Editors:
Andreas Savva, Fujitsu
Hiro Kishimoto, Fujitsu
Steven Newhouse, University of Southampton
Darren Pulsipher, Ovoca, LLC

18 April 2007

OGSA™ EMS Architecture Scenarios

Version 1.0

Status of This Document

This document provides information to the community on the scenarios used to refine the OGSA EMS architecture. It does not define any standards or technical recommendations. Distribution is unlimited.

Trademarks

OGSA is a trademark of the Open Grid Forum.

Abstract

The OGSA EMS (execution management services) architecture defines a number of services and XML document types. To illustrate their combined use a set of scenarios is described in this document. The scenarios cover basic job submission, selection of resources, and deployment of resources. The emphasis is on using already published specifications, or specifications in the last stages of approval. In all scenarios a list of required specifications is included.

Contents

1 INTRODUCTION 3
1.1 SUMMARY OF SPECIFICATIONS USED IN THE SCENARIOS 3
2 FUNDAMENTAL JOB EXECUTION 4
2.1 DIRECT JOB EXECUTION 5
2.2 INDIRECT JOB EXECUTION 6
2.3 JOB TERMINATION 7
3 SELECTED JOB EXECUTION 9
4 DEPLOYMENT AND CONFIGURATION 10
4.1 DEPLOYING AN APPLICATION 10
4.2 DEPLOYING AN APPLICATION USING THE APPLICATION CONTENTS SERVICE 13
4.3 UNDEPLOYING AN APPLICATION 15
5 SECURITY CONSIDERATIONS 17
6 GLOSSARY 17
INTELLECTUAL PROPERTY STATEMENT 19
DISCLAIMER 19
FULL COPYRIGHT NOTICE 19
APPENDIX A. RUNNING A BLAST (BASIC LOCAL ALIGNMENT SEARCH TOOL) JOB . 21
A.1. JSDL DOCUMENT SUBMITTED TO THE JOB MANAGER 21
A.2. CANDIDATE EXECUTION PLAN RETURNED BY EXECUTION PLANNING SERVICE TO JOB MANAGER 23
A.3. CDL DOCUMENT RETRIEVED BY JOB MANAGER TO PREPARE FOR DEPLOYMENT 25
A.4. CDL DOCUMENT SUBMITTED BY JOB MANAGER TO DEPLOYMENT SERVICE .. 26
A.5. CDL DOCUMENT AFTER DEPLOYMENT HAS FINISHED 28
A.6. JSDL DOCUMENT SUBMITTED TO BES 29

1 Introduction

The OGSA™ EMS (execution management services) architecture ([GFD.80] §3.4) has been under discussion for many years. In its entirety it is, quite rightly, a complex set of interdependent services. However, not all of these services are required to perform simpler operations within the EMS space. This document describes a set of scenarios that build upon each other to move towards the full EMS architecture.

The initial focus is on fundamental job execution scenarios as these are well understood and in common use in Grid infrastructures such as EGEE[1] and OSG[2]. Scenarios describing the selection of resources for execution as well as the deployment of the necessary software on the resources are also included. Later versions of this document will expand on other more advanced EMS scenarios, such as the interaction of execution and data services in support of job execution; the role of selection services in deployment and configuration; and provisioning.

The emphasis is on using already published specifications, or specifications in the last stages of approval. In all scenarios a list of required specifications is included. A summary of the specifications used in the scenarios is given in §1.1. A number of special terms, abbreviations, and acronyms are explained in the Glossary (see §6).

1.1 Summary of Specifications Used in the Scenarios

This section gives a brief overview of the specifications used in the scenarios: OGSA Basic Execution Service (BES) [BES]; Job Submission Description Language (JSDL) [GFD.56]; CDDLM Deployment API [GFD.69]; CDDLM Configuration Description Language (CDL) [GFD.85]; Applications Contents Service (ACS) [GFD.73]; and OGSA Resource Selection Services (RSS) [RSS].

1.1.1 OGSA Basic Execution Service

OGSA BES is a set of Web services interfaces for creating, monitoring, and controlling activities. Clients define activities using JSDL (see §1.1.2) and create an activity instance by submitting a JSDL document to a BES container—a container providing the BES interfaces. A BES container implementation executes each activity that it accepts on an appropriate computational resource. Examples of such resources are a single computer; a cluster managed through a resource manager such as Load Leveler, Sun Grid Engine, Portable Batch System, or Condor; a Web service hosting environment; or even another BES container.

Activities are referenced by an EPR (endpoint reference [WS-Addressing]) and may provide their own (management) interfaces.

1.1.2 Job Submission Description Language

JSDL provides an XML-based language specifically for describing single job (activity) submission requirements.

---

[1] http://www.eu-egee.org/

[2] http://www.opensciencegrid.org/

The JSDL vocabulary is informed by a number of existing job management systems such as Condor, Globus, Load Sharing Facility, Portable Batch System, Sun Grid Engine, and Unicore. It includes elements for describing job identification (e.g., job name, project); application to execute (e.g., application name, version), including additional description specifically for POSIX applications (e.g., path to executable); data requirements (e.g., files to stage in or stage out); and resource requirements (e.g., type and characteristics of CPU, memory requirements, operating system).

1.1.3 CDDLM Deployment API

The CDDLM Deployment API is a Web services (WSRF[3]-based) API for deploying applications to one or more target hosts; and for managing the deployment lifecycle. A deployment descriptor described in XML (e.g., CDL, see §1.1.4) defines the deployment dependencies. Once an application is deployed a set of properties and operations defined by the CDDLM component model (not described here) are provided, which can be used to examine the state of the deployed application and retrieved information about its environment.

1.1.4 CDDLM Configuration Description Language

CDL is an XML-based language that supports the declarative description of system configuration as a hierarchy of components. A CDL document defines the order and deployment requirements of a system. Examples of systems are the installation of a POSIX application such as BLAST; and the installation of a complex application such as a three-tier application.

1.1.5 Applications Contents Service

ACS defines a Web services interface to a repository containing grid application content—for example, program binaries; configuration scripts and data; descriptions of required hardware environments. Each application is logically contained in an Application Archive, which may be retrieved in its entirety in a standard format. The ACS repository manages the application content and provides access to it by other services; it does not act on (interpret or execute) the contents.

1.1.6 OGSA Resource Selection Services

RSS defines Web services interfaces, such as an Execution Planning Service, for choosing services or resources on which to perform an operation or execute an activity. An EPS takes an abstract plan of an execution—JSDL document—and suggests particular execution services—BES containers—upon which the abstract plan may be carried out. It returns a set of (concrete) candidate execution plans.

Reservation of resources is not part of RSS and there is no guarantee that a candidate execution plan can in fact be executed.

2 Fundamental Job Execution

Fundamental job execution deals with direct or indirect job submission and management. *Direct* here means that the user, through some user agent, submits and

[3] Web Services Resource Framework, see http://www.oasis-open.org/committees/wsrf/.

monitors a job execution; *indirect* that the user *delegates* the management of the job to a job manager.

2.1 Direct Job Execution

2.1.1 Assumptions

- The User agent knows which BES container to use.
- The required BES container is already setup.

2.1.2 Entities

- User agent
- BES container

2.1.3 Description

The client (User agent) contacts the BES container directly and submits a request to start an activity described by a simple JSDL document. Progress is monitored directly by the User agent. (It may also be possible for the User agent to poll or receive notifications directly from the Activity (not shown).)

