EU project: RI031844-OMII-Europe

Project no: **RI031844-OMII-Europe**

Project acronym: **OMII-Europe**

Project title: **Open Middleware Infrastructure Institute for Europe**

Instrument: **Integrated Infrastructure Initiative**

Thematic Priority: **Communication network development**

M:JRA1.10 OMII-Europe supports BES plus required extensions

Due date of deliverable: Oct 2007
Actual submission date: Apr, 2008

Start date of project: **1 May 2006** Duration: **2 years**

Organisation name of lead contractor for this deliverable: INFN

Revision [1.0]

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)		
Dissemination Level		
PU	Public	**Public**
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

EU project: RI031844-OMII-Europe

Document Control Sheet

Document	**Title**: OMII-Europe supports BES plus required extensions	
	ID: M:JRA1.10	
	Version: 1.0	**Status**: Final
	Available at:	
	Software Tool: OpenOffice.org	
	File(s): OMII-EU-MJRA1.10-v1.0.odt	
Authorship	**Written by:**	Moreno Marzolla (Editor)
	Contributors:	TBD
	Reviewed by:	TBD
	Approved by:	TBD

Document Status Sheet

Version	Date	Status	Comments
0.1	2008-01-29	Draft	Initial CREAM-BES part (incomplete). Many missing sections
0.2	2008-03-10	Draft	All sections complete, except section 4
0.3	2008-03-27	Draft	More additions
0.4	2008-04-04	Draft	Completed CREAM-BES part
1.0	2008-04-29	Final	Final version

Table of Contents

1 Executive Summary

This document describes the implementation of the OGSA Basic Execution Service (OGSA-BES) specification version 1.0 in the CREAM-BES and UNICORE6 job execution and management services. It describes the architectural modifications which has been implemented into the gLite 3.1 CREAM and UNICORE6 systems in order to implement the BES interface.

This document is structured as follows. In section 2 we give a brief introduction to the Basic Execution Service specification, version 1.0; note that a more in-deep description of BES has already been given in [MJRA1.17]. In section 3 we describe the implementation of BES into the CREAM service for the gLite middleware version 3.1, and in section 4 we describe the implementation of BES in UNICORE6. Finally, section 5 contains the conclusions and lessons learned during the implementation effort.

2 The Basic Execution Service (BES) Specification

The OGSA Basic Execution Service specification defines a Web Service interface for submission and management of computational jobs (called *activities*) on execution services. The BES specification defines two Web Service port-types as follows:

- the **BES Management** port-type is used to control the BES service itself; it contains two operations for starting and stopping accepting new submissions. This port-type is normally used by the BES service administrators.
- the **BES Factory** port-type is used by ordinary clients to request creation of new activities, manage and monitors the activities, and ask for the capabilities of the BES service.

The following table contains the list of all defined BES operations, divided according to the port-types they belong to:

BES-Management Port-type	
`StopAcceptingNewActivities`	Request that the BES stop accepting new activities
`StartAcceptingNewActivities`	Request that the BES start accepting new activities
BES-Factory Port-type	
`CreateActivity`	Request the creation of a new activity
`GetActivityStatuses`	Request the status of a set of activities
`TerminateActivities`	Request that a set of activities be terminated
`GetActivityDocuments`	Request the JSDL documents for a set of activities
`GetFactoryAttributesDocument`	Request XML document containing BES properties

BES uses the Job Submission Description Language (JSDL) as the notation used to describe activities. The JSDL specification [JSDL] can be used to describe computational jobs and their requirements. The CreateActivity operation takes a JSDL document as parameter, and instantiates a new activity within the BES service.

One of the goals of the OMII-EU project was to support the BES (and JSDL) specification within the project partner's GRID middlewares. In particular, we implemented the following versions for the above mentioned specifications (which, at the time of writing, are the latest versions available):

- OGSA—BES, Version 1.0
- JSDL, Version 1.0

These interfaces have been implemented into the CREAM-BES Computing Element, which is derived from the legacy CREAM Computing Element of the the gLite 3.1 middleware, and into the UNICORE6 system.

This document describes the implementation effort undertaken in OMII-EU for supporting the OGSA-BES specification. A detailed description of JSDL and BES is outside the scope of this document, as it has already been done in [MJRA1.7] and [MJRA1.17] respectively.

3 Implementation of BES in the CREAM-BES service

3.1 The CREAM-BES Service

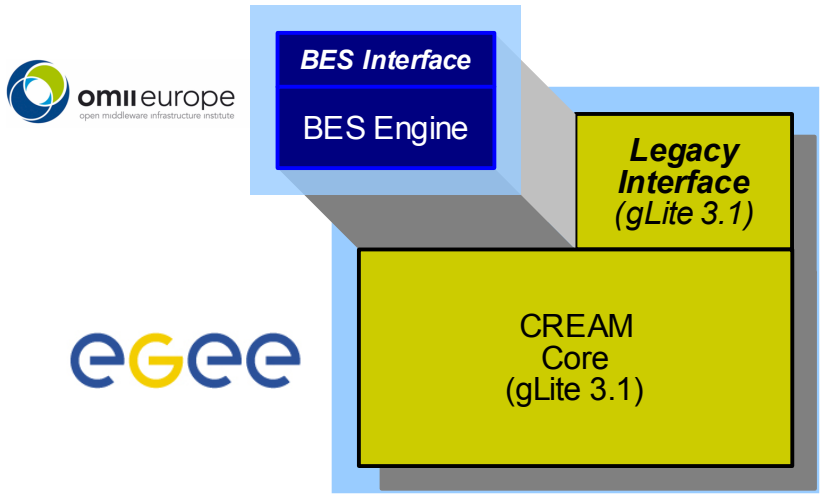
CREAM-BES is an enhancement of the legacy CREAM Computing Element. CREAM is being developed by the EGEE-2 project for the gLite 3.1 middleware [CREAM]. CREAM is a Web Service-based Computing Element (CE) written in Java; technically, it runs as an Axis container [AXIS] in the Tomcat application server [TOMCAT]. CREAM-BES extends the gLite 3.1 CREAM service with additional port-types implementing the BES specification. Thus, using CREAM-BES it is possible to submit and manage activities described as JSDL documents using a BES compliant interface.

