management PT).

The CREAM [R11] (Computing Resource Execution And Management) Service is a simple, lightweight service for job management operation at the Computing Element (CE) level. It has been implemented in the context of the EGEE projects (the first version of the CREAM CE was released in production in October 2008) and it is now deployed in several Grid production Grids on a national, European and international level, and used in a wide variety of application scenarios, such as astrophysics, biomedicine, computational chemistry, earth sciences, high energy physics, finance, fusion, geophysics, multimedia, geophysics etc. In particular it is used in the EGEE/EGI production Grid and in the Worldwide LHC Computing Grid (WLCG). The deployment of CREAM in the Open Science Grid (OSG) is starting now.

The Workload Management System (gLite WMS) [R12] is instead a higher level service responsible for distributing and managing tasks across computing and storage resources available on a Grid. WMS assigns user jobs to CEs and SEs belonging to a Grid environment in a convenient fashion, so that they are always executed on resources that match the job requirements and Grid-wide load balance is maintained (i.e. jobs are evenly and efficiently distributed across the entire Grid).

The first implementation of the WMS was done in the context of European DataGrid (EDG) project in 2002, and it was then evolved and maintained in the context of the EGEE projects. As CREAM, it is also deployed in several Grid infrastructures (in particular the EGEE/EGI and WLCG Grids) and used for several applications.

There are two entry points for job management requests: the gLite WMS User Interface (UI) and the CREAM UI. Both include a set of command line tools which can be used to submit, cancel and query the status of jobs. In gLite, jobs are described using the Job Description Language (JDL) [13] [14] notation, which is a textual notation based on Condor classads [R39]. CREAM and WMS use very similar JDL dialects: there are however some small differences, which are motivated by the different role of the job execution and workload management services.

The CREAM UI is used to interact directly with a specific CREAM CE. It is a set of command line tools which invoke the Web Services operations exposed by CREAM.

On the other hand, the gLite WMS UI allows the user to submit and monitor jobs through the gLite Workload Management System.

Job management through the WMS provides many benefits compared to direct job submission to the CE:

- The WMS can manage multiple CEs, and is able to forward jobs to the one which better satisfies a set of requirements;
- The WMS can be instructed to handle job failures: if a job aborts due to problems related to the execution host (e.g. host misconfiguration) the WMS can automatically resubmit it to a different CE;
- The WMS supports complex job types (job collections, job with dependencies) which cannot be handled directly by the CEs.
The WMS exposes a Web Service interface which is implemented by the WMProxy component. The core of the WMS is the Workload Manager (WM), whose purpose is to accept and satisfy requests for job management. For job submissions, the WM tries to locate an appropriate resource (CE) where the job can be executed. The decision of which resources should be used is the outcome of the matchmaking process between the requests and the available resources. The user can specify a set of requirements (which includes also possible data access requirements) in the job description. These requirements represent a set of constraints which the WM tries to satisfy when selecting the CE where the job will be executed.

Interaction with the gridftp based ARC-CE and with the Globus pre-ws based CEs (LCG-CE and OSG-CE) is handled by the Job Controller (JC) - Log Monitor (LM) - CondorG modules within the WMS. The LCG-CE and the OSG-CE are not covered by the EMI project.
In the case of submission to CREAM-based CEs, jobs are managed by a different module, called ICE (Interface to Cream Environment). ICE receives job submissions and other job management requests from the WM component and then uses the operations of the CREAM interface to perform the requested operation. Moreover, it is responsible for monitoring the state of submitted jobs and for taking the appropriate actions when job status changes are detected (e.g. to trigger a possible resubmission if a Grid failure is detected).

For what concerns CREAM, its main functionality is job management. Users submit jobs described as JDL expressions, and CREAM executes them on an underlying LRMS (batch system). It is a Java application which runs as an Axis servlet inside the Tomcat application server, and exposes a Web service interface.

CREAM supports the execution of batch (normal/serial) and parallel (MPI) jobs.

The CREAM interface exposes a set of operations which can be classified in three groups. The first group of operations (Lease Management) allows the user to define and manage leases associated with jobs. The lease mechanism has been implemented to ensure that all jobs get eventually properly managed, even if the CREAM service loses connection with the client application (e.g. because of network problems). The second group of operations (Job Management) is related with the core functionality of CREAM as a job management service. Operations are provided to create a new job, start execution of a job, suspend/resume or terminate a job. Moreover, the user can get the list of all owned jobs, and it is also possible to get the status of a set of jobs. Finally, the third group of operations (Service Management) deals with the whole CREAM service. It consists of two operations, one for enabling/disabling new job submissions (these operations are accessible only to users with administration privileges), and one for accessing general information about the service itself.

Concerning the architecture, CREAM can be seen as an abstraction layer on top of an LRMS (batch system), which extends the LRMS capabilities with an additional level of security, reliability, and integration with a Grid infrastructure. CREAM supports different batch systems through the concept of LRMS connectors. An LRMS connector is an interface for a generic batch system. Currently, CREAM supports all the batch systems supported by BLAHPD through a specific instance of LRMS connector called the BLAH connector module.