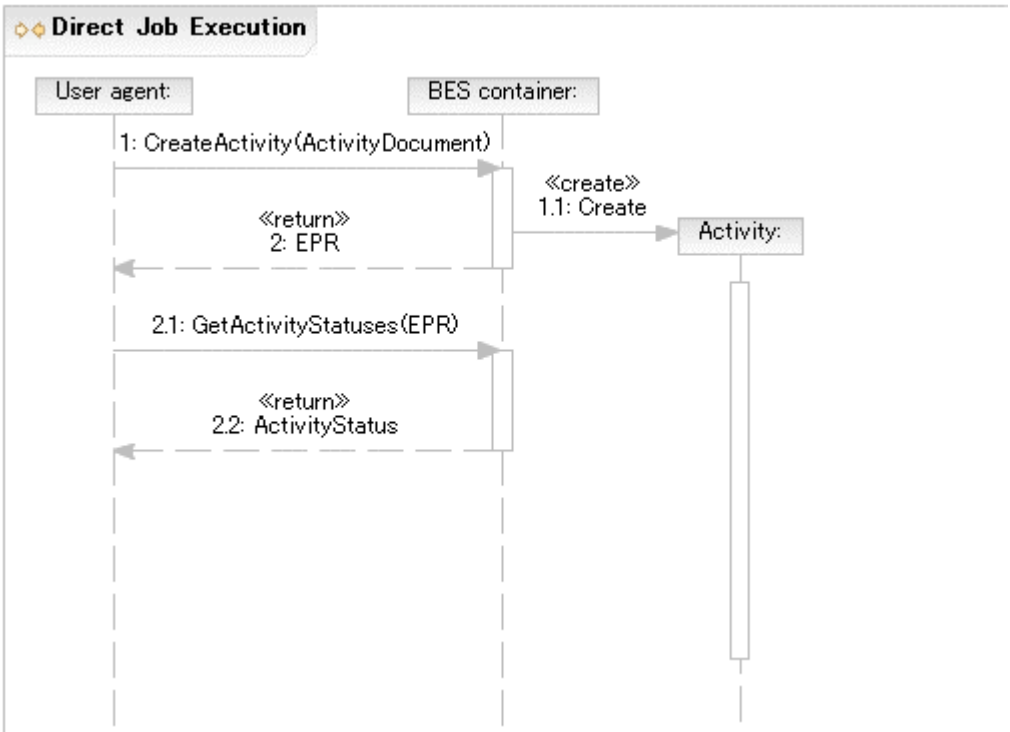

Figure 1 Direct Job Execution

2.1.4 Required Specifications

This scenario requires the BES[BES] and JSDL[GFD.56] specifications.

2.2 Indirect Job Execution

2.2.1 Assumptions

- The User agent knows which Job manager to use.
- The Job manager knows which BES container to use.
- The required BES container is already setup.

2.2.2 Entities

- User agent
- Job manager
- BES container

2.2.3 Description

To enable more sophisticated EMS scenarios it is important that the user can delegate decision making and control to an entity that is more sophisticated than a user agent—a job manager. The User agent contacts the Job manager, specifying the job, which is then executed on an already setup BES container. Once the Activity is established it is monitored by the Job manager through the BES container or directly (not shown).

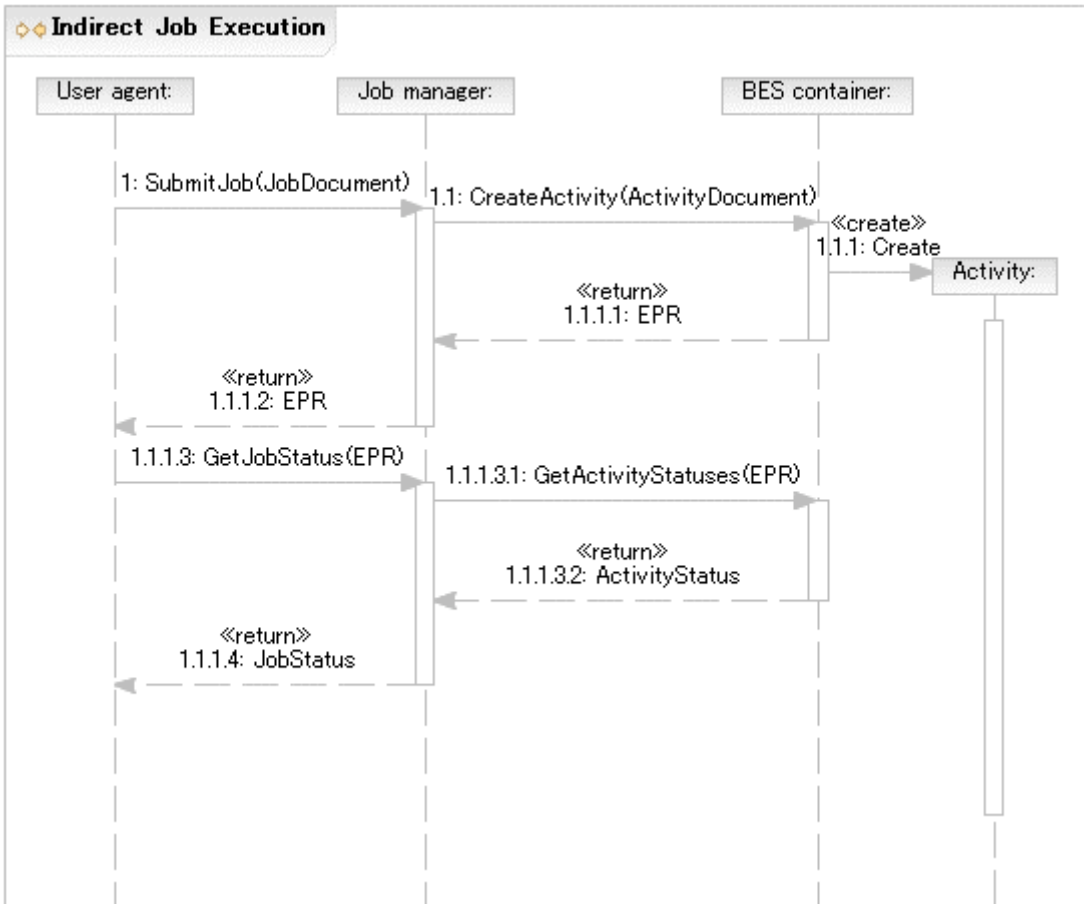

Figure 2 Indirect Job Execution

2.2.4 Required Specifications

This scenario requires the BES[BES], JSDL[GFD.56] specifications. A Job manager specification may also be required.

The BES container and Job manager have different responsibilities and are distinct roles in the sequence in Figure 2. It is possible, however, that a Job manager may have the same interface as a BES container.

2.3 Job Termination

The User agent may wait for the Activity to terminate—successfully or not—or request that the Activity is terminated.

2.3.1 Normal Termination of Job

2.3.1.1 Assumptions

- The User agent has a reference to the BES container.

2.3.1.2 Entities

- User agent
- BES container
- Activity

2.3.1.3 Description

The User agent waits until the Activity terminates normally. The User agent detects this by polling or receiving a notification (not shown).

It may also be possible for the User agent to poll or receive notifications directly from the Activity (not shown).

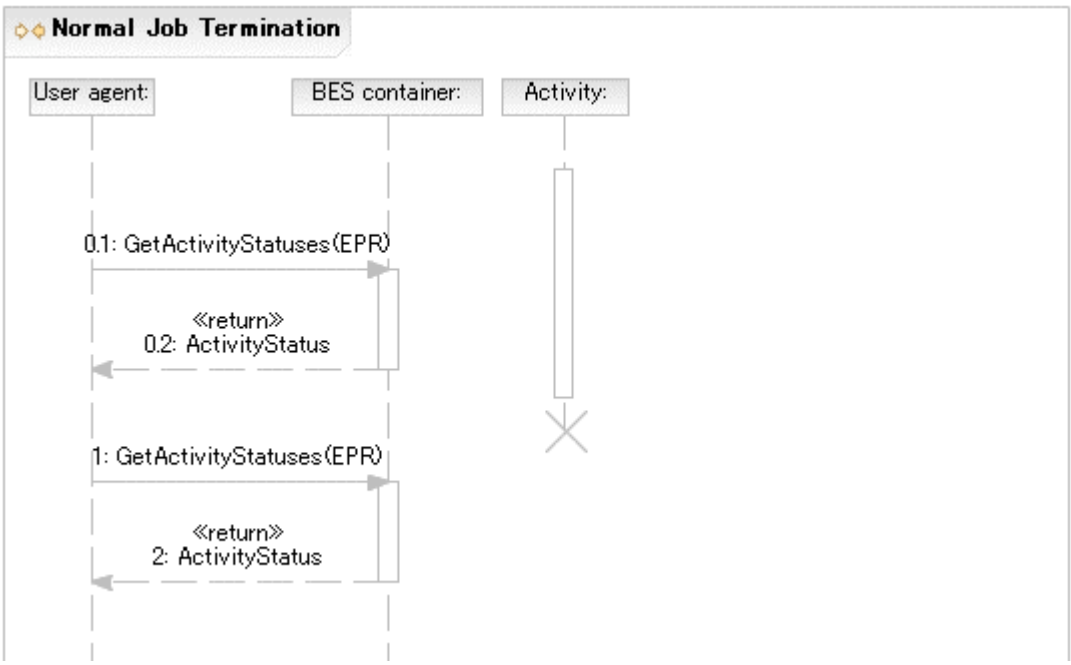

Figure 3 Normal Job Termination

2.3.1.4 Required Specifications

This scenario requires the BES[BES] specification.