As the development of CREAM began long before the availability of the BES/JSDL specifications, CREAM legacy interface is not based on OGSA-BES; moreover, the job description notation used in CREAM is based on Condor ClassAds instead of the JSDL. Currently, the gLite 3.1 middleware relies on CREAM legacy interface: the gLite Workload Management System (WMS) has a specialized component called ICE (Interface to Cream Environment) which is responsible for handling the interaction to the CREAM-based Computing Elements. ICE uses the legacy CREAM interface, because it was developed before BES was available. Moreover, ICE relies on some CREAM-specific features which are yet not available in the BES/JSDL specifications:

- CREAM supports MPI parallel jobs which are described using the ClassAd based JDL notation; MPI jobs are not supported by the JSDL specification;
- The gLite 3.1 security infrastructure is based on X509 proxy certificates. CREAM provides a *delegation port* which is used by ICE to delegate the user's credentials to the Computing Element. The BES specification does not provide (nor require) any specific security infrastructure.
- ICE and CREAM use a lease-based protocol to ensure that all jobs are eventually removed from the Computing Element in case of network partitioning (i.e., in case ICE and CREAM loose contact). While BES implementations MAY support the WS-ResourcelifeTime operation [WS-ResourceLifetime] which basically provide the same functionality, CREAM-BES currently does not support that operation.
- ICE receives asynchronous notifications from the CEMon service, which is a separate service hosted on the same physical node as CREAM. CEMon notifies subscribers when job status change happens, so that ICE does not need to explicitly poll CREAM to get the current job statuses. While BES implementations MAY support the WS-Eventing or WS-Notification protocols [WS-Notification], CREAM-BES currently does not support these protocols.

For these reasons, we extended the CREAM service with an additional BES interface, while still maintaining the legacy one. The resulting (Extended) service, which we called CREAM-BES is made of two separate components (see Drawing 1):

1. the legacy gLite 3.1 CREAM server;
2. the implementation of the BES interface for CREAM, as done by the OMII-EU project.

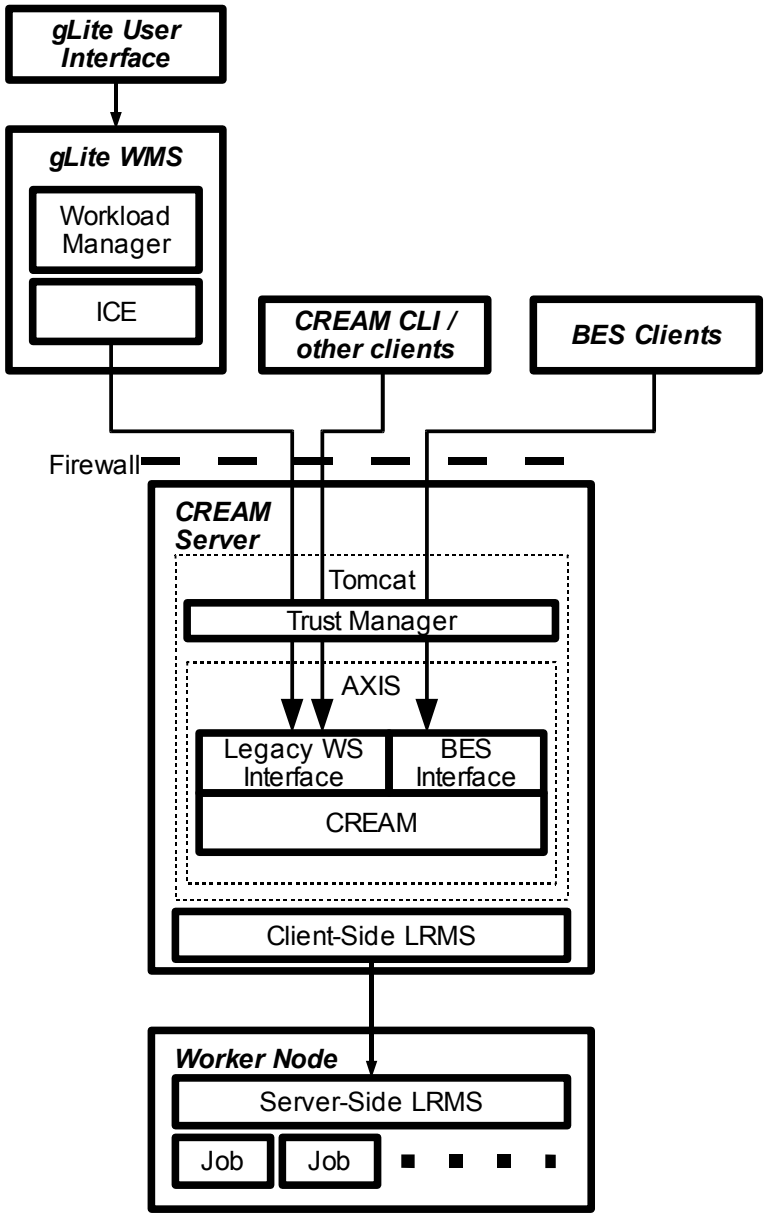


Drawing 1: High-Level view of CREAM-BES

Both CREAM and its BES interface are built using the ETICS build and test facility [ETICS]. ETICS allows developers to specify the list of dependencies needed to build a specific component. These dependencies could be external libraries (e.g., log4j, AXIS, etc.) or other components managed by ETICS. The BES interface for CREAM, among other things, specifies a dependency on the legacy CREAM package, as CREAM-BES uses the CREAM core "as is", without any modification. In this way, every improvement or bug fix which is done in the legacy CREAM is automatically used by CREAM-BES as well. This guarantees that CREAM-BES and the legacy CREAM do not diverge as development proceeds.

