Open Standards-based Interoperability of Job Submission and Management Interfaces across the Grid Middleware Platforms gLite and UNICORE

Moreno Marzolla, Paolo Andreetto
Instituto Nazionale di Fisica Nucleare
Padova
35131 Padova, Italy

Valerio Venturi, Andrea Ferraro
Instituto Nazionale di Fisica Nucleare
CNAF
40127 Bologna, Italy

Shiraz Memon, Shahbaz Memon, Bastian Twedell, Morris Riedel, Daniel Mallmann, Achim Streit
Central Institute of Applied Mathematics
Forschungszentrum Juelich
D-52425 Juelich, Germany

Svan van de Berghe, Vivian, Li, David Snelling
Fujitsu Laboratories of Europe
Hayes
Hayes, Middlesex UB4 8FE, UK

Katerina Stamou, Zeeshan Ali Shah, Fredrik Hedman
Center for Parallel Computers
Royal Institute of Technology
SE-100 44 Stockholm, Sweden

Abstract

In a distributed Grid environment with ambitious service demands the job submission and management interfaces provide functionality of major importance. Emerging e-Science and Grid infrastructures such as EGEE and DEISA rely on highly available services that are capable of managing scientific jobs. It is the adoption of emerging open standard interfaces which allows the distribution of Grid resources in such a way that their actual service implementation or Grid technologies are not isolated from each other, especially when these resources are deployed in different e-Science infrastructures that consist of different types of computational resources. This paper motivates the interoperability of these infrastructures and discusses solutions. We describe the adoption of various open standards that recently emerged from the Open Grid Forum (OGF) in the field of job submission and management by well-known Grid technologies, respectively gLite and UNICORE. This has a fundamental impact on the interoperability between these technologies and thus within the next generation e-Science infrastructures that rely on these technologies.

1. Introduction

In the last couple of years, many Grid and e-Science infrastructures have begun to offer production services to end-users (*e-Scientists*) with an increasing number of scientific application projects that require access to different types of computational resources in multiple Grids.

Today, the *Distributed European Infrastructure for Supercomputing Applications (DEISA)* [5] provides Grid services to e-Scientists via the *UNICORE 5 Grid middleware* [43]. In the context of job submission and monitoring, the non Web services-based UNICORE 5 is used within this infrastructure to provide e-Scientists with a proprietary interface to run *massively parallel scientific jobs* on *High Performance Computing (HPC)* resources like supercomputers.

Another well-known e-Science infrastructure is the *Enabling Grids for e-Science (EGEE)* [6] project that provides various Grid services for end-users via the *gLite Grid middleware* [32]. The non Web services-based gLite middleware is used within this infrastructure to submit mostly *embarrassingly parallel scientific jobs* on a wide variety of Grid resources such as clusters and PC pools of many *Virtual Organizations (VOs)* [28] that are part of EGEE.

0-7695-3064-8/07 $25.00 © 2007 IEEE
DOI 10.1109/e-Science.2007.38

One of the above described application projects that require cross-Grid access between these well-known e-Science infrastructures is the *Wide in Silicio Docking on Malaria (WISDOM)* project [17]. It aims at developing new drugs for Malaria and the WISDOM scientists use the resources within EGEE for large scale in silicio docking. In more details, they use a computational method for the prediction of whether one molecule will bind to another using the *AutoDock* [35] and *FlexX* [38] software. Both software stacks are provided by the gLite middleware within EGEE and the output of these applications is a list of chemical compounds that may become potentially drugs. Hence, this list is not the final compound list, because it must be refined using *molecular dynamics (MD)*. These MD computations use the highly scalable *assisted model building with energy refinement (AMBER)* [23] code that could run on HPC resources within DEISA. Hence, cross-Grid usage lead to the benefit of significantly accelerating the drug discovery step.

However, e-Science infrastructures such as DEISA and EGEE are currently *not technically interoperable* which means that no scientists can leverage the different types of Grid resources within both of them for one application. This gap of interoperable cross-Grid technologies is a major drawback for many e-Science projects (e.g. WISDOM project) that require the access to the different types of Grid resources as provided within DEISA and EGEE. Therefore the European Commission has set up another project named as the *Open Middleware Infrastructure Institute for Europe (OMII-Europe)* [13] that tries to provide a wide variety of interoperable components for UNICORE and gLite to enable technical interoperability among DEISA and EGEE in the near future. This paper reveals the adoption of job submission and management standards recently emerged from the *Open Grid Forum (OGF)* [14] within the new *Web services-based UNICORE 6* [16] and the *CREAM* [20] system of gLite. Also, this paper addresses known security challenges during interoperability between UNICORE and gLite and presents a technical solution for the realization of job interoperability between DEISA and EGEE, enabling scientists to securely submit jobs to both infrastructures.