2.3.2 Job killed by User Agent

2.3.2.1 Assumptions

- The User agent already has a reference to the BES container.

2.3.2.2 Entities

- User agent
- BES container
- Activity

2.3.2.3 Description

The User agent requests that the Activity is terminated.

It may also be possible for the User agent to terminate the Activity directly (not shown).

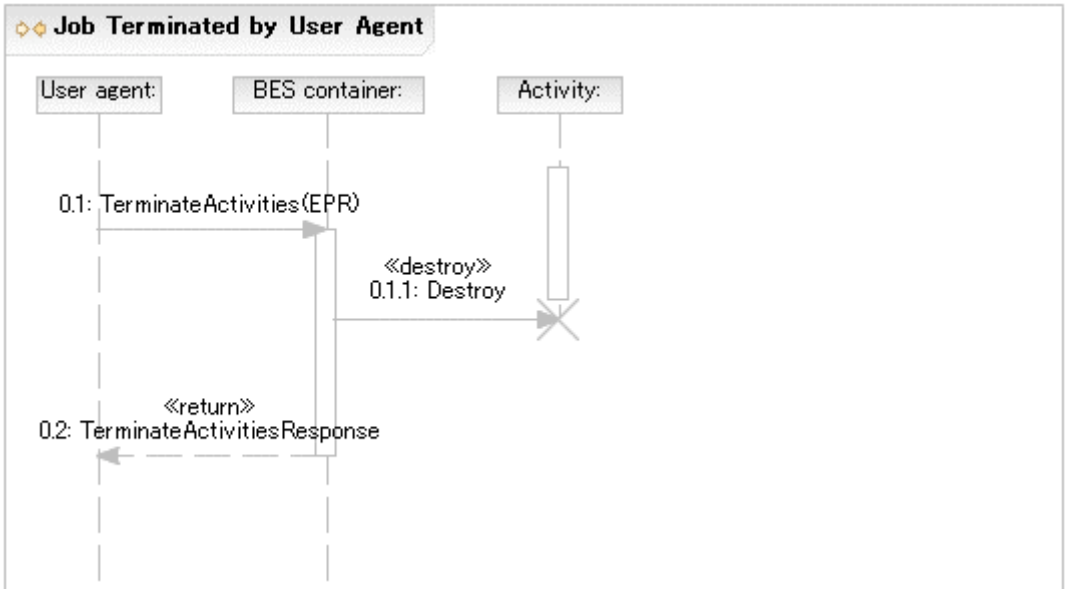

Figure 4 Job killed by User Agent

2.3.2.4 Required Specifications

This scenario requires the BES[BES] specification and an (unspecified) Activity interface specification.

2.3.2.5 Notes

In Figure 4 the Activity is shown as actually terminating before the TerminateActivities operation returns. This is for convenience only and is not

required behavior according to the BES specification. An activity may, in fact, not terminate for some time after the TerminateActivities operation returns. Eventual termination of an activity should be determined by other means[BES].

3 Selected Job Execution

3.1.1 Assumptions

- The User agent already has a reference to a Job manager.
- The Job manager already knows which Execution Planning Service (EPS) to use.
- The BES container has not been determined at submission time.
- There is a BES container already setup that can run the submitted job.

3.1.2 Entities

- User agent
- Job manager
- BES container
- EPS

3.1.3 Description

The User agent submits a JSDL document requiring the BLAST application to the Job manager as in the scenario described in §2.2. The Job manager uses the Execution Planning Service (EPS) (part of the Resource Selection Services (RSS)) to select a BES container. It submits the JSDL document to the EPS and receives a, possibly null, list of candidate execution plans for the job. A candidate execution plan is made up of at least a JSDL document (possibly refined) and a reference to a BES container. An example of a candidate execution plan is given in Appendix A.2. (Note that the JSDL document included in this plan requires further refinement before submission, however such refinement is not always necessary.)

For the purposes of this scenario the first plan returned is considered to be the best choice. The Job manager starts the requested activity by submitting the JSDL document returned in the candidate execution plan to the selected BES container as described in §2.2.

If the submission to the BES container in the first plan fails (not shown) then the next plan may be chosen instead as a failover option; or if there are no other plans left the job execution overall fails.

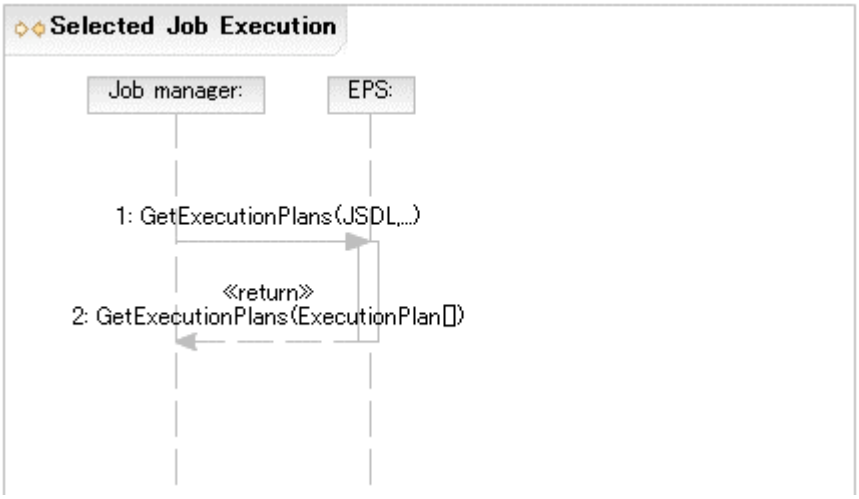

Figure 5 Selected Job Execution

3.1.4 Required Specifications

This scenario requires the BES[BES], JSDL[GFD.56], Job manager, and RSS[RSS] specifications.

3.1.5 Notes

1. The JSDL document returned in the candidate execution plan may be refined to allow the rewriting of things such as abstract resources (such as one identifying the abstract name of some sequence database) into a set of requirements for making that resource available at the suggested BES container (e.g. by extending the set of DataStaging elements, or by providing an appropriate CDL document).
2. Internally, the RSS may use a simple selection algorithm (e.g., number of pre-defined BES containers to round-robin the job request from the Job manager to one of these interfaces; or could obtain and select BES containers using an information service such as a service registry). With a defined selection language the RSS implementation can be more sophisticated by allowing a user defined selection policy to be applied to the available services in the registry.

4 Deployment and Configuration

4.1 Deploying an Application

A simple POSIX application (BLAST) is deployed to support an activity that is to be initiated within a BES container.

4.1.1 Assumptions

- The User agent already has a reference to a Job manager.
- The Job manager already knows which EPS to use.
- The BES container is determined at submission time—e.g., it may be specified as an argument to the submission operation—but the actual resource, a.k.a. host, will be chosen at deployment time.

- The BES container is not already setup to run the job.
- The Job manager has suitable binaries and other relevant information required to setup the BES container.
    a. For example, a BLAST binary, a BLAST database—possibly created by an earlier preparatory job—as well as a BLAST Application CDL document describing the deployment requirements. A later scenario (see §4.2) describes how such data can be retrieved from a repository service.
    b. The BLAST Application CDL document (see Appendix A.4) may have a number of *lazy elements*—e.g., Hostname, PathName. Also it can be assumed that the JSDL and CDL documents are created by tooling and are consistent—but not necessarily complete.
- The Job manager knows which deployment service can be used to deploy to the specified BES container.