Drawing 2 shows in more details the structure of the CREAM-BES component, and how it interacts with external clients. There are currently two classes of clients which can submit and monitor jobs using CREAM-BES. The first class are the gLite 3.1 clients, which includes the gLite Workload Management System (WMS) and the CREAM Command Line Tools (CLI). The WMS is used to manage and dispatch jobs to gLite Computing Elements; the ICE module within the WMS takes care of all interactions with CREAM service. gLite users submit jobs through the gLite User Interface (UI), which then forwards the requests to the WMS where, among other things, resource matchmaking takes place. After the WMS selects a CREAM CE as a suitable candidate execution host for the job, the WMS sends the request to ICE, which handles all interactions with CREAM.

The legacy CREAM interface is used by the CLI utilities, which can be used for direct interaction with the CREAM service. Using the CLI, users can submit and manage jobs directly on the CREAM node, bypassing the gLite WMS. In this way it is possible to install CREAM in stand-alone mode, that is, outside of a gLite middleware installation.



Drawing 2: CREAM overall structure

The second class of clients for CREAM includes all BES-compliant clients; with the CREAM-BES installation, a simple java-based command line tool is provided. Using this tool, a user can submit and monitor jobs using the methods of the BES interface. Of course, any BES client can be used here, so the choice is not limited to the simple one provided with the CREAM-BES installation.

Before being processed by CREAM-BES, all requests are checked by the Trust Manager module, which is responsible for accepting/rejecting requests based on administrator-defined access control rules. The Trustmanager extracts the user Distinguished Name (DN) from the certificate which is used for the TSL/SSL handshate; the Virtual Organization (VO) extensions are obtained from the certificate itself, if a VOMS certificate is used, or from the SAML assertions in the SOAP header, if

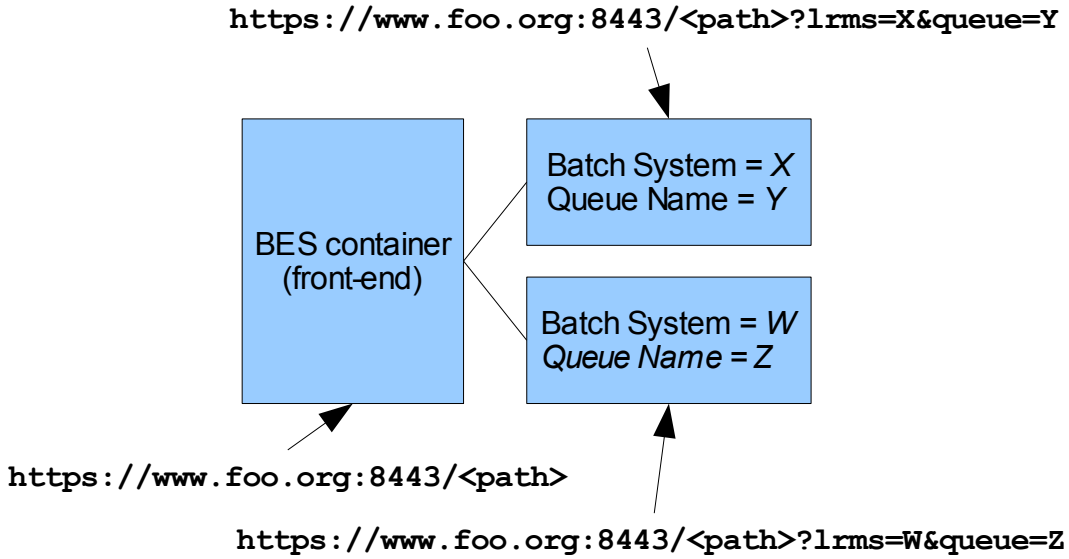
VOMS-SAML is used instead. See the following section for more details on the security aspects of CREAM/CREAM-BES.

After the request has been authenticated, and the issuer of the request is authorized to execute the associated action, the request is processed by the CREAM engine. Usually, each request corresponds to one or more commands which are dispatched to a Local Resource Management System (LRMS). Currently, CREAM (and hence CREAM-BES) supports the PBS and LSF batch systems. Interaction with the batch system is not done directly, but through the BLAH component (not shown in the picture) [BLAH]. BLAH provides a common abstraction layer to different batch systems; for this reason, every batch system which is (or will be) supported by BLAH can automatically, and without modifications be used by CREAM-BES.

3.2 Resource representation

The gLite 3.1 CREAM service can be used as a front-end to multiple LRMS (batch queues). The user is able to select where a job will be executed by specifying the batch system type and queue name in the submitted JDL. JSDL does not offer the possibility to select the batch queue where the activity must be executed. In order to be compliant with the JSDL specification, we decided not to define a JSDL extension for that, but use the standard BES resource model.

A CREAM-BES service is logically represented as in the following picture:



We have a "front-end" BES container, which is identified by an appropriate URI (https://www.foo.org:8443/<path> in this case). Submitting an activity to the front-end BES container will result in the activity being dispatched to one of the available batch queues. The selection policy can be configured as a plugin to CREAM-BES (the current default plugin selects a batch queue randomly). The GetFactoryAttributesDocument BES operation returns a "hierarchical" representation of the available resources, that is, it contains a Basic Resource description, with additional Basic Resources appearing as children of the ContainedResource XML element. Each contained resource is represented by an URI which is exactly the same as the front-end, with additional query elements specifying the batch system and queue name of a single batch queue. It should be observed that, from

the logical point of view, each contained resource is a BES resource itself. Thus, it is possible to invoke all BES operations on the URIs of the contained resources.