This paper is structured as follows. After reviewing the challenges in interoperability in the context of job submission and management within well-known e-Science infrastructures, Section 2 introduces emerging standards of OGF in this context. Section 3 describes the adoption of these standards by the Grid middleware system UNICORE and the CREAM-BES system of gLite. The security issues of both systems are discussed and solutions are presented in Section 4. Based on the adopted technologies, Section 5 describes the achieved interoperability between UNICORE and CREAM-BES of gLite in the context of job submission and management. Finally, after surveying related work in Section 6, the paper ends with concluding remarks.

2. Job Submission and Management Standards

There are several open standards emerging from the OGF in the context of job submission and management. First and foremost, the *Job Submission Description Language (JSDL)* [21] defines standard XML schema elements for job descriptions that should be executed on computational Grid resources, for instance a simple execution of a *POSIXApplication*. The main JSDL schema is extensible and thus there are several application extensions evolving that add several standard schema elements to JSDL for specific application domains, for instance the parallel *Single Program Multiple Data (SPMD)* application extension [41].

Another set of elements that expand the basic schema of JSDL are the *application extensions for HPC scenarios* [31]. This schema basically shares much in common with the JSDL *POSIXApplication*, but is tailored for HPC environments and also removes some of the features of JSDL that present barriers to interoperability by using the XML element *jsdl-hpcpa:Executable* and other similar elements.

So far, we only listed languages and schemas for the standardized description of computational job submissions, but the submit operation itself should also be represented by a standard interface. Therefore, the *Open Grid Services Architecture - Basic Execution Services (OGSA-BES)* specification [12] describes an 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 names as *activities* described via JSDL.

In more detail, the OGSA-BES interface consists of three standardized services. First and foremost, the *BES-Factory Service* is responsible for the creation and control of a set of activities described with JSDL documents. To provide an example, the *CreateActivity(JSDL)* operation of the BES-Factory Service leads to the creation of a computational job on the computational Grid resource. This job can be controlled and monitored by using the *BES-Activity Service* operations, for instance by using the *Terminate()* or *GetStatus()* operations. The third service of the OGSA-BES interface definition is the *BES-Management Service* that can be used for remote administration of the service behavior. That means the BES-Management Service provides operations such as *StopAcceptingNewActivities()* and *StartAcceptingNewActivities()*.

Finally, the *HPC Basic Profile (HPC-P)* specifies the usage of the OGSA-BES interface in conjunction with the above described extensions for HPC environments to JSDL. The HPC Basic Profile relies on the *WS-I Basic Security Profile* [33] for interoperable message security. That means security in terms of the *Web Services Security UsernameToken Profile* [37] and *TLS/SSL using X.509 Certificate Based Mutual Authentication* according to RFC2246 [26].

3. Adoption of Job Submission and Management Standards in Grid Technologies

This section describes the adoption of the OGSA-BES interface and JSDL specification schemas by the Grid middleware systems UNICORE and CREAM-BES of gLite. Both systems also integrate the proposed extensions to JSDL to reach the *HPC-P compliance* and thus being interoperable with each other on the portType level. While different security setups within the context are described later in the paper, this section provides an overview of how the HPC-P adoption is realized in the different technologies.

3.1. CREAM-BES of gLite

An efficient and robust system for accessing computational resources and managing job operations is a key component of the gLite Grid middleware that supports large distributed computing environments, for instance the EGEE infrastructure. The *Computing Resource Execution and Management (CREAM)* [20] is such a system designed to provide efficient processing of a large number of requests for computation on managed resources. CREAM accepts requests from distributed clients using Web services technologies and its architecture is designed to be robust, scalable and fault tolerant. As shown in Figure 1, CREAM is integrated within gLite and works together with the well-known *workload management system (WMS)* [24].

In more detail, CREAM is a Java application running as an extension of a Java AXIS [2] servlet inside a Tomcat application server [3]. CREAM contains a separate thread named as the *Journal Manager* [20] that submits requests to the local *resource management system (RMS)* through the *Basic Local Ascii Helper (BLAH)* [20] component, which represents an abstraction layer for the underlying RMSs. The CREAM backend is a permanent memory space used to store the data related to the jobs that are currently managed within the system. In the context of the overall gLite Grid middleware, the CREAM functionality can also be used by the gLite WMS component. In fact, the jobs submitted to the gLite WMS can be forwarded for their execution on CREAM based *Computing Elements (CEs)*. This is useful, because the WMS comprises a set of components responsible for the distribution and management of jobs across Grid resources in such a way that applications are conveniently and effectively executed. This integration is being done by a separate module named as *Interface to Cream Environment (ICE)* [20] that receives job submissions and other job management requests from the WMS component and afterwards uses the appropriate CREAM methods to perform the requested operation on the CE.