BLAHPD [16] is a light component accepting commands according to the BLAH (Batch Local Ascii Helper) protocol to manage jobs on different Local Resources Management Systems (LRMS). The BLAH service provides a minimal, pragmatically designed common interface for job submission and control to a set of batch systems. Interaction with BLAH is achieved via text commands while interaction with each one of the various supported batch system is achieved via customized scripts. BLAH, with its Blparser component, is also responsible to efficiently obtain the current status of BLAH active jobs and notify CREAM accordingly.

CREAM is not the only client of the BLAH service. For example BLAH is also part of the Condor distribution where it can be used to interact with the batch system.

CEMonitor (CEMon) [17] is another service which is part of the CREAM CE. Actually CEMonitor is a generic framework for information gathering and provisioning, and thus can also be used without being coupled with CREAM. For example in the context of the Open Science Grid (OSG) [R40] it is widely used to manage information about computing resources.

CEMon provides an asynchronous event notification framework, which can be coupled with CREAM to notify the users when job status changes occur. CEMonitor publishes information as Topics. For each Topic, CEMonitor maintains the list of Events to be notified to users. Users can create Subscriptions for topics of interest. Each Topic is produced by a corresponding Sensor. A Sensor is a
component which is responsible for actually generating Events to be notified for a specific Topic. The most important Sensor for the CREAM CE is called JobSensor, which produces the events corresponding to CREAM job status changes. When CREAM detects that a job changes its status (for example, an Idle job starts execution, thus becoming Running), it notifies the JobSensor by sending a message on the network socket where the sensor is listening. Then, the JobSensor triggers a new notification which is eventually sent to all subscribed users.

Each Sensor can provide either asynchronous notifications to registered listeners, or can be queried synchronously. In both cases, Sensors support a list of so-called Query Languages. A Query Language is a notation (e.g., XPath, classad expressions and so on) which can be used to ask a Sensor to provide only Events satisfying a user-provided condition. When an Event satisfies a condition, CEMonitor triggers an Action on that event. In most cases, the Action simply instructs CEMonitor to send a notification to the user for that event. Of course, it is possible to extend CEMonitor with additional types of user-defined Actions. When registering for asynchronous notifications, the user passes a query expressed in one of the supported Query Languages as parameter. So, for that subscription, only events matching the query are notified.

Sensors support different Dialects. A Dialect is a specific output format which can be used to render Events. This means that a Sensor can publish information in different formats: for example, job status change information could be made available in Condor classad format or in XML format.

3.4. GLITE MPI

The gLite MPI product team is responsible for the MPI-start and MPI-utils components.

Execution of parallel applications on a Grid environment requires the cooperation of several middleware tools and services. Although current grid middleware stacks have a basic level of support for such applications, this support is very limited. Users face a very heterogeneous environment lacking any standards for managing their jobs.

MPI-Start, which has been designed and implemented in the context of the int.eu.grid project [R18] and which has been used in the gLite based EGEE Grid, addresses these issues. Once the resources needed for the execution of a parallel job have been allocated by the Computing Element, MPI-Start provides a unified layer for starting the parallel application. It hides the underlying details of the resources and MPI frameworks to the upper middleware layers and to the users. MPI-Start automatically detects and uses site-specific configuration features (such as the batch scheduler and the file system at the site) without user intervention. Users just need to specify their executable and arguments to run their application, while at the same time, advanced users can easily customize the behavior and have a tighter control on the application execution.

MPI-Start is developed as a set of shell scripts that ease the execution of MPI programs by using a unique and stable interface to the middleware. These scripts are written in a modular way: there is a core and around it there are different frameworks with plug-ins that provide specific functionality. Three main frameworks are defined in the MPI-Start architecture:

- **Scheduler framework.** Every plug-in for this framework provides support for different Local Resource Management Systems (LRMS). They must detect the availability of a given scheduler and generate a list of machines that will be used for executing the application. The currently supported LRMS are SGE, PBS/Torque and LSF [R41].
• **Execution framework.** These plug-ins set the special parameters that are used to start the MPI implementation. This MPI implementation is not automatically detected; hence the user or any job management system must explicitly specify which MPI flavor will be used to execute the application. There are plug-ins implemented for Open MPI, MPICH (including MPICH-G2), MPICH2, LAM-MPI and PACX-MPI.

• **Hooks framework.** In this framework any additional features that may be needed for the job execution is included. It is customizable by the end-user without the need of any special privileges. File distribution and compiler support is included in the hooks framework.

**gLite-MPI-utils** is a set of tools to make easy the installation and configuration of MPI and MPI-start in gLite based resources. It includes support for the most common MPI implementations and support for PBS/Torque batch systems.

### 3.5. UNICORE JOB MANAGEMENT COMPONENTS
The UNICORE [R19] middleware is used in a number of Grid initiatives on a national, European and international level. Most notably, it is the Grid middleware deployed on DEISA (Distributed European Infrastructure for Supercomputing Applications) initiative. Furthermore, UNICORE resources are offered by national Grid initiatives such as D-Grid (Germany), PL-Grid (Poland) or SKIF-Grid (Belarus).

UNICORE is used in a wide variety of application scenarios, from high-throughput screening in bioinformatics to large parallel jobs on state-of-the-art HPC machines.