4.1.2 Entities

- User agent
- Job manager
- Information service
- Deployment service
- BES container
- BLAST system

4.1.3 Description

The User agent submits to the Job manager a JSDL document requiring a BLAST application execution on a specific BES container. An example of such a JSDL document is given in Appendix A.1.

The Job manager queries the Information service, or the platform directly, and discovers that the specified BES container does not have the BLAST application installed on it.

The Job manager submits the BLAST CDL document to the Deployment service. An example of such a CDL document is given in Appendix A.4. The Deployment service completes any remaining lazy elements in the CDL document—e.g., the name of the resource (host) on which to deploy is a lazy element in this scenario—carries out the installation and returns an EPR (endpoint reference [WS-Addressing]) to the Job manager providing a reference to the installed software—the BLAST system. An example of a CDL document after deployment has been carried out is in Appendix A.5

The Information service entry of the resource is updated with the location of the BLAST application. (This update may be done by the resource itself, the Deployment service, the Job manager, etc.) This prevents the removal of the resource as the BLAST application is using it.

The Job manager uses the EPR to the installed software to retrieve information it may need if it has to refine the JSDL document further before submission. This is

effectively an operation to retrieve relevant chunks of the XML deployment tree (i.e., the instantiated CDL document) back to the Job manager for insertion into the JSDL document. An example of such a refined JSDL document is given in Appendix A.6.

The, now complete, JSDL document is submitted to the BES container on that platform.

The Information service entry of the BLAST application is updated with the information that the Activity now uses the application (not shown). (This update may be done by the Activity, the Job manager, etc.) If another Activity uses the same BLAST application a similar update to the Information service entry is carried out to make sure that the BLAST application remains available as long as it is in use.

Once the Activity completes successfully the Information service entry of the BLAST application is updated accordingly (not shown). The User agent is informed that the job is complete.

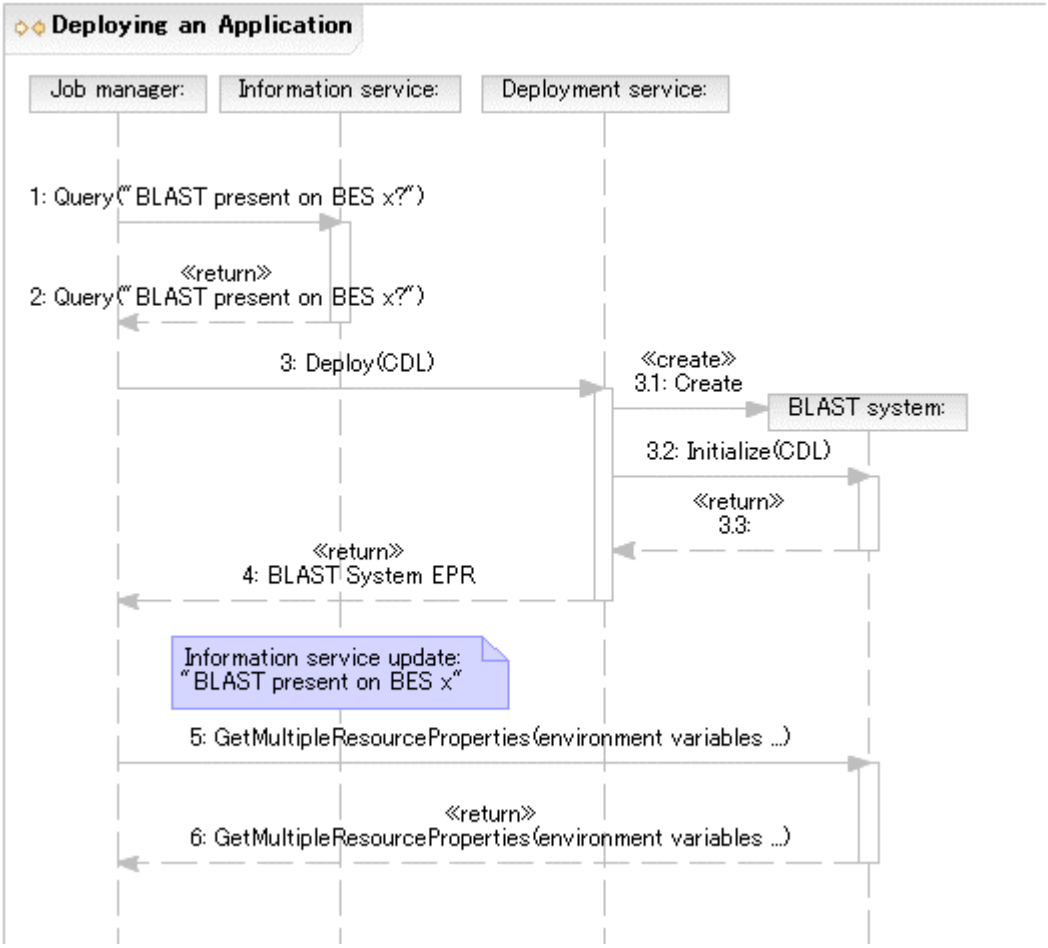

Figure 6 Deploying an Application

4.1.4 Required Specifications

This scenario requires the BES[BES], JSDL[GFD.56], Job manager, CDL[GFD.85], CDDLM Deployment API[GFD.69] and other supporting component specifications such as a CDL POSIXApplication component, Deployment service and (unspecified) Information service specifications.

4.1.5 Notes

In the simplest case the deployment service may be located on the platform hosting the BES container, or there is some other well known (pre-configured) relationship. In general a deployment service may, however, act as a proxy for deployment on platforms other than the one it is running on. It may also be possible that there are multiple such services in a system. It is therefore recommended that a deployment service advertises (somewhere, in a currently unspecified manner) where it is capable of deploying stuff to, for example, front end node for a cluster.

The deployment service operations called by the Job manager in the sequence in Figure 6 are different from those defined by the CDDLM Deployment API. The operations called by the Deployment service are those of the CDDLM Deployment API. The CDDLM Deployment *portal* is not shown for simplicity.

In the scenario above it is suggested that the Job manager may carry out some simple refinement of the JSDL document perhaps by retrieving information available after deployment. This functionality is not required of the Job manager and it is expected that, if necessary, a specialized service (currently undefined and left out for simplicity) may be available to carry out such refinements.

4.2 Deploying an Application Using the Application Contents Service

In this scenario a simple POSIX application (BLAST) needs to be deployed to support an activity that is to be initiated within a BES container.

4.2.1 Assumptions

- The User agent has a reference to a Job manager.
- The BES container is determined at submission time— e.g., it may be specified as an argument to the submission operation—but the actual resource, a.k.a. host, will be chosen at deployment time.
- The BES container is not already setup to run the job.
- The Job manager does NOT have suitable binaries and other relevant information required to setup the BES container.
- The Job manager has a reference to an Application Contents Service (ACS) Repository.
- The Application Contents Services Repository already has registered a BLAST Application Archive (AA)—addressable by an EPR—that contains all information required to install and configure BLAST.
- The Job manager knows which Deployment service can be used to deploy to the specified BES container.

4.2.2 Entities

- User agent
- Job manager
- ACS repository

- BLAST AA
- Deployment service
- BLAST system
- BES container

4.2.3 Description

As in the previous deployment scenario the User agent submits a JSDL document, which requires the BLAST application, to the Job manager. The BES container is specified as an argument to the submission operation. The Job manager determines that the BES container does not have BLAST.

The Job manager uses the contents of the JSDL ApplicationName and ApplicationVersion elements to search for a matching Application Archive (AA) in the ACS repository using the LookupArchives operation. The EPR of the AA that has the matching BLAST application is returned—the BLAST AA.

The Job manager uses the AA GetContents operation to retrieve the CDL document describing the BLAST deployment requirements. An example of such a CDL document is shown in Appendix A.2. (Comments made in previous scenarios about this CDL document also apply here.)