Figure 1 illustrates that the OGSA-BES interface implementation within CREAM, collectively named as CREAM-BES, is also deployed within the same container as CREAM and thus represents a standardized interface to the legacy CREAM system. Hence, instead of accepting job descriptions in the *Job Definition Language (JDL)* format of gLite used by the WMS, the CREAM-BES is deployed in parallel to the legacy WS interface and accepts JSDL compliant job descriptions. CREAM-BES accepts JSDL documents that consists of HPC-based extensions [31] to JSDL and thus CREAM-BES is HPC-P compliant. To sum up, CREAM-BES provides a standard interface to the legacy CREAM.

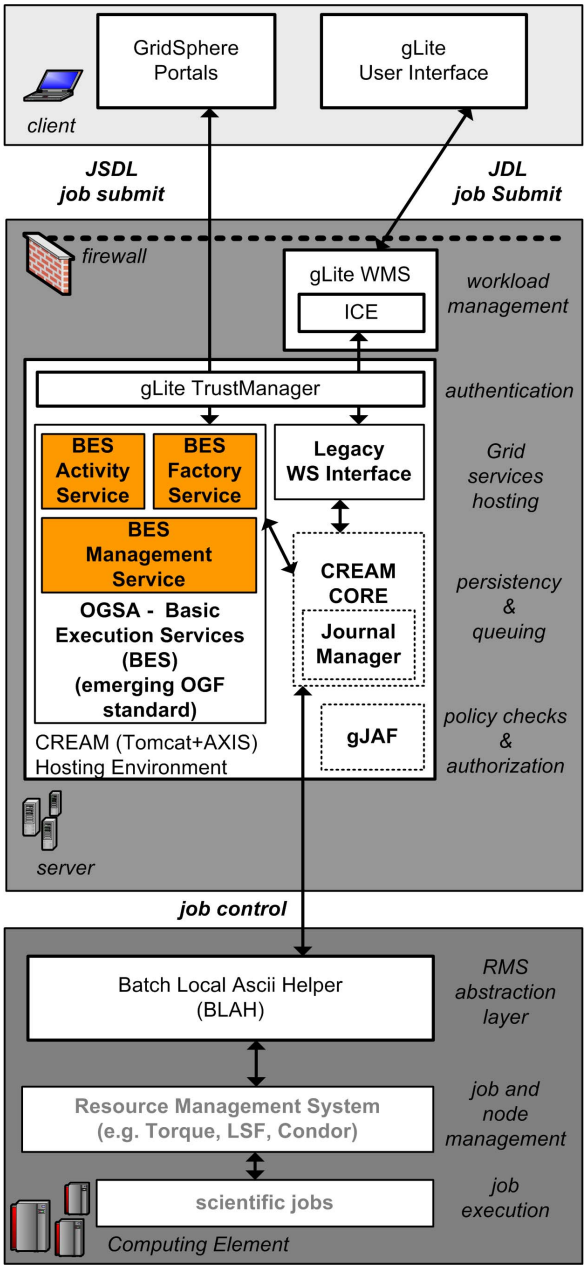

Figure 1. CREAM-BES of gLite.

3.2. HPC-Profile adoption of UNICORE

The *Uniform Interface to Computing Resources (UNICORE)* Grid middleware [43] provides a seamless, secure, and intuitive access to distributed Grid resources such as supercomputers, clusters, or server-farms. The non Web services-based UNICORE 5 middleware is used in daily production at world-wide supercomputer centers and research facilities. Beyond this production usage, e.g. within the European DEISA infrastructure [5] or the German National Grid D-Grid [7], it serves as a solid basis in many European and International research projects (e.g. in OMII-Europe [13], A-Ware [1], Chemomentum [4]).

More recently, the new Web services-based UNICORE 6 becomes available that is based on open standards such as *Web Services Resource Framework (WS-RF)* [11] and *Web services Addressing (WS-A)* [18] just to list but a few. All in all it conforms to the *Open Grid Services Architectures (OGSA)* [29] of the *Open Grid Forum (OGF)*. This section describes the interface integration of some of these emerging open standards related to job submission and management such as OGSA-BES and JSDL into this new version of UNICORE that is HPC-P compliant. As the successor system of UNICORE 5, DEISA and D-Grid have already stated interest in analyzing the features of UNICORE 6 for production usage.

Figure 2 provides an overview of UNICORE 6 with a focus on job submission and management. In detail, the implementation of the OGSA-BES interface is similar to the proprietary *Target System Service (TSS)* and *Job Management Service (JMS)* of the *UNICORE Atomic Services (UAS)* [40]. The UAS provide a WS-RF compliant interface layer to underlying resource management capabilities such as job management and file transfer. While the UAS rely on standardized WS-RF compliant message exchange, the syntax of the UAS operations are still proprietary such as the actual *submit()* operation that takes a JSDL document as parameter. Hence, the additional benefit of OGSA-BES interfaces for UNICORE 6 is a provisioning of a standardized syntax of Web services operations.