The current major version UNICORE 6 was designed and implemented in the years 2004-2006, and officially released in August 2006.

The job execution functionality of UNICORE is covered by the following product teams and components:

- UNICORE TSA (XNJS, TSI)
- UNICORE WS Interfaces (UAS, OGSA-BES, etc.)
- UNICORE Client and APIs (UCC, internal libs, HILA)

The architecture of UNICORE is three-layered in client layer, service layer and system layer, as shown in Figure 3.

More details of the UNICORE job management related components are provided in the following subsections.
3.5.1 UNICORE TARGET SYSTEM ACCESS

The Unicore Target System Access comprises two components: the UNICORE XNJS [R20] and the UNICORE TSI [R21]. These products provide the fundamental capabilities of the UNICORE middleware, bridging the web services layer and the target system layer, i.e. interfacing to the operating system and batch system.

The XNJS (eXtensible Networked Job Supervisor) component is the job management and execution engine of UNICORE 6 and can be seen as the heart of a UNICORE 6 site. It provides access to storage resources, file transfer and job management functionality. It is a stateful component, which can persist its state information in the file system as well as in a number of relational databases like MySQL.

The XNJS has a pluggable, reconfigurable core, which can be extended easily to interact with other systems and perform new tasks. In the current UNICORE version, the XNJS is used as backend for execution of single jobs, and also as the execution engine in the workflow system, which shows the versatility and flexibility of the XNJS component.
The basic task of the XNJS in the UNICORE Target system access is to accept tasks from the upper layers (i.e. the web service interfaces), parses the task description and generates site-specific scripts for performing specific tasks like creating a directory or submitting a batch job. These scripts are then sent to the TSI component for execution.

The TSI component is a daemon written in Perl which is running on a frontend node of the target resource (usually a cluster or HPC system, but can also be a single computer). In general, XNJS and TSI run on different machines and communicate via sockets (with optional SSL). Multiple XNJSs can use the same TSI, and multiple TSIs can be connected to the same XNJS, allowing for load-balancing configurations.

In the TSI component the abstracted commands from the XNJS component are translated to system-specific commands (e.g. in the case of job submission, the specific commands like *lsubmit* or *qsub* of the local resource management system are called). The TSI component performs the proper setting of the user ID and invocation of his/her environment.

The TSI is available for a variety of commonly used batch systems such as Torque, LoadLeveler, SGE, LSF, SLURM, OpenCCS [R42], etc. Since it is written in Perl, it can be easily deployed in supercomputer environments that in the majority of the cases provide Perl as application. In contrast on Windows systems, one has to use the so-called embedded TSI, which is written in Java and is integrated directly into the XNJS.

### 3.5.2 UNICORE WEB SERVICE INTERFACES

The UNICORE web service interfaces provide the actual web services that are deployed in the UNICORE/X container, and which use the XNJS/TSI for performing the needed work on the backend.

The functionality of the XNJS is accessible via two services interfaces:

- A UNICORE's proprietary interface, called UAS (UNICORE Atomic Service) [R24]
- A standardized OGSA-BES interface [R25]

The UAS consists of several WS-RF standards compliant stateful Web services. First and foremost, the Target System Factory (TSF) represents a compute site and acts as a factory to the Target System Service (TSS) instance. The Target System Service (TSS) provides access to a stateful resource that models a physical computational Grid resource like a supercomputer. It exposes various pieces of information, e.g. details about the total numbers of CPUs, memory, etc. and preinstalled applications on the Grid enabled resource. Through the TSS Grid jobs described in the OGF standard Job Submission Description Language (JSDL) can be submitted to the UNICORE site. The jobs are managed and monitored with the Job Management Service (JMS). The JMS interface is used to manage a single job instance. Through the web service methods offered by the JMS, the job can be started, aborted, held, resumed and destroyed. Being a WSRF service, it provides an XML resource property document that contains relevant information about the job instance, such as:

- job status, consisting of the status (RUNNING, DONE, etc), status message, the job error code and the application progress
- job log that was produced by the execution service
To support data staging of JSDL jobs, the Storage Management Service (SMS) is used to access storages within Grid infrastructures. The transfer itself is implemented by the File Transfer Service (FTS).

A standardized set of interfaces based on open, common standards (OGSA-*) is available in UNICORE 6 in addition to the UAS. Currently implemented standards are OGSA-BES [R25] and HPC-BP [R43], used for the creation, monitoring and control of jobs. The OGSA-BES implementation describes a Web services based interface that is standardized by the OGF. In particular, the interface comprises functionality for the creation, monitoring, and control of computational jobs. Within the OGSA-BES specification such jobs are named as activities described via JSDL documents.

The fundamental benefit of the OGSA-BES is that it allows the end-users to transparently use (through the UNICORE clients) also non-UNICORE Grid implementations compliant with this standard.

The HPC Basic Profile (HPC-BP) specifies the usage of the OGSA-BES interface in conjunction with certain extensions for HPC environments, which are by far not matching the requirements from state-of-the-art supercomputers where UNICORE is typically deployed. Therefore UNICORE provides a tuned, but proprietary execution environment to address this issue.

3.5.3 UNICORE CLIENT & APIs

The functionality offered by UNICORE is fully accessible through a set of basic client classes that are part of the core UNICORE libraries. On top of these client classes, the actual UNICORE end user clients are built.