3.3 Security considerations

While security considerations are outside the scope of the BES specification, it is nevertheless necessary to ensure that users accessing any BES service are identified using appropriate credentials (Authentication), and access to the service respect (Authorization).

Given that CREAM-BES is an additional interface of the legacy CREAM service, it shares the core functionalities (including authentication/authorization) with CREAM. For this reason we shall describe now how security is managed in CREAM.

3.3.1 Authentication/Authorization

The aim of the authorization process is to check whether an authenticated user has the rights to access services and resources and to perform certain tasks. The decision is taken on the basis of policies that can be either local or decided at the VO level. Administrators need a tool that allows them to easily configure the authorization system in order to combine and integrate both these policies. For this reason, CREAM adopts a framework that provides a light-weight, configurable, and easily deployable policy-engine-chaining infrastructure for enforcing, retrieving, evaluating and combining policies locally at the individual resource sites.

The framework provides a way to invoke a chain of policy engines and get a decision result about the authorization of a user. The policy engines are divided in two types, depending on their functionality. They can be plugged into the framework in order to form a chain of policy engines as selected by the administrator in order to let him set up a complete authorization system. A policy engine may be either a Policy Information Point (PIP) or a Policy Decision Point (PDP). PIP collect and verify assertions and capabilities associated with the user, checking his role, group and VO attributes. PDP may use the information retrieved by a PIP to decide whether the user is allowed to perform the requested action, whether further evaluation is needed, or whether the evaluation should be interrupted and the user denied access.

In CREAM both VO and “ban/allow” based authorizations are supported. In the former scenario, implemented via the VOMS PDP, the administrator can specify authorization policies based on the VO the jobs' owners belong to (or on particular VO attributes). In the latter case the administrator of the CREAM-based CE can explicitly list all the Grid users (identified by their X.509 Distinguished Names) authorized to access CREAM services.

For what concerns authorization on job operations, by default each user can manage (e.g. cancel, suspend, etc.) only her own jobs. However, the CREAM administrator can define specific “super-users” who are empowered to manage also jobs submitted by other users.

The execution of user jobs in a Grid environment requires isolation mechanisms for both applications (to protect these applications from each other) and resource owners (to control the behaviour of these arbitrary applications). In the absence of solutions based on the virtualization of resources (VM), CREAM implements isolation via local credential mapping, exploiting traditional Unix-level security mechanisms like a separate user account per Grid user or per job. This Unix domain isolation is implemented in the form of the glexec system, a sudo-style program which allows the execution of

the user's job with local credentials derived from the user's identity and any accompanying authorization assertions. This relation between the Grid credentials and the local Unix accounts and groups is determined by the LCMAPS. glexec also uses the LCAS to verify the user proxy, to check if the user has the proper authorization to use the glexec service, and to check if the target executable has been properly "enabled" by the resource owner.

3.3.2 Security in CREAM-BES

CREAM-BES supports user authentication using X509 proxy certificates, either with VOMS extensions or using VOMS-SAML headers in the SOAP request. As stated above, VOMS extensions are fundamental because access control policies and user mapping is performed also at the VO level. The CREAM-BES interface can be accessed only using the TLS (HTTPS) encrypted connections.

VOMS extensions can be transferred to the CREAM-BES service in two different ways:

1. Using an X509 proxy certificate with VOMS extensions (VOMS proxy). In this case, VOMS extensions are part of the user credentials, which are transferred using the TSL connection to the CREAM-BES service. The gLite trustmanager verifies the validity of the X509 certificate, and unpacks any VOMS extension contained therein;
2. Using an X509 proxy certificate *without* VOMS extensions; in this case VO information must be inserted into the SOAP header as SAML assertions.

In both cases, the CREAM-BES service (to be precise, the Trustmanager layer, see Drawing 1) authenticates the user according to the DN contained in the certificate used during the TLS handshake; this certificate could be either a VOMS proxy or an X509 certificate without VOMS extensions.

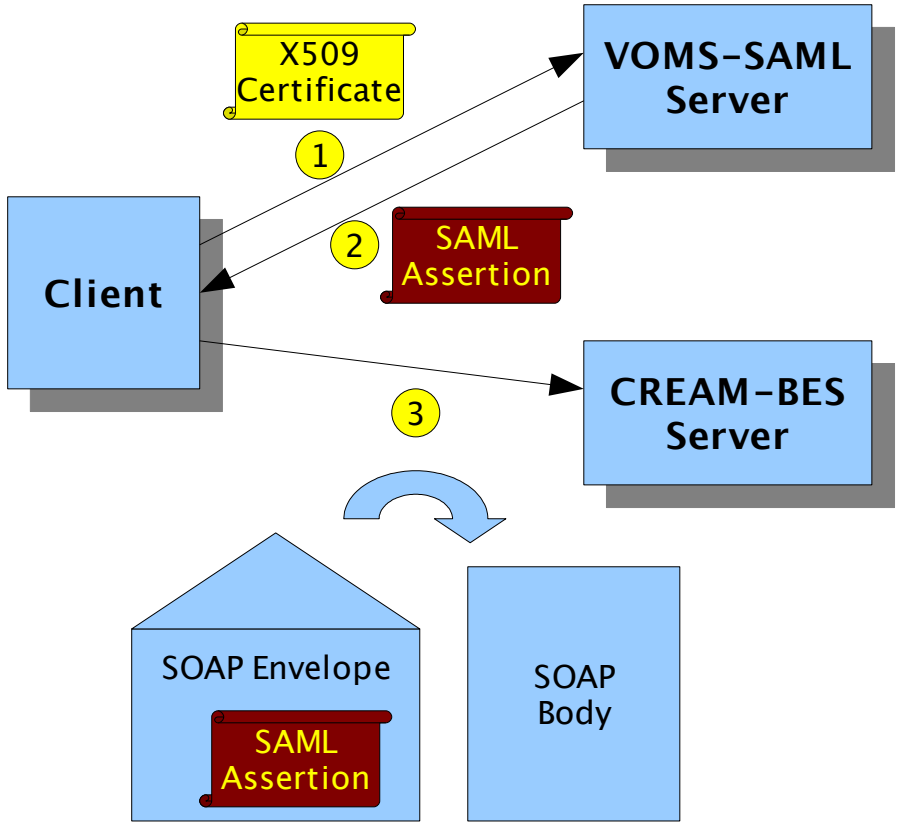
Once the user is authenticated, the CREAM-BES service needs to associate VO information to that user in order to authorize the user of resources (e.g., batch queues contained into the BES service). VO information are extracted in different ways depending whether access method 1. or 2. above was used.