In more detail, the UAS and the OGSA-BES implementation are using the same *enhanced NJS (XNJS)* [42] as execution backend within UNICORE. Hence, the Web service layer is well encapsulated from the lower-level execution engine layer, but both are deployed within the same hosting environment based on Jetty [10] (using the XFire SOAP engine [19]). All OGSA-BES invocations are transmitted to the server-side in the SOAP body, except several pieces of information wihin the SOAP header that are used to address job WS-Resource instances [11] and security tokens.

As shown in Figure 2, the integrated OGSA-BES interface of UNICORE 6 consists of three standardized services. First and foremost, the *BES-Factory Service* is responsible for the creation and control of activities described with JSDL. The invocation of its *CreateActivity(JSDL)* operation leads to the creation of a job resource within the enhanced NJS that represents a computational job described by the JSDL. The underlying enhanced NJS accepts JSDL with extensions [31] required by the HPC-P and thus UNICORE is HPC-P compliant. This JSDL is converted to *Resource Management System (RMS)* (e.g. Torque, LoadLeveler) specific commands on the target system. On the Web services level, the job resources can be controlled and monitored by using the *BES-Activity Service*, for instance by using the *Terminate()* or *GetStatus()* operations. The *BES-Management Service* can be used to control whether new jobs can be submitted or not.

Finally, UNICORE 6 supports the *WS-RF rendering* of the OGSA-BES specification [12], which includes that OGSA-BES activity properties of the *internal WS-RF resource model* are provided and supported by UNICORE 6. Hence, beside the mandatory operations defined in OGSA-BES, UNICORE 6 also provides operations according to the *WS-Resource properties* [11] for standardized property requests of jobs submitted through the OGSA-BES interface.

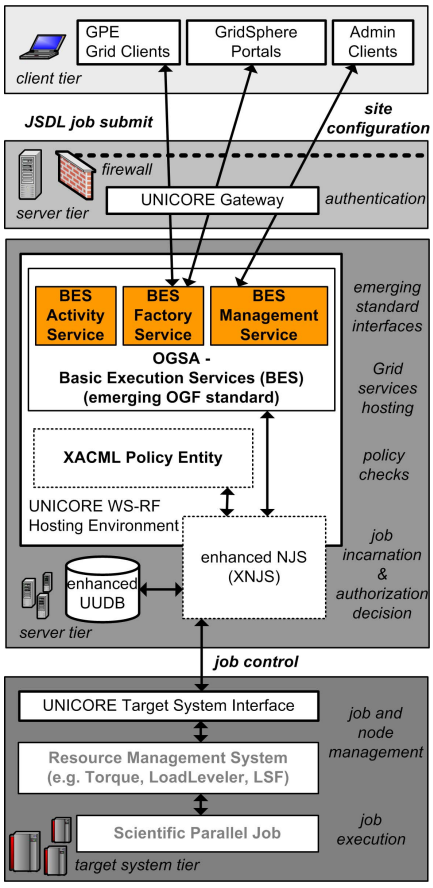

Figure 2. UNICORE HPC-P Profile adoption.

4. Security Considerations and Solutions

The OGSA-BES and JSDL specification focus on job description, submission and management and thus security considerations have been kept out of these specifications. The HPC-P on the other hand describes several security specifications to enable basic interoperability. This comprises the *WS-Security Username Token Profile* [37] specification by using *HTTP/SSL (HTTPS)* encrypted connections.

Of course, this is a very simple approach only used during demo purposes of the OGSA-BES interface itself (i.e. Supercomputing 2006 demonstrations) and thus this security set up is not suitable for large-scale deployments within the DEISA or EGEE infrastructure. Even if the most Grid middlewares provide their own security models, OMII-Europe currently develops a *common security profile* that defines several security specifications and technologies that should be used during interoperability scenarios. To understand this profile, we review the *authentication (authN)* and *authorization (authZ)* mechanisms recently developed for CREAM-BES of gLite and UNICORE 6.

UNICORE 6 uses the *UNICORE Gateway* [34] for Authentication decisions by the means that each certificate of end-users are checked if they are signed by a trusted CA, valid and not revoked. CREAM-BES of gLite uses the *gLite Trustmanager* [32] for authentication. The TrustManager is external to CREAM and is an implementation of the J2EE security specifications slightly modified to work with gLite. Similar to the UNICORE gateway, the TrustManager checks if the certificates of the end-users are signed by a trusted CA, valid and not revoked.