The UNICORE commandline client (UCC) [R26] is a universal, easy to use tool which offers an extensible set of commands for interacting with UNICORE sites, and even non-UNICORE OGSA-BES services.

The basic UCC features are:

- Allow to access the basic UNICORE 6 services, i.e. run jobs, manage output files, transfer data, etc.
- Easy-to-learn job definition syntax.
- Batch mode for running many jobs. UCC can run large numbers of jobs, which will be distributed to matching sites in a round-robin fashion (the matchmaking is done by UCC).
- Scripting using Groovy. UCC can execute arbitrary code written in the dynamic Java-based programming language Groovy. This allows complex applications which would otherwise entail elaborate coding.
- A UCC “shell” mode allowing executing UCC multiple commands without the need to restart UCC for each command.
UCC is extensible through additional commands written in Java.

UCC offers support for the UNICORE workflow system.

For integration of Grid functionality into other applications and for the simple development of new Grid clients or Grid portals, the HiLA client API [R27] was developed. The use of such high-level API facilitates the implementation of clients accessing Grid resources: application developers can just focus on the purely functional aspects of programming, since the extra amount of work necessary to access and use a Grid resource is hidden behind a concise interface. Another advantage provided by such API is the possibility to use multiple backend Grid environments. Besides the UNICORE 6 implementation, an implementation for OGSA-BES is in a development state. Additional backend implementations are planned to support new, standardized interfaces in the same uniform way.

Rather than designing a task focused API, HiLA takes a resource oriented approach to the definition of the Java interfaces. The linkages between the resources on the Grid are approached in a consistent manner. A resource is Locatable and possesses a Location. As such, there is a generic mechanism in place for navigating the linkages (for example, a Site has a number of Storages, a Storage contains a number of Files and a Site references its running Tasks). In this way different underlying Grid infrastructures become browsable in a uniform way. Analyzing and using Grid resources can then be as easy as browsing a UNIX file system.
4. WORKPLAN
In this section the JRA1 activities for the Computing services area foreseen for the first year of the project are presented. Such activities can be classified in two main categories:

- Harmonization activities, such as the implementation of standard solutions, the removal of duplications, the simplification of usage and maintenance.
- Evolution activities, which are the evolutionary new developments, to address requirements expressed by existing and new user communities or to adapt to the changes in the underlying technologies.

The workplan, described in detail in the following sections, is summarized in the following table:

<table>
<thead>
<tr>
<th>What</th>
<th>Who</th>
<th>When</th>
</tr>
</thead>
<tbody>
<tr>
<td>Definition and adoption of job related standard interfaces: specification of the EMI Execution Service (milestone MJRA1.2)</td>
<td>EMI-ES task force</td>
<td>PM7</td>
</tr>
<tr>
<td>Definition and adoption of job related standard interfaces: implementation of the EMI-ES specification in A-REX, CREAM CE, UNICORE XNJS, UNICORE web service interfaces, ARC Compute client, CREAM CE client, UNICORE HiLA and UCC (milestone MJRA1.3)</td>
<td>ARC CE PT, ARC Compute Clients PT, gLite job management PT, UNICORE Target System Access PT, UNICORE Web Services interface PT, UNICORE Web Services interface PT, UNICORE Client and APIs PT</td>
<td>PM18</td>
</tr>
<tr>
<td>Finalization of Glue2 support in the A-REX based ARC CE and ARC compute clients</td>
<td>ARC CE PT, ARC Compute Clients PT</td>
<td>PM10</td>
</tr>
<tr>
<td>Glue 2.0 support in the CREAM CE</td>
<td>gLite job management PT</td>
<td>PM10</td>
</tr>
<tr>
<td>Matchmaker module of the gLite WMS Glue 2.0 compliant.</td>
<td>gLite job management PT</td>
<td>PM10</td>
</tr>
<tr>
<td>Glue2 support in UNICORE job management services</td>
<td>UNICORE web service interfaces</td>
<td>PM10</td>
</tr>
<tr>
<td>Harmonization of the authorization mechanisms: ARC CE integrated with Argus</td>
<td>ARC CE PT</td>
<td>PM12</td>
</tr>
<tr>
<td>Harmonization of the authorization mechanisms: definition of a XACML authorization profile for the CREAM computing element</td>
<td>gLite job management PT</td>
<td>PM3</td>
</tr>
<tr>
<td>Project Description</td>
<td>Responsible Party</td>
<td>PM</td>
</tr>
<tr>
<td>-----------------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------</td>
<td>-----</td>
</tr>
<tr>
<td>Harmonization of the authorization mechanisms: implementation in the CREAM CE of the profile for Argus integration</td>
<td>gLite job management PT</td>
<td>PM10</td>
</tr>
<tr>
<td>Harmonization of the authorization mechanisms: evaluation of Argus for its use in the UNICORE job management services</td>
<td>UNICORE target system access PT</td>
<td>PM8</td>
</tr>
<tr>
<td>Report on the exercise to achieve a transparent user experience in running MPI jobs</td>
<td>MPI task force</td>
<td>PM7</td>
</tr>
<tr>
<td>Consolidation and harmonization of compute area clients/APIs: evaluation of the current state of the clients in the middlewares</td>
<td>EMI Compute Clients task force</td>
<td>PM7</td>
</tr>
<tr>
<td>Consolidation and harmonization of compute area clients/APIs: description of the harmonization opportunities</td>
<td>EMI Compute Clients task force</td>
<td>PM9</td>
</tr>
<tr>
<td>Consolidation and harmonization of compute area clients/APIs: definition of a detailed plan</td>
<td>EMI Compute Clients task force</td>
<td>PM11</td>
</tr>
<tr>
<td>Consolidation and harmonization of compute area clients/APIs: implementation of the plan</td>
<td>ARC Compute Clients PT, gLite job management PT, UNICORE Client and APIs PT</td>
<td>PM24</td>
</tr>
<tr>
<td>Identification of compute area components that will need to be migrated from GSI</td>
<td>GSIToSSL Task force</td>
<td>PM9</td>
</tr>
<tr>
<td>Identification of the points where the migration from GSI may encounter obstacles; identification of possible solutions</td>
<td>GSIToSSL Task force</td>
<td>PM12</td>
</tr>
<tr>
<td>Planning of GSI to SSL migration, with proper procedures for each component</td>
<td>GSIToSSL Task force</td>
<td>PM14</td>
</tr>
<tr>
<td>Removal of GSI from compute area components</td>
<td>ARC CE PT, ARC Compute Clients PT, gLite job management PT</td>
<td>PM24</td>
</tr>
<tr>
<td>Improvements and consolidation of the ARC-CE Janitor component</td>
<td>ARC CE PT</td>
<td>PM12</td>
</tr>
<tr>
<td>Provision of accounting hooks in the ARC CE</td>
<td>ARC CE PT</td>
<td>PM9</td>
</tr>
<tr>
<td>Provision of the proposal for a new data staging system in the ARC CE</td>
<td>ARC CE PT</td>
<td>PM12</td>
</tr>
<tr>
<td>Libarcclient language bindings available on the Windows platform</td>
<td>ARC Compute Clients PT</td>
<td>PM9</td>
</tr>
</tbody>
</table>
4.1. HARMONIZATION ACTIVITIES