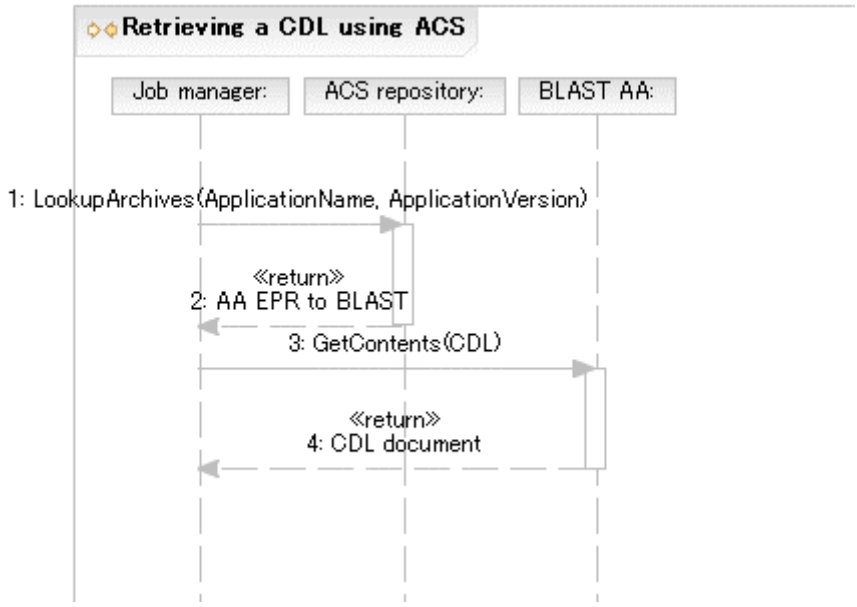

Figure 7 Retrieving a CDL document using ACS

The CDL document (which may still contain lazy elements) is then submitted to the Deployment service that can deploy to the platform hosting the BES container (see previous scenarios).

As part of the deployment the binary and other data required for deployment are retrieved from the ACS repository. For example, the BLAST system can use the ACS GetContents operation to retrieve the BLAST binary and possibly the BLAST database from BLAST AA.

The remaining steps are the same as in previous scenarios.

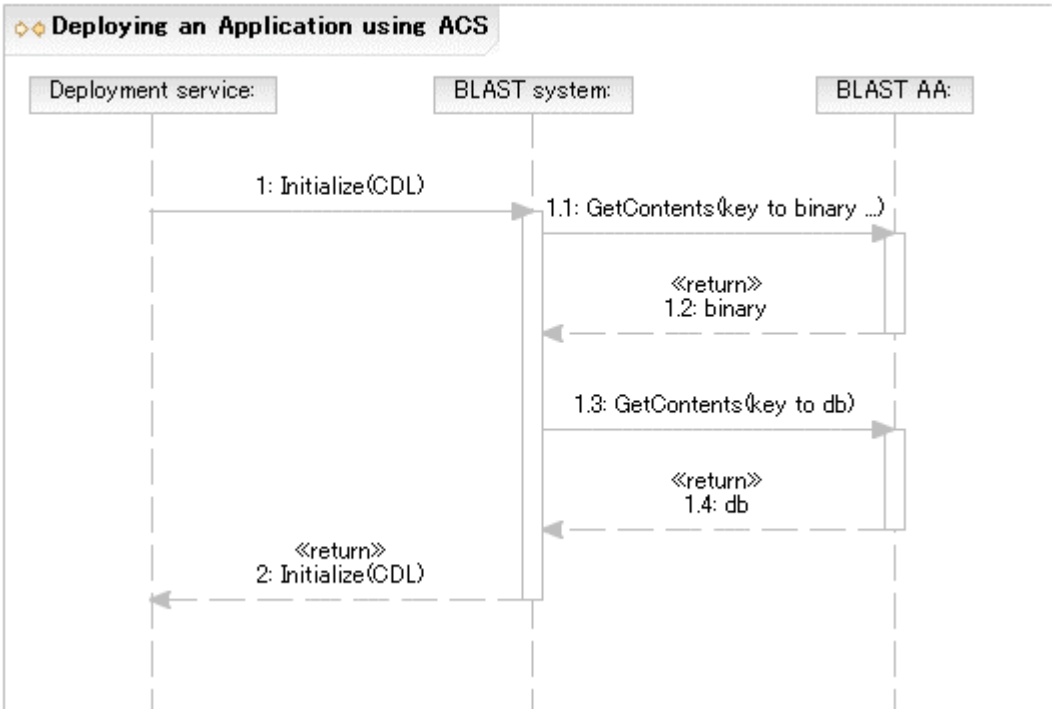

Figure 8 Deploying an Application using ACS

4.2.4 Required Specifications

This scenario requires the BES[BES], CDDLM Deployment API[GFD.69], JSDL[GFD.56], Job manager, ACS[GFD.73], CDL[GFD.85] and supporting CDL component specifications, and (unspecified) Information service specifications.

4.2.5 Notes

For simplicity it is assumed that matching ApplicationName and ApplicationVersion with an Application Archive is sufficient. In scenarios using the EPS this search-and-match may be done by the EPS and it may include a matching of Archive requirements with the Job description.

Alternatives to using the LookupArchives operation to locate an appropriate archive exist and may be preferable in real deployments. For example, it is possible to provide the EPR to the required Application Archive as part of the submission, e.g., inside the JSDL document as an extension to the Application element. Also systems may support naming applications using RNS (Resource Namespace Service [RNS]). For example, the JSDL ApplicationName element may contain an RNS application name, which can then be used to locate the corresponding archive in an ACS repository.

4.3 Undeploying an Application

4.3.1 Assumptions

- The Activity has terminated
- The Job manager knows which Deployment service can be used to undeploy the application represented by the BLAST system.

4.3.2 Entities

- Information service
- Deployment service
- BES container
- BLAST system

4.3.3 Description

Once the Activity completes successfully the Information service entry of the BLAST application is updated accordingly. When no other activities are registering an interest in a specific instance of a BLAST application on a resource then the Undeploy operation may be invoked on the Deployment service to remove this BLAST application from the resource.

The application may remain installed for some period of time if it is expected that it may be re-used, even if no activity is registering a specific interest for it.

If the BLAST application is removed then its entry in the Information service is also removed.

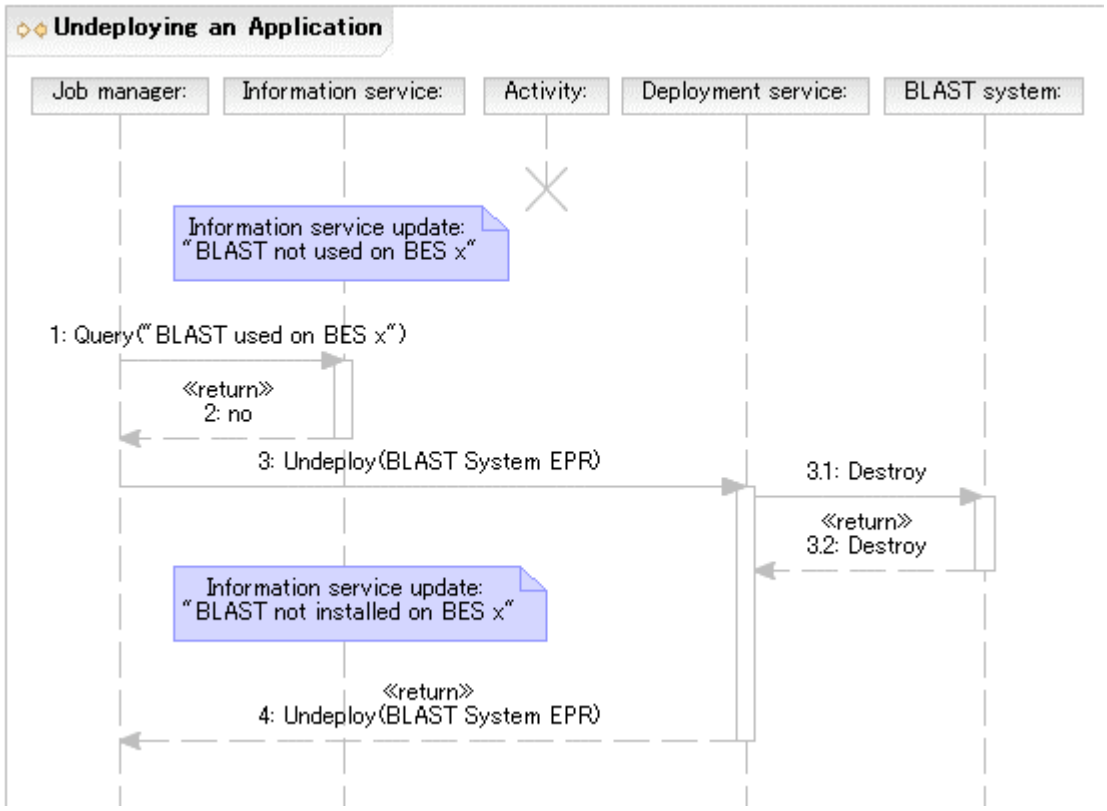

Figure 9 Undeploying an Application

4.3.4 Required Specifications

This scenario requires the Deployment service, CDDLM Deployment API[GFD.69] and (unspecified) Information service specifications.