In contrast to the authentication, the authorization decisions of both systems are much more complex and rely on the recently developed new *Virtual Organization Membership Service (VOMS)* [25], which releases signed *Security Assertion and Markup Language (SAML)* assertions [22] as shown in Figure 3 (1). The SAML assertions released by the new VOMS server indicate project/group membership or role possession. In particular, the SAML assertions are transmitted during a Web services operation call within the SOAP header using standardized Web service Security message extensions (2) while the invocation of the OGSA-BES related functionalities are places within the SOAP body. Hence, the VOMS SAML assertions are actually used during Web service invocations.

In UNICORE, the initial step during the authorization decision within the enhanced XNJS of UNICORE 6 is to check the VOMS attributes within a SAML assertion by using *Extensible Access Control and Markup Language (XACML)* [36] based callouts (3). The UNICORE 6 XNJS acts as a *Policy Enforcement Point (PEP)* and we use an XACML policy callout as a *Policy Decision Point (PDP)* for authorization that checks the attributes of SAML assertions and grant or deny access. Another *Policy Decision Point (PDP)* is the *UNICORE User DataBase (UUDB)* [16] that checks if the correspondent user is generally allowed to access the correspondent computational resource. In addition the UUDB is used to apply the credential mapping from certificate identities to real *xlogins* that exist on the correspondnent computational resources.

Within gLite, the authorization is performed by the *gLite Java Authorization Framework (gJAF)* (3), a pluggable component of the gLite distribution. The SAML assertion is evaluated by a plugin of that framework using its VOMS attributes to grant or deny access. In more detail, the assertions are extracted from the header of the SOAP message and afterwards they are validated as they must be signed. Then the VOMS attributes are retrieved from the SAML assertion itself. If the attribute matches one item in a given list of roles/groups then the correspondent OGSA-BES operation can be performed.

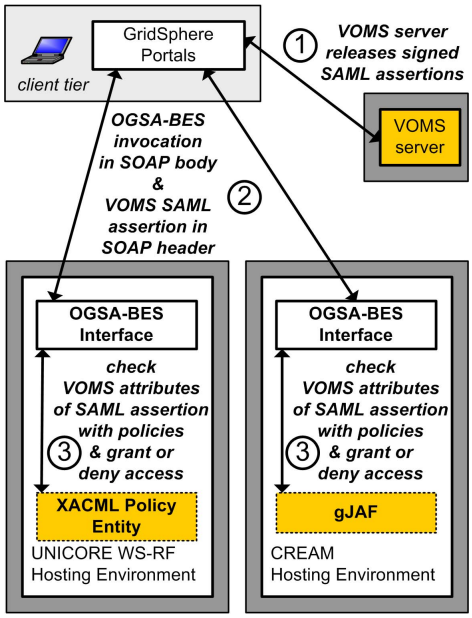

Figure 3. OGSA-BES invocations using VOMS SAML-based authorization.

To sum up, both CREAM-BES of gLite and the OGSA-BES interface of UNICORE 6 supports the common security profile of OMII-Europe, which includes the support of X.509 proxy certificates [44] as well as support for VOMS authorizations that are based on SAML [22]. Hence, apart from interoperability on the portType level as described earlier, this lays the foundation for secure interoperability of both systems. This security set up is more likely to be deployed in production e-Science infrastructures than the *WS-Security Username Token Profile* [37] used during SC2006.

5. Interoperability Achievements

The previous sections illustrated the adoption of the HPC-P profile by the gLite and UNICORE 6 Grid middleware systems and thus both are interoperable on the portType level. That means the same client can invoke an operation of OGSA-BES within UNICORE or gLite without any difference using a HPC-P compliant JSDL. However, for the integration of the OGSA-BES interface in real production e-Science infrastructures such as DEISA or EGEE, the security setup is of major importance as well. Therefore, a security setup such as defined within the common security profile within OMII-Europe allows us to perform secure interoperability among services within gLite and UNICORE. To understand more precisely the benefits of using Web services-based open standards for job submission and management, we discuss the improvements over the legacy interfaces in UNICORE and gLite in the context of the newly developed *native interoperability capabilities*.

UNICORE 5 used the proprietary protocol *UNICORE Protocol Layer (UPL)* [43] for message exchanges between clients and UNICORE 5 servers. Instead, the newly developed UNICORE 6 Grid middleware is based on theWS-RF for message exchanges and thus it is much easier to achieve interoperability with other WS-RF compliant middlewares such as the *Globus Toolkit 4 (GT4)* [27]. Also, the Web services-based UNICORE 6 makes it easier to achieve interoperability with other Web services-based systems such as CREAM-BES of gLite, because of similar addressing and message exchange mechanisms (i.e. SOAP [30]).