In this section the planned harmonization activities are presented.

4.1.1 Definition and adoption of job related standard interfaces

The EMI components currently implement job description, job management and job status query in different, incompatible ways, even if the provided core functionality is mostly the same.
While standard mechanisms for job description and standard interfaces for job submission, management and status query do exist (namely BES and JSDL), they are not really suited for production use because they lack significant capabilities, such as:

- lack of standard security profile in relation to OGSA-BES
- missing functionality with respect to data staging for computation
- missing attributes in job description (e.g. for data requirements, to support epilogue/prologue, to support resubmission, to manage run time environments, etc.)

This issue is being addressed in parallel on two fronts:

- within the EMI JRA1 working group, where a specification for an EMI (also known as AGU: ARC, gLite, UNICORE) Execution Service (ES) is being defined
- In the PGI OGF working group [R28] where, besides the EMI partners, some other international Grid community partners are involved.

The ARC, gLite and UNICORE job submission product team people are participating and will keep participating in these activities.

In particular for what concerns the specification of the EMI Execution Service, a specific task force composed by 10 persons (ARC, gLite and UNICORE representatives) has been formed.

The activities of this task force are done in:

- Face to face meetings.
  Two workshops (one in Padova, 26-28 July 2010 and one in Budapest 22-24 September 2010) already took place.
- Weekly teleconferences (Thursday 9.30- 11:30).
- “Off-line” tasks assigned to specific members of the task forces.

The activities of this working group (including the detailed list of actions) are and will be tracked in the following twiki page:

https://twiki.cern.ch/twiki/bin/view/EMI/EmiExecutionService

This twiki page is also used as repository for the EMI ES specification draft documents.
The final version of such document, that is the specification of the EMI Execution Service, is planned to be ready by PM7 (milestone MJRA1.2).

As soon as this EMI specification is ready, it will be implemented in the EMI job management products (A-REX based ARC CE, CREAM CE, UNICORE TSI, UNICORE XNJS, UNICORE web service interfaces) and in the relevant clients (ARC Compute client, CREAM CE client, UNICORE HiLA and UCC). The plan is to finalize such implementations in the second year of the EMI project (milestone MJRA1.3, planned for PM18).

The EMI-ES specification will also be exposed to the OGF PGI working group, and it will possibly influence the specification being defined in such standard body.

Support for the existing middleware proprietary interfaces (gLite CREAM, UNICORE Atomic Services, ARC Proprietary Interface) will continue as long as needed for production activities (it is very unlikely that such support can be drop before the end of the EMI project).

4.1.2 Adoption of information service related standards

The well established relevant standard for the the specification of the characteristics and status of Grid entities is GLUE [R29].

The latest available specification is GLUE v. 2.0, which has been published as OGF Proposed Recommendation, and which is going to be widely deployed in the production Grid infrastructures.