4.3.5 Notes

Since the Job manager has the EPR to the BLAST system it is possible to call the Destroy operation directly. In the scenario, the Deployment service is used instead because further operations (not shown) may also be necessary to complete the removal. One example is updating the information service entry.

5 Security Considerations

This is a use case document and does not deal with security issues.

Security considerations of the referenced specifications should be thoroughly reviewed and appropriate mechanisms should be used to secure all service interactions.

It is worth calling out the requirement for appropriate authentication and authorization mechanisms. Also mechanisms for delegation of rights are required. In particular, the Job manager has to be able to delegate requestor access rights in such a way that each service provider involved in a job execution receives the appropriate (restricted) set of rights it needs. OGSA 1.5[GFD.80] §3.7 offers a detailed discussion of some of these issues.

Contributors

Editor Information:

Hiro Kishimoto
IT Core Laboratories
Fujitsu Laboratories
4-1-1, Kamikodanaka, Nakahara, Kawasaki City, Japan
Email: hiro.kishimoto@jp.fujitsu.com

Andreas Savva
IT Architecture Lab.
Fujitsu Laboratories
4-1-1, Kamikodanaka, Nakahara, Kawasaki City, Japan
Email: andreas.savva@jp.fujitsu.com

The following persons have contributed to this document: Donal Fellows, Andrew Grimshaw, Chris Smith, Jun Tatemura.

We are grateful to numerous colleagues for discussions on the topics covered in this document, in particular (in alphabetical order, with apologies to anybody we have missed) Michael Behrens, Dave Berry, Michel Drescher, Keisuke Fukui, Tom Maguire, Steve McGough, Dejan Milojicic, Mark Morgan, Jem Treadwell, as well as to the whole OGSA team for their invaluable input.

6 Glossary

A number of terms used in this document are defined here. These definitions draw from and are consistent with the OGSA Glossary[GFD.81] and offer additional explanation or clarification.

In the following table, words or phrases in *italics* are themselves defined in the table.

Term	Definition	Ref's
Activity	The smallest part of possibly a larger sequence of activities generated from a single submitted *job*. In this document a job consists of exactly one activity and the terms 'job' and 'activity' are sometimes used interchangeably.	
Candidate Execution Plan	A plan for executing an *activity*. At a minimum it consists of a *JSDL* document and a reference to the *BES* container where the JSDL document should be submitted to instantiate the required activity. Other information that may be used to prepare the environment for the activity, such as a *CDL* document and a reference to a *deployment* service, may also be included.	
BES	Basic Execution Service The entity that accepts requests (essentially *JSDL* documents) to instantiate an *activity*. Note that BES does not do *provisioning* or *deployment*.	[BES]
BLAST	Basic Local Alignment Search Tool	[GFD.81]
CDL	Configuration Description Language	[GFD.85]
Deployment	The instantiation of an environment on a resource to meet the needs of a specific *activity* such as a submitted *job*. Deployment is generally a 'lightweight' action in that it may just provide a binary, dataset, etc., onto the resource. This is in contrast to *provisioning*. Once used by the specified activity that environment may be removed immediately; or left as part of a 'cache' to be cleaned up or reused at a later date. (A 'keep-alive' model might be needed in the latter case.) For example, an activity wants to run Gaussian. The deployment actions would be to download and install the required version for the selected platform.	
Deployment Service	The entity that accepts a *deployment* (or undeployment) action defined by a *CDL* document to instantiate an environment on a resource to support an *activity*.	
EMS	Execution Management Services A capability defined by OGSA.	[GFD.80]
EPR	A WS-Addressing Endpoint Reference	[WS-Addressing]
EPS	Execution Planning Service Part of the Resource Selection Services.	[RSS]
Job Manager	The entity that accepts a *job* (defined by a *JSDL* document) from the user agent. The Job Manager coordinates further invocations and interactions with other elements of the EMS architecture, such as the *EPS*.	

Term	Definition	Ref's
JSDL	Job Submission Description Language	[GFD.56]
Provisioning	The instantiation of an environment on a resource (i.e., a *deployment*) that may be used by more than one *activity*. Generally such *deployments* are 'heavyweight' in that they may take substantial time to instantiate. Provisioning actions may be triggered by a specific activity (e.g., following a *job* submission); or they may be the result of a general response to the state of the current resources in the system, possibly defined by a policy. An example of such a policy: "If my free Red Hat Enterprise Linux Version 4 resources drop to 5%, dynamically provision new nodes until the total free nodes have increased to 15%."	
RNS	Resource Namespace Service	[RNS]

Intellectual Property Statement

Disclaimer

Full Copyright Notice

purpose of developing Grid Recommendations in which case the procedures for copyrights defined in the OGF Document process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the OGF or its successors or assignees.

References

[BES] A. Grimshaw, S. Newhouse, D. Pulsipher, M. Morgan, "OGSA Basic Execution Service Version 1.0 (Draft 26)," OGSA BES-WG, Open Grid Forum, 7 September 2006.

[GFD.56] A. Anjomshoaa, F. Brisard, M. Drescher, D. Fellows, A. Ly, S. McGough, D. Pulsipher, and A. Savva (ed.), "Job Submission Description Language (JSDL) Specification, Version 1.0," Global Grid Forum, Lemont, Illinois, U.S.A., GFD.56, November 2005. http://www.ogf.org/gf/docs/?final

[GFD.69] S. Loughran (ed.), "Configuration Description, Deployment and Lifecycle Management: CDDLM Deployment API," Global Grid Forum, Lemont, Illinois, U.S.A., GFD.69, March 2006. http://www.ogf.org/gf/docs/?final

[GFD.73] K. Fukui (ed.), "Application Contents Service Specification 1.0," Open Grid Forum, Lemont, Illinois, U.S.A., GFD.73, May 2006. http://www.ogf.org/gf/docs/?final

[GFD.80] I. Foster, H. Kishimoto, A. Savva, D. Berry, A. Djaoui, A. Grimshaw, B. Horn, F. Maciel, F. Siebenlist, R. Subramaniam, J. Treadwell, and J. Von Reich, "The Open Grid Services Architecture, Version 1.5," Global Grid Forum, Lemont, Illinois, U.S.A., GFD.80, July 2006. http://www.ogf.org/gf/docs/?final

[GFD.81] J. Treadwell (ed.), "Open Grid Services Architecture Glossary of Terms Version 1.5," Global Grid Forum, Lemont, Illinois, U.S.A., GFD.81, July 2006. http://www.ogf.org/gf/docs/?final

[GFD.85] J. Tatemura (ed.), "Configuration Description, Deployment and Lifecycle Management: Configuration Description Language Version 1.0," Open Grid Forum, Lemont, Illinois, U.S.A., GFD.85, August 2006. http://www.ogf.org/gf/docs/?final

[RNS] M. Pereira, O. Tatebe, L. Luan, T. Anderson, "Resource Namespace Service Specification," Open Grid Forum, Lemont, Illinois, U.S.A., GWD-R, September 2006. Available at http://www.ogf.org/Public_Comment_Docs/Documents/Oct-2006/RNS-Specification-20060922.pdf

[RSS] D. Fellows, "OGSA Resource Selection Services (Draft 3)," OGSA RSS-WG, Open Grid Forum, December 2006. Available at https://forge.gridforum.org/sf/go/doc14076

[WS-Addressing] Gudgin, M., Hadley, M., Rogers, T. (eds.), "Web Services Addressing 1.0 – Core (WS-Addressing)," W3C Recommendation, 9 May 2006. http://www.w3.org/TR/2006/REC-ws-addr-core-20060509

Appendix A. Running a BLAST (Basic Local Alignment Search Tool) job

A.1. JSDL document submitted to the Job manager

This appendix shows an example JSDL document that may be initially submitted to the Job Manager. According to the scenarios in this document the JSDL document must contain at a minimum the Application name.