Most components of gLite like the WMS or the gLite UI works with the proprietary JDL [32] job description language. Hence, this limits interoperability with other Grid middleware systems that all have developed their own job description languages. To provide an example. the Globus Toolkit GRAM component [27] for job submissions accepts jobs described in the *Resource Specification Language (RSL)* [27] and UNICORE 5 uses proprietary *Abstract Job Objects (AJOs)* [43] for job descriptions. Hence, the new support in CREAM-BES of gLite that relies on JSDL significantly improves the interoperability capabilities with other JSDL-based systems such as UNICORE 6.

However, even if UNICORE 6 provided JSDL support already, the actual syntax of the interface implementation that accepts standardized JSDL job descriptions is still proprietary and named as the UAS [40] in UNICORE. Therefore, the OGSA-BES implementation within UNICORE described here fills this gap and provides the OGSA-BES interface with a standardized syntax for operations that are capable of accepting JSDL job descriptions. Because CREAM-BES follows the same interface specification both are interoperable on the portType level, which does not imply interoperability in terms of security mechanisms.

Hence, another important cornerstone of interoperability is the security as described in detail in the previous section. The *newly developed (optional) support for proxy certificates* within UNICORE makes it possible to interact with proxy-based gLite components such as CREAM-BES. Also, while UNICORE 5 does not support VO-based access and management, production gLite uses the former attribute-certificate-based VOMS server [25]. This means, a VO-based job interoperability was basically not possible until now. Therefore, to enable more sophisticated ways of interoperability with role and group/project membership relationships, the OMII-Europe project defined as one part of its common security profile the support for SAML-based VOMS authorization. Hence, both UNICORE 6 and CREAM-BES of gLite provide support for this technology and thus both are not only interoperable on the portType level, but also interoperable in terms of a sophisticated security mechanism based on the open standard SAML [22]. In fact, not only VOMS is based on SAML, but also Shibboleth [15] and thus it became more easier to support Shibboleth-based authorization in the future as well since it basically uses the same open standard mechanism.

All in all, WISDOM (see Section 1) scientists can use the same client to submit a job to gLite and use the same credentials and output from this job to submit a job using the UNICORE job interface because both systems follow the same open standards specifications for job submission and management. In addition, both follow the same security profile enabling not only secure job submission and management, but also secure access to other services and single sign-on. Many client tools such as the Intel *Grid Programming Environment (GPE)* [39] client suite or *GridSphere portals* [9] benefit from the standardized interfaces for job submission and management. In particular, these HPC-P compliant systems can be used from clients with the same security setup (X.509 proxy certificates and SAML-based assertions) and thus both systems provide access to their Grid infrastructures via single-sign on. The native interoperability of these systems makes it more easy to develop clients on top of these systems that can leverage Grid resources of different types.

Finally, it must be mentioned that even with these standardized interfaces and security solutions, there is still a demand for interoperability in the context of missing open standards for data staging technologies. OGSA-BES only focuses on simple job executions that are described by JSDL documents, while a standardized interface for storage and file transfer is still missing. This gap is filled with proprietary solutions such as with the *Storage Management Service (SMS)* [40] and *File Transfer Service (FTS)* [40] in UNICORE or the various file transfer techniques [32] in gLite. However, such services are still needed for JSDL-based cross-Grid job executions that rely on staged data.

6. Related Work

Work within the *EGEE project* [6] is based on the interoperation between UNICORE 5 and production gLite to enable the systems to simply interoperate without using any job submission and management standard. This short-term achievement works with a specific version of UNICORE and with a specific version of gLite, while we provide here a more long-term solution by using common open standards.

In the last year, many middleware providers have augmented their system with an OGSA-BES interface to participate in the *Supercomputing 2006 demonstrations*. More precisely, most of them even provide HPC-P support, which includes the usage of the OGSA-BES interface with JSDL. However, the major difference with our work is a more stable implementation of the OGSA-BES standard after the public comment period of OGF, while most of the older implementations are still based on the version 26 of OGSA-BES. Hence, for the HPC-P demonstration at Supercomputing 2006, it was decided to temporarily freeze the specification draft 26 so that implementers could have the time to implement that version of OGSA-BES. Another major difference to these efforts that use UsernameToken profile security is a security model better suited for production e-Science infrastructures such as EGEE and DEISA.

Another well-known implementation of the OGSA-BES interface is provided by the *OMII-UK software stack* and names as the *Grid Job Submission and Monitoring Service (GridSAM)* [8]. It provides Web service interfaces for submission, monitoring and execution of JSDL jobs and offers an interface compliant with OGSA-BES. The core of GridSAM is the *Job Management Library (JML)*, which encapsulates message exchanges mechanisms from JSDL to the RMS-specific commands of the underlying batch system. The JML is used for orchestrating the execution of so called *Distributed Resource Manager (DRM)* - Connectors that are a reusable set of components comprising job management functions. It is similar to OGSA-BES implementations within UNICORE and CREAM-BES, however they are using a different security setup relying on the OMII-UK security model that implies the support for the older Attribute Certificate-based VOMS server [25]. In addition, another difference to the approach of augmenting the GridSAM component with an OGSA-BES interface is that we integrate the standard interface in widely accepted Grid middleware systems (i.e. UNICORE, gLite) with a lot of other integrated functionality (e.g. delegation, file-transfers, workflows) and other higher level services.