Although most existing components of the A-REX based ARC CE, together with the ARC Compute clients, already comply in general with GLUE2, its adoption must be finalized: this will be done by PM10.

gLite job management services, instead, still refer to GLUE 1.3: work will be done to support in these job management services the v. 2.0 specification: this is also planned to be finalized by the first project year. In more detail the plan is:

- Glue 2.0 support in the CREAM CE. This includes the integration in the CREAM CE of the Glue 2.0 compliant LRMS specific information providers (which are not provided by the gLite job management product team). This task is planned to be finalized by PM10

- Adaptation of the matchmaker module of the Workload Management System to be Glue 2.0 compliant. This task is also planned to be completed by PM10

The current UNICORE version 6.3 already offers GLUE2 compliant information in XML format about the compute part through its Information Provider component and as part of the OGSA-BES implementation. However, since there is no official XML GLUE2 rendering available yet (this is being finalized in the OGF Glue WG), the UNICORE GLUE2 support will have to be adjusted once the official OGF GLUE2 rendering document is published. This is planned to be done by PM10, assuming that the official XML Glue2 rendering is available at least one month before.
4.1.3 Harmonization of the authorization mechanisms

Different authorization mechanisms, providing the same or similar functionality, are now used in the EMI job management services. This is clearly a complication from a deployment and maintenance point of view.

Moreover in some cases multiple authorization systems are even used within the same job management service, and this can bring to inconsistent authorization decisions: because of bugs or misconfigurations a certain Grid user could be authorized by a certain authorization service, while the authorization could be denied by another authorization component.

As already identified in the DoW document, these issues will be addressed referring to a single EMI authorization service, where the gLite Argus service [R30] will be the reference implementation.

The existing EMI job management services will therefore have to be properly integrated with such authorization framework in collaboration with the security area.

Since all security in ARC is handled by HED, and since HED will be integrated with ARGUS, all services hosted by HED (that is also A-REX) will automatically get integrated with it. This is planned to be finalized by the first project year. If needed, some other components of the ARC-CE will be modified to be Argus-aware.

The integration between the CREAM CE and the Argus authorization service will be finalized for the first EMI major release. In more detail the plan is:

- To define, in collaboration with the security group, a XACML authorization profile for the CREAM computing element, to get the meaning of each operation with respect to the security requirements and then provide the suitable set of policy rules to enforce. This was done at PM3.
- To implement in the CREAM CE this profile. In this way ARGUS will be the only system used within the CREAM CE for authorization and user mapping. This is planned to be finalized for the first EMI major release (for which the code must be finalized by PM10).

For what concerns UNICORE, it already has a clean, single XACML callout for making authorization decisions for each incoming web service call. In EMI, the authorization will be based on Argus. Thus, as a first step for integrating Argus support into UNICORE, an evaluation of Argus will be performed during the first project year, in close collaboration with the UNICORE security team, in order to check whether Argus serves the same purpose and has the same scope as the current XACML policy check. Such evaluation is expected to be finalized by PM8.

In case Argus is suitable, support for Argus will be added to UNICORE by the UNICORE security PT (and therefore such activity is not reported in detail in this deliverable document which covers just the compute area). Otherwise the missing functionality will be identified (so that they can be eventually implemented in ARGUS).

4.1.4 Harmonization in the parallel and MPI job support

The current support for the execution of parallel and MPI jobs is quite different in the existing EMI job management services.
In the ARC CE the middleware is just responsible to allocate the needed resources, and then the execution of the parallel job is fully managed by specific RunTime Environments (RTEs) which are set up at sites and advertised in the information system (see Sect. 3.1).

Something similar applies to the gLite job management services: once the resources needed for the execution of a parallel job have been allocated by the Computing Element, MPI-Start provides a unified layer for starting the parallel application, hiding the underlying details of the resources and MPI frameworks to the upper middleware layers and to the users.

UNICORE offers its users a easy-to-use and completely transparent way of submitting jobs to MPI and other types of parallel environments such as OpenMP through so called “execution environments”. The site administrator describes the existing parallel environments in a configuration file, since they are typically very heterogeneous and as such tough to be covered by a general implementation framework. This file is used by UNICORE in two ways: on the one hand, to advertise the parallel environments to the clients. On the other hand, it prescribes how UNICORE should setup and start the parallel job. The UNICORE client software thus offers convenient ways to create parallel jobs, without requiring the user to know any site specifics.

Some harmonization in this area is definitely needed. The description for parallel and MPI jobs (along with the description of the needed resources) will be standardized in the context of the job related standardization activities (see Sect. 4.1.1). The standardization of the “back-end”, which is the provision of a standard EMI unified layer, targeted to the end user and responsible to manage the parallel application hiding the underlying details, will be assessed during the first project year.

A small task force composed by three persons (i.e. one representative per each middleware stack, ARC, gLite, UNICORE) has been formed and the following first steps were decided:

- In order to better understand the differences and common things in the three middlewares, develop and circulate examples for submitting simple "hello world" MPI job with precompiled binary, source code that gets compiled at the resource and complex application like gromacs.
- Study what it should be done to provide transparent user experience in all systems.
- Findings of the exercise to achieve a "transparent user experience in running MPI jobs" to be presented at the November 2010 all-hands meeting.

New plans will be defined after this all-hands meeting.