If the first method (authentication using VOMS proxies) is used, then VO information are transferred to the CREAM-BES service together with the user's certificate. Thus, VO information are extracted from the VOMS proxy which is provided during the TLS handwhake.

If the second method (authentication using X509 certificates plus SAML assertions), then the CREAM-BES service extracts VO information from the SAML assertions which must be contained into the SOAP header used to invoke BES operations. More specifically, in order to access the the CREAM-BES service, the user needs to perform the following steps:

1. The client requests the SAML information to the VOMS-SAML service. The user needs to access VOMS-SAML with appropriate credentials (X509 certificate) in order to get the SAML assertions.
2. The VOMS-SAML service identifies the client from the supplied credentials. It then supplies a SAML assertion (signed with the server certificate in order to guarantee its authenticity).
3. The client inserts the SAML assertions into the SOAP header which is then used to access the BES interface of the CREAM service. CREAM-BES extracts the SAML assertions (which, among other things, contains the Virtual Organization the user belongs to).

The situation is summarized in Drawing 3.



Drawing 3: BES authentication with SAML

3.4 CREAM-BES usage example

A simple command-line client for CREAM-BES is built automatically when the user builds CREAM-BES using the ETICS build system. A pre-compiled, "plug-and-play" client is available from the CREAM-BES wiki site (http://grid.pd.infn.it/omii/cream-bes#using). We refer to the wiki for up-to-date usage examples, and illustrate here some basic usage examples.

The CREAM-BES command-line client allows the user to invoke all the operations on the CREAM-BES endpoint: users can create activities, query the activity status, query the BES endpoint capabilities and terminate activities. The client archive contains, in the top-level directory, a cream-bes.sh shell script which actually invokes the operation. The archive also contains a test/ subdirectory with a simple JSDL (./test/test.jsdl) document test.jsdl which can be used to create an activity on a CREAM-BES service.

In the following we consider the CREAM-BES endpoint which is available at the INFN-CNAF OMII-EU Evaluation Infrastructure. The URI of the endpoint is https://omiivm03.cnaf.infn.it:8443/ce-cream/services/CreamBes. This service is configured to use VOMS-SAML authentication only.

3.4.1 Querying endpoint attributes

OGSA—BES provides the GetFactoryAttributesDocument operation which can be used to get the capabilities of the BES endpoint. Suppose that we want to invoke this operation on the omiivm03 BES endpoint. In order to do that, you need to issue the following command:

```
./cream-bes.sh attributes \
  -r https://omiivm03.cnaf.infn.it:8443/ce-cream/services/CreamBes \
  --voms \
  https://omii002.cnaf.infn.it:8443/voms/saml/test/services/AttributeAuthorityPortType
```

where: https://omiivm03.cnaf.infn.it:8443/ce-cream/services/CreamBes is the URI of the BES endpoint and https://omii002.cnaf.infn.it:8443/voms/saml/test/services/AttributeAuthorityPortType is the URI of the VOMS-SAML service endpoint. The output is a textual rendering of the XML fragment contained in the SOAP reply message, which describes the capabilities of the selected BES endpoint:

```
CommonName = CREAM-BES test
Long Description = CREAM-BES CE for tests
Is Accepting New Activities = true
Local Resource Manager Type = urn:lrms.type.undefined
Total number of contained resources = 2
Total Number Of Activities = 4
        activity epr = https://omiivm03.cnaf.infn.it:8443/ce-cream/services/CreamBes
        jobId = https://omiivm03.cnaf.infn.it:8443/CREAM882464212
---------------------------- raw epr begin ----------------------------
[...]
----------------------------- raw epr end -----------------------------
[...]
Root basic resource attributes:
        No basic attributes found
Contained basic resource attributes:
        Resource Name = https://omiivm03.cnaf.infn.it:8443/ce-
cream/services/CreamBes?lrms=pbs&queue=cert
        CPUArchitecture = other
        Operating System Name = other
        Operating System Version = SLC
        CPUCount = 4.0
        CPUSpeed (Hz) = 2000.0
        Physical Memory = 1024.0
        Virtual Memory = 2048.0
Contained basic resource attributes:
        Resource Name = https://omiivm03.cnaf.infn.it:8443/ce-
cream/services/CreamBes?lrms=pbs&queue=omii
        CPUCount = 4.0
        CPUSpeed (Hz) = null
        Physical Memory = null
        Virtual Memory = null
```