Highlighted text (in yellow) indicates portions that may need refinement or have linkage to CDL. This JSDL document is missing the following elements: Executable, Environment, CandidateHosts and part of the configuration for the required Filesystems. These will be provided after deployment is finished.

It should be noted that according to JSDL 1.0[GFD.56] the Executable element is required by the schema definition. This element is treated as optional in the examples to illustrate how refinement may be carried out.

```
<?xml version="1.0" encoding="UTF-8"?>
<jsdl:JobDefinition
   xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl"
   xmlns:jsdl-posix="http://schemas.ggf.org/jsdl/2005/11/jsdl-posix"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <jsdl:JobDescription>
      <jsdl:JobIdentification>
         <jsdl:JobName>Blast1</jsdl:JobName>
         <jsdl:Description>Blast query number 1</jsdl:Description>
         <jsdl:JobProject>BlastProject</jsdl:JobProject>
      </jsdl:JobIdentification>
      <jsdl:Application>
         <jsdl:ApplicationName>BlastN</jsdl:ApplicationName>
         <jsdl:ApplicationVersion>2.2.13</jsdl:ApplicationVersion>
         <jsdl:Description>BlastN performs nucleotide similarity
            searching</jsdl:Description>
         <jsdl-posix:POSIXApplication>
            <!-- EXECUTABLE LOCATION NOT KNOWN YET -->
            <jsdl-posix:Argument>-p</jsdl-posix:Argument>
            <jsdl-posix:Argument>blastn</jsdl-posix:Argument>
            <jsdl-posix:Argument>-d</jsdl-posix:Argument>
            <jsdl-posix:Argument>/db/ncbiblast/est
               </jsdl-posix:Argument>
            <jsdl-posix:Argument>-T</jsdl-posix:Argument>
            <jsdl-posix:Argument>T</jsdl-posix:Argument>
            <jsdl-posix:Input filesystemName="HOME">sequences1.txt
               </jsdl-posix:Input>
            <jsdl-posix:Output filesystemName="HOME">sequences1.html
               </jsdl-posix:Output>
            <jsdl-posix:Error filesystemName="HOME">sequences1.err
               </jsdl-posix:Error>
            <jsdl-posix:WorkingDirectory filesystemName="HOME">
               blastqueries</jsdl-posix:WorkingDirectory>
            <!--ENVIRONMENT TO BE INSERTED POST DEPLOYMENT-->
            <!--Limits to be defined based on what user is allowed-->
            <jsdl-posix:UserName>csmith</jsdl-posix:UserName>
            <jsdl-posix:GroupName>bio</jsdl-posix:GroupName>
         </jsdl-posix:POSIXApplication>
      </jsdl:Application>
```

```
      <jsdl:Resources>
         <!--Once the host is determined it is included here-->
         <jsdl:FileSystem name="TMP">
            <jsdl:FileSystemType>temporary</jsdl:FileSystemType>
            <jsdl:Description>Well-known 'name' for temporary space
               that does not necessarily persist after the job
               terminates.
            </jsdl:Description>
            <jsdl:DiskSpace>
               <jsdl:LowerBoundedRange>10737418240.0
                   </jsdl:LowerBoundedRange>
            </jsdl:DiskSpace>
            <!--Detailed setup to be provided by a CDL document.-->
            <!--Some values may be added here when refining.-->
         </jsdl:FileSystem>
         <jsdl:FileSystem name="HOME">
            <jsdl:FileSystemType>normal</jsdl:FileSystemType>
            <jsdl:Description>Chris's home directory
               </jsdl:Description>
            <!--Detailed setup to be provided by a CDL document.-->
            <!--Some values may be added here when refining.-->
         </jsdl:FileSystem>
         <jsdl:FileSystem name="DB">
            <jsdl:FileSystemType>normal</jsdl:FileSystemType>
            <jsdl:Description>Formatted BLAST database
               </jsdl:Description>
            <!--Detailed setup to be provided by a CDL document.-->
            <!--Some values may be added here when refining.-->
            <jsdl:MountPoint>/db/ncbiblast</jsdl:MountPoint>
         </jsdl:FileSystem>
         <jsdl:ExclusiveExecution>true</jsdl:ExclusiveExecution>
         <jsdl:TotalCPUCount>
            <jsdl:Exact>1.0</jsdl:Exact>
         </jsdl:TotalCPUCount>
      </jsdl:Resources>
      <jsdl:DataStaging>
         <jsdl:FileName>blastqueries/sequences1.txt</jsdl:FileName>
         <jsdl:FilesystemName>HOME</jsdl:FilesystemName>
         <jsdl:CreationFlag>overwrite</jsdl:CreationFlag>
         <jsdl:Source>
            <jsdl:URI>
               http://csmith.otherhost.com/blastqueries/sequences1.txt
            </jsdl:URI>
         </jsdl:Source>
      </jsdl:DataStaging>
      <jsdl:DataStaging>
         <jsdl:FileName>blastqueries/sequences1.html</jsdl:FileName>
         <jsdl:FilesystemName>HOME</jsdl:FilesystemName>
         <jsdl:CreationFlag>overwrite</jsdl:CreationFlag>
         <jsdl:Target>
            <jsdl:URI>
            http://csmith.otherhost.com/blastqueries/sequences1.html
            </jsdl:URI>
         </jsdl:Target>
      </jsdl:DataStaging>
      <jsdl:DataStaging>
         <jsdl:FileName>blastqueries/sequences1.err</jsdl:FileName>
         <jsdl:FilesystemName>HOME</jsdl:FilesystemName>
         <jsdl:CreationFlag>append</jsdl:CreationFlag>
         <jsdl:Target>
            <jsdl:URI>
```

```
            http://csmith.otherhost.com/blastqueries/sequences1.err
            </jsdl:URI>
         </jsdl:Target>
      </jsdl:DataStaging>
   </jsdl:JobDescription>
</jsdl:JobDefinition>
```

A.2. Candidate execution plan returned by Execution planning service to Job manager