Finally, OMII-Europe is currently working on an OGSA-BES interface for GT4 that can be seen as a Web service frontend to the *Grid Resource Allocation Manager (GRAM)* [27] of GT4. But its still a work in progress and currently provides no support for VOMS at all.

7. Conclusions

We described the interoperability between the UNICORE 6 Grid middleware and CREAM-BES system of gLite by using common open standards such as OGSA-BES and the JSDL specification. In addition and crucial to the process of integrating these interface implementation within real production Grid and e-Science infrastructures, we have shown that CREAM-BES as well as the OGSA-BES interface implementation of UNICORE 6 can be used with the same security profile based on X.509 certificates for authentication and SAML assertions (released by a newly developed VOMS server) for authorization.

The solution here described is not only applicable within DEISA and EGEE, because also other national and international Grids are using the major Grid middlewares gLite and UNICORE. To provide an example, the German national Grid D-Grid uses gLite, UNICORE and Globus for the access of computational resources. Hence, the interoperability of gLite and UNICORE with respect to job submission and management is also very useful for this production e-Science infrastructure. Furthermore, it is important to understand that the described interoperability between UNICORE 6 and CREAM-BES of gLite is not tailored to specific versions of them or tailored for these Grid middlewares. In principle, any HPC-P compliant Grid technology that is capable of providing support for the same common security profile is interoperable with UNICORE and gLite.

An important security area within job submission and management was not addressed in the paper. In fact, delegation has been kept out of this paper to keep it more simple and understandable since both systems recently provide proxy support (optional in UNICORE), delegation can be achieved by using this delegation mechanism although the delegation mechanism of UNICORE 6 is typically based on SAML assertions.

Future work will focus on work in the area of interoperability with other OGSA-BES implementations that might follow the same security profile such as the OGSA-BES implementation of the CROWN Grid middleware and the OGSA-BES implementation from OMII-Europe for GT4.

Even if we have shown that technical interoperability can be achieved today between UNICORE 6 and CREAM-BES of glite, deploying these OGSA-BES implementations is an important next step to broadly incorporate implementations of OGSA-BES and the HPC-P into production Grid and e-Science infrastructures. The implementation described here relies partly on the WS-based UNICORE 6 middleware. UNICORE 6 will be soon evaluated by the DEISA and D-Grid e-Science infrastructures for production usage. In addition, CREAM-BES will be soon evaluated by the EGEE infrastructure as one suitable component for deployment within the next generation gLite middleware.

Acknowledgments

The work presented in this paper has been supported by the OMII - Europe project under EC grant RIO31844-OMII-EUROPE, duration May 2006 - April 2008.