### 4.1.5 Consolidation and harmonization of compute area clients/APIs

Besides the harmonization of job related standard interfaces (see Sect. 4.1.1) some consolidation and harmonization is also needed for the clients.

During the first project year it will be investigated which kind of harmonization and consolidation can be done and concrete plans (to be then implemented in the rest of the first project year and in the second year) will be settled. The main objectives are to improve:

- usability
- maintainability
portability to different architectures and operating systems

A specific task force composed by three persons (one per middleware stack) has been formed to address such objectives. This task force will produce:

- An evaluation of the current state of the clients in the middlewares taking into account the above mentioned main objectives. This is expected to be finalized by PM7
- A description of the harmonization opportunities taking into account results or work in progress of other task forces or product teams. This is expected to be finalized by PM9
- A detailed plan. This will be finalized by PM11
- This plan will then be implemented in the relevant compute client tools. This is expected to be done by PM24

The activities of this task force are and will be tracked in:

https://twiki.cern.ch/twiki/bin/view/EMI/EmiJra1T2Compute_Client

4.1.6 Investigate possible use cases for a common standard messaging system in the computing area

The infrastructure area of the EMI project currently investigates a solution for messaging to be used within the greater EMI architectural framework.

During the first year of the project, the investigation of how such messaging infrastructure can be useful for the services of the compute area will start (the goal is to finalize such investigation by the mid of the second project year). As a result of such investigation, a set of use cases for which this common messaging system is assessed as useful, will be identified.

Such work will start after the messaging workshop that will take place before the November 2010 All-hands meeting, where the foreseen scope and functionality that will be provided by the EMI messaging system will be clarified.

4.1.7 GSI removal

An overall strategy of the EMI project is to remove the Globus proprietary GSI protocol in the EMI components and replace it with standard SSL/TLS readily available in target operating systems. This involves also the compute services (apart the UNICORE ones, which are not using GSI already). Since the move to pure SSL/TLS is not compatible with the legacy GSI, this transition must be carefully planned such that operations are not impacted.

A task force, lead by the security area and composed also by members of the compute area, was formed to properly organize such GSI removal.
The activities of such task force are and will be tracked in the following twiki page:

https://twiki.cern.ch/twiki/bin/view/EMI/EmiJra1T4GSISSLMigration

In more details, the work plan is:

- To get the list of components that will undergo migration. This is planned to be finalized by PM9.
- To find the points where the migration may encounter obstacles and find possible solutions. This is planned to be done by PM12.
- To plan the migration, with proper procedures for each component. This is planned to be done by PM14.
- To perform the actual removal of GSI. This is planned to be finalized by PM24.

4.2. EVOLUTION PLANS

In this section the planned activities targeted to evolutionary new developments are described. The activities described in this section are targeted to meet requirements raised by the relevant user communities and to address some problems in the existing implementations of the job management services.

4.2.1 Evolution plans for the ARC CE

For what concerns the developments in the ARC CE, the following new areas will be active in the first year.

Dynamic management of RTEs by ARC CE is a feature requested by several users; Janitor component allows achieving such requirements. However, it currently exists as an early prototype, and requires substantial hardening before being released. To do this, the RTE management environment Janitor has to be deployed over the EMI testbed, and undergo similar testing procedures as other components. Discovered problems will have to be promptly fixed. It is difficult to assess at the moment how much work it will take, but it is expected Janitor to become stable enough for a release by the end of the first project year (PM12).

Development of accounting hooks in the ARC CE will depend on the overall progress in the accounting and monitoring work performed in the infrastructure area through dedicated task forces. By PM9 accounting hooks will be implemented in the ARC CE. If in the context of the mentioned task forces it won't be clarified in time a clear roadmap, accounting hooks for at least SGAS will be provided.

Very important components in need of serious evolution are those responsible for ARC CE data staging. This is because the existing model based on downloaders and uploaders has a number of shortcomings that negatively affect ARC CE throughput and is in general rather simplistic and inflexible. For example the existing model doesn't allow the handling of priorities between different users or VOs, or other optimizations [R44]. By PM12 it is expected to deliver the proposal for a new
data staging system for the ARC CE, which will replace current downloaders and uploaders. This is a pure research and development activity, not involving any maintenance at this stage. Later in the year, it is planned to start implementing the proposal.

4.2.2 Evolution plans for the ARC compute clients

For what concerns the ARC Compute Clients product team, the following new areas will be active in the first year.

Additional enhancements in form of extending the resource discovery module and enhancing the job submission interface are improvements which speed-up the functionality of the library, while extending job migration support will widen the functionality. These enhancements will inevitable cause changes, which need to be propagated to third party developers relying on the library, in particular the ArcJobTool GUI developers. Similarly it may be necessary to adjust libarcclient to satisfy the needs of these parties, e.g. the libarcclient language bindings (Python and Java).

The libarcclient language bindings provide an interface to the library for non C++ programming languages (Python and Java). Since libarcclient is fully supported on non-linux platforms (e.g. Windows), the language bindings should work similarly well on these. Efforts will be invested to make the language bindings available on the Windows platform as well. It is foreseen that language bindings on Windows will be available by PM9.