Observe that the selected BES endpoint contains two resources (each one being a batch queue). The resources can be accessed using the URI in the “Resource Name” line (i.e., https://omiivm03.cnaf.infn.it:8443/ce-cream/services/CreamBes?lrms=pbs&queue=cert and https://omiivm03.cnaf.infn.it:8443/ce-cream/services/CreamBes?lrms=pbs&queue=omii). Both these URI point to a BES service (which is the exact same instance of the BES service at the URI https://omiivm03.cnaf.infn.it:8443/ce-cream/services/CreamBes). It is possible to check the capabilities of each single batch queue, as follows:

```
./cream-bes.sh attributes -r\
  https://omiivm03.cnaf.infn.it:8443/ce-\cream/services/CreamBes?lrms=pbs&queue=cert \
  --voms \
  https://omii002.cnaf.infn.it:8443/voms/saml/test/services/AttributeAuthorityPortType
```

the output being:

```
CommonName = CREAM-BES test
Long Description = CREAM-BES CE for tests
Is Accepting New Activities = true
Local Resource Manager Type = urn:lrms.type.undefined
Total number of contained resources = 0
Total Number Of Activities = 4
        activity epr = https://omiivm03.cnaf.infn.it:8443/ce-cream/services/CreamBes
        jobId = https://omiivm03.cnaf.infn.it:8443/CREAM882464212
---------------------------- raw epr begin ----------------------------
[...]
----------------------------- raw epr end -----------------------------
Root basic resource attributes:
        Resource Name = https://omiivm03.cnaf.infn.it:8443/ce-
cream/services/CreamBes?lrms=pbs&queue=cert
        CPUArchitecture = other
        Operating System Name = other
        Operating System Version = SLC
        CPUCount = 4.0
        CPUSpeed (Hz) = 2000.0
        Physical Memory = 1024.0
        Virtual Memory = 2048.0
```

3.4.2 Submitting a new activity

The ./test/test.jsdl activity document, as contained in the CREAM-BES client archive, is the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<jsdl:JobDefinition xmlns="http://www.example.org/"
                    xmlns:jsdl="http://schema.ggf.org/jsdl/2005/11/jsdl"
                    xmlns:jsdl-posix="http://schemas.ggf.org/jsdl/2005/11/jsdl-posix"
                    xmlns:jsdl-hpcpa="http://schemas.ggf.org/jsdl/2006/07/jsdl-hpcpa">
  <jsdl:JobDescription>

    <!-- JobIdentification block -->
    <jsdl:JobIdentification>
      <jsdl:JobName>Simple Test Job</jsdl:JobName>
      <jsdl:Description>A simple job which basically does nothing</jsdl:Description>
    </jsdl:JobIdentification>

    <!-- Application block -->
    <jsdl:Application>
      <jsdl:ApplicationName>sleep</jsdl:ApplicationName>
      <jsdl-hpcpa:HPCProfileApplication>
        <jsdl-hpcpa:Executable>/bin/sleep</jsdl-hpcpa:Executable>
        <jsdl-hpcpa:Argument>20</jsdl-hpcpa:Argument>
      </jsdl-hpcpa:HPCProfileApplication>
    </jsdl:Application>

    <!-- Resources block -->
    <jsdl:Resources>
      <jsdl:TotalCPUCount>
        <jsdl:Exact>2.0</jsdl:Exact>
      </jsdl:TotalCPUCount>
    </jsdl:Resources>

  </jsdl:JobDescription>
</jsdl:JobDefinition>
```

This job executes a "/bin/sleep 20" command on the execution host. The resource requirements block specifies that this activity requires an execution host with exactly two CPUs.

In order to submit this activity it is necessary to invoke the client with the following command:

```
./cream-bes.sh create \
  -r https://omiivm03.cnaf.infn.it:8443/ce-cream/services/CreamBes \
  --voms \
  https://omii002.cnaf.infn.it:8443/voms/saml/test/services/AttributeAuthorityPortType \
  test/test.jsdl
```

If the submission is succesful, the output will be the activity identifier:

```
---------------------------- raw epr begin ----------------------------
[...]
----------------------------- raw epr end -----------------------------

Activity Identifier address: https://omiivm03.cnaf.infn.it:8443/ce-cream/services/CreamBes
Activity Identifier Reference Parameters: https://omiivm03.cnaf.infn.it:8443/CREAM183524481
```

3.4.3 Querying the activity status

To query the status of an activity, it is necessary to invoke the GetActivityStatuses BES operation, as follows:

```
cream-bes.sh status \
  -r https://cream-ce-01.pd.infn.it:8443/ce-cream/services/CreamBes \
--voms \
  https://omii002.cnaf.infn.it:8443/voms/saml/test/services/AttributeAuthorityPortType \
  https://omiivm03.cnaf.infn.it:8443/CREAM183524481
```

The output is the activity status:

```
Retrieved 1 activities
activity epr = https://cream-ce-01.pd.infn.it:8443/ce-cream/services/CreamBes
          jobId = https://omiivm03.cnaf.infn.it:8443/CREAM183524481
          activity status = Finished
```

4 Implementation of BES in UNICORE6

As far as UNICORE is concerned, a study of the the OGSA-BES interface reveals that it is similar to the Target System Service (TSS) and Job Management Service (JMS) of the rather proprietary UNICORE Atomic Services (UAS) of UNICORE 6.

While the UAS does have WS-RF compliant message exchanges and submit operation takes a JSDL document as parameter, the syntax of the UAS operations is proprietary. The OGSA-BES interfaces offer UNICORE 6 a standardized syntax for Web service operations dealing with job control and management which can be internally implemented on top of the same execution backend XNJS as the UAS.

The Web service layer is well isolated from the lower-level execution engine. All OGSA-BES invocations are transmitted to the server-side within the SOAP body, except the pieces of information within the SOAP header that are used to address specific job WS-Resource instances and also the security tokens (i.e. SAML assertions, see JRA1 VOM Activity). As shown in Illustration 4, the integrated OGSA-BES UNICORE 6 interface has three services: Activity, Factory and Management services.

The BES Factory Service is responsible for the creation and control of a set of activities (described using JSDL). In the UNICORE 6 implementation, this Web service interface uses the newly developed and enhanced (X)NJS as execution backend that provide significant performance and scalability improvements over UNICORE 5. The CreateActivity(JSDL) operation of the BES-Factory Service leads to the creation of a job resource within the NJS that represents the computational job described by the JSDL.

At the Web service level, this resource can be controlled and monitored by using the BES Activity Service operations, e.g. by using the `Terminate()` or `GetStatus()` operations. In addition, the OGSA-BES implementation for UNICORE includes a BES Management Service. This service improves the functionality of UNICORE in terms of remote administration. An administration client, for instance, can be used to access this service and control whether new jobs can be submitted to UNICORE or not. To support this, the BES operations `StopAcceptingNewActivities()` and `StartAcceptingNewActivities()` are supported by UNICORE 6.

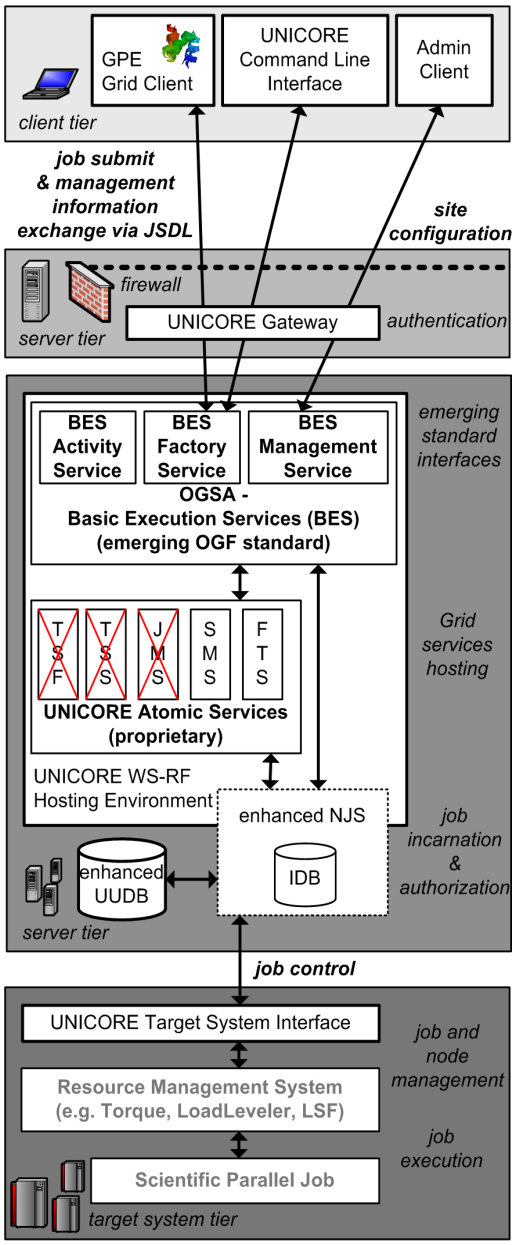


Illustration 4: UNICORE 6 with a deployed OGSA–BES services. In OMII–Europe the proprietary UNICORE Atomic Services are partly replaced by OGSA–BES services.

UNICORE 6 is also compliant with the HPC-Basic Profile. This profile specifies the usage of the OGSA-BES interface in conjunction with certain extensions to JSDL. UNICORE 6 implements the HPC Application Extension Specification [HPCP] as an extension to JSDL that is used to describe an executable running as an operating system process. Basically it shares much in common with the JSDL POSIXApplication, but removes some of the features that present barriers to interoperability by using the XML element jsdl-hpcpa:Executable and other similar elements.

The OGSA-BES specification has left this concern to implementers. But both file transfer and storage management are needed if complete JSDL descriptions are to be supported. Therefore, the OGSA-BES implementation within UNICORE 6 also provides a file transfer service (FTS). The FTS provides Streamable ByteIO (SByteIO) and Random ByteIO (RByteIO) data access mechanisms to support job submissions that rely on staged data. These specifications are standardized by the OGF and thus lay the foundation for data staging capabilities in interoperability scenarios and cross-Grid use cases. In CREAM-BES, a simple file staging mechanism has been implemented based on plain FTP; Other, more secure and scalable alternatives will be implemented in the future.

As already shown in Illustration 4, the integrated OGSA-BES interface of UNICORE 6 consists of three standardized services. The BES-Factory Service is responsible for the creation and control of activities described with JSDL. The invocation of its CreateActivity() 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 required by the HPC-BP and thus UNICORE is HPC-BP 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, 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 OGSABES, UNICORE 6 also provides operations according to the WS-Resource properties for standardized property requests of jobs submitted through the OGSA-BES interface.

Both UNICORE 6 and CREAM-BES support user authentication using X509 proxy certificates with VOMS-SAML headers in the SOAP request, while UNICORE 6 can also run full X509 certificates in conjunction with VOMS-SAML headers in the SOAP messages. As stated above, VOMS extensions are fundamental because access control policies and user mapping is performed also at the VO level. The BES interfaces for UNICORE 6 and CREAM-BES can be accessed using the TLS (HTTPS) encrypted connections.

5 Conclusions

The goal of the JRA1/Job Submission and Monitoring task was to re-engineer existing job submission and management components according to the emerging standards. During the second year of the OMII-EU project, we implemented the OGSA Basic Execution Service (BES) version 1.0 and Job Submission Description Language (JSDL) version 1.0 into two existing job management systems for two different Grid platforms, namely UNICORE 6 and the CREAM-BES Computing Element from gLite 3.1.