References

[1] A-WARE Project. http://www.a-ware.org/.
[2] Apache AXIS. http://ws.apache.org/axis/.
[3] Apache Tomcat. http://tomcat.apache.org/.
[4] Chemomentum Project. http://www.chemomentum.org/.
[5] DEISA. http://www.deisa.org/.
[6] EGEE. http://www.eu-egee.org/.
[7] German National Grid: D-Grid. http://www.d-grid.de.
[8] GridSAM. http://gridsam.sourceforge.net/2.0.1/index.html.
[9] GRIDSphere. http://www.gridsphere.org/.
[10] Jetty Web Server. http://www.mortbay.org.
[11] OASIS - WSRF Technical Committee. http://www.oasis-open.org/committees/wsrf.
[12] OGSA - Basic Execution Services (OGSA-BES). http://forge.ogf.org/sf/projects/ogsa-bes-wg.
[13] OMII - Europe. http://omii-europe.org/.
[14] Open Grid Forum. http://www.ogf.org/.
[15] Shibboleth. http://shibboleth.internet2.edu/.
[16] UNICORE. http://www.unicore.eu/.
[17] WISDOM Project. http://wisdom.eu-egee.fr/.
[18] WS- Addressing. http://www.w3.org/Submission/ws-addressing/.
[19] XFIRE. http://xfire.codehaus.org.
[20] P. Andreetto et al. CREAM: A simple, Grid-accessible, Job Management System for local Computational Resources. In *CHEP 2006, Mumbay, India*, 2006.
[21] A. Anjomshoaa et al. *Job Submission Description Language (JSDL) - Specification Version 1.0*. Open Grid Forum Proposed Recommendation, 2006.
[22] S. Cantor, J. Kemp, R. Philpott, and E. Maler. *Assertions and Protocols for the OASIS Security Assertion Markup Language*. OASIS Standard, 2005. http://docs.oasis-open.org/security/saml/v2.0/.
[23] D. Case, T. Cheatham, T. Darden, H. Gohlke, R. Luo, K. Merz, A. Onufriev, C. Simmerling, B. Wang, and R. Woods. The Amber biomolecular simulation programs. In *J. Computational Chemistry, 26: 1668-1688*, 2005.
[24] D. Colling et al. The EU DataGrid Workload Management System: towards the second major release. In *In Proceedings of CHEP 2003, La Jolla, California, USA*, 2003.
[25] Y. Demchenko et al. VO-based Dynamic Security Associations in Collaborative Grid Environments. In *Proc. of the Int. Symp. on Collaborative Technologies and Systems*, 2006.
[26] T. Dierks and C. Allen. *The TLS protocol version 1.0, Internet Engineering TaskForce, RFC 2246*. 1999. http:///www.ietf.org/rfc/rfc2246.txt.
[27] I. Foster. Globus Toolkit version 4: Software for Service-Oriented Science. In *Proceedings of IFIP International Conference on Network and Parallel Computing, LNCS 3779*, pages 213–223. Springer-Verlag, 2005.
[28] I. Foster, C. Kesselmann, J. M. Nick, and S. Tuecke. The Physiology of the Grid. In F. Berman, G. C. Fox, and A. J. G. Hey, editors, *Grid Computing - Making the Global Infrastructure a Reality*, pages 217–249. John Wiley & Sons Ltd, 2003.
[29] I. Foster, H. Kishimoto, A. Savva, D. Berry, A. Djaoui, A. Grimshaw, B. Horn, F. Maciel, F. Siebenlist, R. Subramaniam, J. Treadwell, and J. Reich. *Open Grid Services Architecture, Version 1.0*. Open Grid Forum Draft 30, 2005.
[30] M. Gudgin, M. Hadley, N. Mendelsohn, J. Moreau, and H. Nielsen. *SOAP Version 1.2 Part 1: Messaging Framework*. W3C Recommendation, 2003. http://www.w3.org/TR/soap12-part1/.
[31] M. Humphrey, C. Smith, M. Theimer, and G. Wasson. *HPC Profile Apllication Specification*. OGF draft, 2007.
[32] E. Laure et al. Programming The Grid with gLite. In *Computational Methods in Science and Technology*, pages 33–46. Scientific Publishers OWN, 2006.
[33] M. McIntosh et al. *WS-I Basic Security Profile Version 1.0*. WS-I Working Group Draft, 2006. http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0.html.
[34] R. Menday. The Web Services Architecture and the UNICORE Gateway. In *Proceedings of the International Conference on Internet and Web Applications and Services (ICIW) 2006, Guadeloupe, French Caribbean*, 2006.
[35] G. M. Morris, D. S. Goodsell, R. Halliday, R. Huey, W. E. Hart, R. K. Belew, and A. J. Olson. Automated Docking Using a Lamarckian Genetic Algorithm and and Empirical Binding Free Energy Function. In *J. Computational Chemistry, 19: 1639-1662.*, 1998.
[36] T. Moses et al. *eXtensible Access Control Markup Language*. OASIS Standard, 2005.
[37] A. Nadalin et al. *OASIS Web Security Username Token Profile 1.1*. 2006. OASIS Standard Specification.
[38] M. Rarey, B. Kramer, T. Lengauer, and G. Klebe. Predicting Receptor-Ligand interactions by an incremental construction algorithm. In *J. Mol. Biol. 261: 470-489*, 1996.
[39] R. Ratering et al. GridBeans: Supporting e-Science and Grid Applications. In *2nd IEEE International Conference on e-Science and Grid Computing (E-Science 2006), Amsterdam, The Netherlands*, 2006.
[40] M. Riedel and D. Mallmann. Standardization Processes of the UNICORE Grid System. In *Proceedings of 1st Austrian Grid Symposium 2005, Schloss Hagenberg, Austria*, pages 191–203. Austrian Computer Society, 2005.
[41] A. Savva. *JSDL Single Program Multiple Data Application Extensions*. OGF draft, 2007. http://www.ogf.org/Public_Comment_Docs/Documents/May-2007/draft-ogf-jsdl-spmd-008.pdf.
[42] B. Schuller et al. A Versatile Execution Management System for Next Generation UNICORE Grids. In *Proc. of the 2nd UNICORE Summit*, 2006.
[43] A. Streit et al. UNICORE - From Project Results to Production Grids. In L. Grandinetti, editor, *Grid Computing: The New Frontiers of High Performance Processing, Advances in Parallel Computing 14*, pages 357–376. Elsevier.
[44] S. Tuecke et al. *Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile, RFC 3820*. 2004. http:///www.ietf.org/rfc/rfc3820.txt.