An important feature of libarcclient is a functional data broker which is able to select the most appropriate target for job execution, thus limiting the expensive time spent on transferring data requested by a job. It is planned that the data broker module in libarcclient should be enhanced and extended in particular to support the wide area of storage protocols and information systems. Investigation on the requirements and the needed effort will be done; a more detailed development plan for such a broker will be ready at the end of PM7. The improved data broker is then expected at PM9.

An interesting extension for the libarcclient, is to make it able to support job management for non Grid resources, through the development of dedicated modules. More specifically it would be very useful to be able to test job executions on the local machine in order to subsequently improve the rate of successful executions when the submissions are done on a large scale. Requirements and effort needed for creating such job management modules for libarcclient will also be investigated at the end of PM11.

4.2.3 Evolution plans for the gLite job management services

The gLite job management services have already been used in production environments, some new functionality addressing issues already raised by users and administrators have already been identified (see below). Some others will be raised by existing and new user communities during the duration of the project, and it will be important to be able to promptly address such requests.

Among the already identified area where work is needed, there is the design and implementation of general feedback mechanisms. Because of different reasons (such as different site policies, different VO priorities, high usage of site resources, etc), scenarios for which a job is stuck in the queue for too long, waiting for an available job slot, can happen. This means that the job would start its execution many hours or days after its submission. On the other hand free resources could be available in other sites. The issue is even worst for those computations which can be split in uniform and independent jobs (which can therefore be executed in parallel), which however must be all successfully completed
to consider the overall computation as completed. This is for example a typical use-case for the CDF HEP experiment (who was the first but not the only one raising this issue): using the CDF analysis framework (CAF), the data set to be processed is split in several independent "sections" which can then be processed in parallel but which must be all processed to consider the overall analysis task as completed. For such scenarios a Minimum Completion Time (MCT) dynamic scheduling policy, where not even a single job should lay in non-terminal Grid states, is therefore required.

Considering the gLite job submission environment where jobs are submitted to computing elements though the Workload Management System, this means the implementation of mechanism to migrate jobs stuck in blocking queues. In more detail the plan is:

- Implementation of a prototype WMS able to migrate to different CEs jobs stuck in the queue for too long (waiting for an available executing node). A fixed (but configurable) timeout will be considered. This is planned to be finalized by PM6
- Validation of this prototype by selected users (in particular by representatives of the CDF experiment). This should be done by PM8
- Finalization of this first implementation of the WMS with feedback mechanisms. This will be provided with the first EMI major release.

Further evolutions (in particular the timeout triggering job migration will be dynamic, according to the status of the Grid) will be implemented during the second year of the EMI project. They will also depend on new functionality to be provided by other components (in particular the LB service).

Another area where work is needed is the improvement of parallel and MPI support. Although the EGEE projects (the projects where the gLite middleware was developed and used) were initially focused on the HEP community, other communities were added later, such as chemistry, biology, medical imaging, etc. In these communities the use of parallel programs is quite common. While these jobs are already supported by the gLite job management services, there are still some open issues, such as the inability to fully describe and allocate accordingly the resources in a multi-core environment. The EGEE MPI WG at the end of the EGEE-III project produced a document [R31] with a set of recommendations for the middleware, to address such shortcomings. Such requests will be implemented in the gLite job management services in the first project year. In particular the plan is:

- Implementation of a prototype providing support for the new required functionality through scripts to be plugged within the CREAM CE, exploiting the functionality described at [R45]. This is planned to be finalized by PM6
- Validation of this prototype (to be done by selected user communities, in particular some who asked for the mentioned functionality). This is planned to be finalized by PM8.
- Finalization of the implementation in the CREAM and BLAH components. This is planned to be ready for the first EMI major release.

4.2.4 Evolution plans for gLite-MPI
As specified in Sect. 4.1.4, the goal is to provide a standard EMI unified layer for the management of parallel applications. In the meantime the current solutions need to be maintained.
There are three main identified areas in which work is needed for the gLite MPI tools:

- Implementation of better control of user processes. User should be able to specify how the logical processes will be mapped to physical resources, taking profit of multi-core architectures. This is planned to be finalized by PM8.

- Better plugin (hooks) management in MPI-Start. The current architecture enforces the sites to modify the source code of MPI-Start in order to implement new plugins, a new architecture should facilitate the development of such hooks. This is planned to be finalized by PM10.

- Enhancement of scheduler module: better support for the current provided modules (PBS, SGE and LSF) and introduction of support for new schedulers, like Condor and SLURM. This is planned to be finalized by PM12.

### 4.2.5 Evolutions plans for UNICORE job management components

For what concerns the UNICORE Target System Access, a new development foreseen for the first year is a more flexible generation of site-specific scripts. In many environments, it is necessary influence the script created by XNJS based on the current user's attributes and the current request. For example, it might be needed to select the batch queue, influence resource limits, UNIX primary and secondary groups depending on the application that the user is running.

This work is planned to be finalized for the first EMI major release.
5. CONCLUSIONS

This document describes the first year work plan for the compute area of the EMI project. This plan is based on the best information at the time and the general plan set out in the parallel overall Technical Development Plan document.

The workplan will be updated at the end of the first project year (PM12) and at the end of the second year of the project (PM24). The updated workplans will be presented respectively in deliverables DJRA1.1.2 and DJRA1.1.3.