UNICORE 6 represents the most recent version of the UNICORE system, meaning that now UNICORE provides BES and JSDL support out of the box. Hence, the support of both is part of the core package within UNICORE 6 that can be obtained from the OMII-Europe repository and on UNICORE@SourceForge. Early versions are already at some DEISA sites evaluated. Also some sites within D-Grid as well as other European projects have already started to work with UNICORE 6 in general and the OGSA-BES implementation in particular.

CREAM-BES represents an extension of the CREAM Computing Element from gLite 3.1; CREAM-BES uses the legacy CREAM core, and adds an additional BES interface to the legacy one (which is still accessible). Future plans include the adoption of updated versions of the BES/JSDL specifications, in order to keep the software aligned with the most recent interface specification available, and the adoption of appropriate standard extensions whenever they will be considered useful for the middlewares users. CREAM-BES will also be merged into the main CREAM code branch. This means that, similarly to what has been done with UNICORE 6, future versions of CREAM will support BES/JSDL out of the box, without the need to install additional components. This will ensure that the code developed by OMII-EU will be used and maintained by the CREAM developers in the gLite 3.1 middleware.

Finally, the reached standardization of the OGSA-BES interface and their integration into the major Grid middleware systems gLite and UNICORE are a major success for Grids, especially within Europe. That means that the fundamental job submission technologies in conjunction with an advanced security setup (from JRA3) can be used in real production use cases on EGEE and DEISA in the near future.

In addition, the standardization allows end-users to use these two rather different infrastructures seamlessly. The use cases of JRA3 – Task 2 already have shown that the interfaces can be used in real scientific scenarios (e.g. WISDOM, EUFORIA, EU-IndiaGrid).

6 References

[M:JRA1.7] *Definition of the required extensions needed to JSDL to satisfy OMII-Europe requirements*, OMII-EU Milestone Document M:JRA1.7, available on the project site: http://www.omii-europe.org/

[M:JRA1.17] *BES will be evaluated with respect to its adoption in the middleware of the OMII-Europe partners*, OMII-EU Milestone Document M:JRA1.17, available on the project site: http://www.omii-europe.org/

[M:JRA1.8] *GridSAM integration into UNICORE: architecture and implementation available*, OMII-EU Milestone Document M:JRA1.8, available on the project site: http://www.omii-europe.org/

[D:JRA1.9] *The first yearly report on the Job Submission and Job Monitoring task*, OMII-EU Deliverable D:JRA1.9, available on the project site: http://www.omii-europe.org/

[MJRA1.9] *Implementation of JSDL into OMII-Europe middleware together with the identified extensions*, OMII-EU Milestone MJRA1.9

[MJRA1.10] *OMII-Europe supports BES plus the required extensions*, OMII-EU Milestone MJRA1.10

[SHAH07] Z. Ali Shah, P. Andreetto, S. van de Berghe, A. Ferraro, F. Hedman, V. Li, M. Marzolla, Shabaz Memon, Shiraz Memon, M. Riedel, D. Snelling, K. Stamou, A. Streit, B. Twedell, V. Venturi, *Open Standards-based Interoperability of Job Submission and Management Interfaces across the Grid Middleware Platforms gLite and UNICORE*, Proceedings of International Interoperability and Interoperation Workshop (IGIIW) 2007 at 3rd IEEE International Conference on e-Science and Grid Computing, Bangalore, India, December, 2007, IEEE Computer Society, ISBN 0-7695-3064-8, pp. 592—599

[MEMON07] M.S. Memon, A.S. Memon, M. Riedel, B. Schuller, D. Mallmann, B. Tweddell, A. Streit, S. van den Berghe, D. Snelling, V. Li, M. Marzolla, P. Andreetto, *Enhanced Resource Management Capabilities using Standardized Job Management and Data Access Interfaces within UNICORE Grids*, Proceedings of 3rd Workshop on Scheduling and Resource Management for Parallel and Distributed Systems SRMPDS 2007, ICPADS'07 - The 13th International Conference on Parallel and Distributed Systems Hsinchu, Taiwan, December, 2007, Volume 2, 5—7 dec 2007, pp. 1—6, ISBN 978-1-4244-1890-9

[RIEDEL07] M. Riedel, B. Schuller, D. Mallmann, R. Menday, A. Streit, B. Tweddell, M.S. Memon, A.S. Memon, B. Demuth, Th. Lippert, D. Snelling, S. van den Berghe, V. Li, M. Drescher, A. Geiger, G. Ohme, K. Benedyczak, P. Bala, R. Ratering, A. Lukichev *Web Services Interfaces and Open Standards Integration into the European UNICORE 6 Grid Middleware* Proceedings of 2007 Middleware for Web Services (MWS 2007) Workshop at 11th International IEEE EDOC Conference "The Enterprise Computing Conference", 2007, Annapolis, Maryland, USA, to appear

[AXIS2] Apache Axis2 project, http://ws.apache.org/axis2/

[XFIRE] Codehaus XFire, http://xfire.codehaus.org/

[JRA1WIKI] OMII-EU JRA1/Job Submission and Monitoring task wiki page: http://grid.pd.infn.it/omii/

[OPENSAML] The OpenSAML Toolkit, https://spaces.internet2.edu/display/OpenSAML

[ETICS] ETICS home page, http://etics.web.cern.ch/etics/

[JSDL] A. Savva (editor), Job Submission Description Language (JSDL) Specification, Version 1.0, GGF November 2005, available at http://www.gridforum.org/documents/GFD.56.pdf

[BES] I. Foster et al., OGSA Basic Execution Service, Version 1.0, August 2007. available at http://www.ogf.org/documents/GFD108.pdf

[HPCP] M. Humphrey et al., JSDP HPC Profile Application Extension, Version 1.0, october 2006

[AWARE] An easy Way to Access grid Resources Project, http://www.a-ware-project.eu/