An example of a candidate execution plan returned by the Execution planning service to the Job manager includes a JSDL document; a reference to the BES container; a quality of service element; and a validity period. In this case the JSDL document returned in the plan is unchanged. (It is the same as in A.1.)

```
<?xml version="1.0" encoding="UTF-8"?>
<eps:CandidateExecutionPlan
   xmlns:eps="http://schemas.ggf.org/rss/2006/eps-draft"
   xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl"
   xmlns:jsdl-posix="http://schemas.ggf.org/jsdl/2005/11/jsdl-posix"
   xmlns:wsa=" http://www.w3.org/2005/08/addressing"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <jsdl:JobDefinition>
      <jsdl:JobDescription>
         <jsdl:JobIdentification>
            <jsdl:JobName>Blast1</jsdl:JobName>
            <jsdl:Description>Blast query number 1</jsdl:Description>
            <jsdl:JobProject>BlastProject</jsdl:JobProject>
         </jsdl:JobIdentification>
         <jsdl:Application>
            <jsdl:ApplicationName>BlastN</jsdl:ApplicationName>
            <jsdl:ApplicationVersion>2.2.13</jsdl:ApplicationVersion>
            <jsdl:Description>BlastN performs nucleotide similarity
                        Searching
            </jsdl:Description>
            <jsdl-posix:POSIXApplication>
               <!-- EXECUTABLE LOCATION NOT KNOWN YET -->
               <jsdl-posix:Argument>-p</jsdl-posix:Argument>
               <jsdl-posix:Argument>blastn</jsdl-posix:Argument>
               <jsdl-posix:Argument>-d</jsdl-posix:Argument>
               <jsdl-posix:Argument>/db/ncbiblast/est
               </jsdl-posix:Argument>
               <jsdl-posix:Argument>-T</jsdl-posix:Argument>
               <jsdl-posix:Argument>T</jsdl-posix:Argument>
               <jsdl-posix:Input filesystemName="HOME">
                   sequences1.txt
               </jsdl-posix:Input>
               <jsdl-posix:Output filesystemName="HOME">
                   sequences1.html
               </jsdl-posix:Output>
               <jsdl-posix:Error filesystemName="HOME">
                   sequences1.err
               </jsdl-posix:Error>
               <jsdl-posix:WorkingDirectory filesystemName="HOME">
                   Blastqueries
               </jsdl-posix:WorkingDirectory>
               <!--ENVIRONMENT TO BE INSERTED POST DEPLOYMENT-->
               <!--Limits to be defined based on what the user is
                   allowed-->
               <jsdl-posix:UserName>csmith</jsdl-posix:UserName>
```

```
            <jsdl-posix:GroupName>bio</jsdl-posix:GroupName>
         </jsdl-posix:POSIXApplication>
      </jsdl:Application>
      <jsdl:Resources>
         <!--Once the host is determined it is included here-->
         <jsdl:FileSystem name="TMP">
            <jsdl:FileSystemType>temporary</jsdl:FileSystemType>
            <jsdl:Description>Well-known 'name' for temporary space
                that does not necessarily persist after the
                job terminates.
            </jsdl:Description>
            <jsdl:DiskSpace>
                <jsdl:LowerBoundedRange>
                      10737418240.0
                </jsdl:LowerBoundedRange>
            </jsdl:DiskSpace>
            <!--Detailed setup to be provided by a CDL document.
                Some values may be added here when refining.-->
         </jsdl:FileSystem>
         <jsdl:FileSystem name="HOME">
            <jsdl:FileSystemType>normal</jsdl:FileSystemType>
            <jsdl:Description>Chris's home directory
            </jsdl:Description>
            <!--Detailed setup to be provided by a CDL document.
                Some values may be added here when refining.-->
         </jsdl:FileSystem>
         <jsdl:FileSystem name="DB">
            <jsdl:FileSystemType>normal</jsdl:FileSystemType>
            <jsdl:Description>Formatted BLAST database
                </jsdl:Description>
            <!--Detailed setup to be provided by a CDL document.
                Some values may be added here when refining.-->
                <jsdl:MountPoint>/db/ncbiblast</jsdl:MountPoint>
         </jsdl:FileSystem>
         <jsdl:ExclusiveExecution>true</jsdl:ExclusiveExecution>
         <jsdl:TotalCPUCount>
            <jsdl:Exact>1.0</jsdl:Exact>
         </jsdl:TotalCPUCount>
      </jsdl:Resources>
      <jsdl:DataStaging>
         <jsdl:FileName>
            blastqueries/sequences1.txt
         </jsdl:FileName>
         <jsdl:FilesystemName>HOME</jsdl:FilesystemName>
         <jsdl:CreationFlag>overwrite</jsdl:CreationFlag>
         <jsdl:Source>
            <jsdl:URI>
            http://csmith.otherhost.com/blastqueries/sequences1.txt
            </jsdl:URI>
         </jsdl:Source>
      </jsdl:DataStaging>
      <jsdl:DataStaging>
         <jsdl:FileName>
            blastqueries/sequences1.html
         </jsdl:FileName>
         <jsdl:FilesystemName>HOME</jsdl:FilesystemName>
         <jsdl:CreationFlag>overwrite</jsdl:CreationFlag>
         <jsdl:Target>
            <jsdl:URI>
          http://csmith.otherhost.com/blastqueries/sequences1.html
            </jsdl:URI>
```

```
            </jsdl:Target>
         </jsdl:DataStaging>
         <jsdl:DataStaging>
            <jsdl:FileName>
               blastqueries/sequences1.err
            </jsdl:FileName>
            <jsdl:FilesystemName>HOME</jsdl:FilesystemName>
            <jsdl:CreationFlag>append</jsdl:CreationFlag>
            <jsdl:Target>
               <jsdl:URI>
                http://csmith.otherhost.com/blastqueries/sequences1.err
               </jsdl:URI>
            </jsdl:Target>
         </jsdl:DataStaging>
      </jsdl:JobDescription>
   </jsdl:JobDefinition>
   <eps:BESReference>
      <wsa:Address>
         http://server.com:8000/serv/BasicExecution
      </wsa:Address>
      <wsa:ReferenceProperties>
         SomeReferenceData
      </wsa:ReferenceProperties>
   </eps:BESReference>
   <eps:QualityOfService>
      <eps:StartDelays>
      <!-- May take up to 10 minutes to get through the queue -->
      <eps:DelayRange from="0" to="600"/>
      </eps:StartDelays>
      <eps:ExecutionDuration>
         <!--Will take 60 to 90 seconds to
               actually run once started-->
         <eps:RuntimeRange from="60" to="90"/>
      </eps:ExecutionDuration>
   </eps:QualityOfService>
   <!-- Most validities would be shorter than this! -->
   <eps:Validity
         notBefore="2006-01-01T00:00+00:00"
         notAfter="2008-12-31T23:59+00:00"/>
</eps:CandidateExecutionPlan>
```

A.3. CDL document retrieved by Job manager to prepare for deployment

This appendix shows an example initial CDL document that may be retrieved by the Job manager. This CDL document assumes the definition of BasicPosixComponent, which is not described in this document.

A number of elements, highlighted (in yellow) in the document below, are defined as *lazy* by extension from the assumed BasicPosicComponent: PATH, TMPDIR, HOSTNAME. The values of these elements will be provided as part of the deployment.

```
<?xml version="1.0" encoding="UTF-8"?>
<cdl:cdl
targetNamespace="http://cddlm.org/component-model-example"
xmlns="http://cddlm.org/component-model-example"
xmlns:cdl="http://www.gridforum.org/namespaces/2005/02/cddlm/CDL-1.0"
xmlns:cmp="http://www.gridforum.org/cddlm/components/2005/02"
```

```
xmlns:bpc="http://www.gridforum.org/ogsa/BasicPosixComponent/2006/06"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<cdl:system>
   <Blast cdl:extends="bpc:PosixApplication">
      <cdl:documentation>
         Configuration data for BLAST binary deployment.
      </cdl:documentation>
      <BlastApp cdl:extends="bpc:Application">
         <Binary type="tar">http://blast.app.com/blast.tar</Binary>
         <UserName>csmith</UserName>
         <GroupName>bio</GroupName>
         <Mode>755</Mode>
      </BlastApp>
      <TmpFileSystem cdl:extends="bpc:FileSystem">
        <cdl:documentation>
            Configuration data for temporary file system.
        </cdl:documentation>
        <FileSystemType>temporary</FileSystemType>
        <DiskSpace>
          <LowerBoundedRange>10737418240</LowerBoundedRange>
        </DiskSpace>
      </TmpFileSystem>
      <HomeFileSystem cdl:extends="bpc:FileSystem">
        <cdl:documentation>
            Configuration data for home file system.
        </cdl:documentation>
        <FileSystemType>normal</FileSystemType>
        <MountSource>server.acme.com:/home/csmith</MountSource>
        <MountPoint>/home/csmith</MountPoint>
      </HomeFileSystem>
      <DataBaseFileSystem cdl:extends="bpc:FileSystem">
         <cdl:documentation>
            Configuration data for formatted database file system.
         </cdl:documentation>
         <FileSystemType>normal</FileSystemType>
         <MountSource>server.acme.com:/db/ncbiblast</MountSource>
         <MountPoint>/db/ncbiblast</MountPoint>
      </DataBaseFileSystem>
      <PATH cdl:extends="bpc:EnvironmentVariable"/>
          <!--Proper value of PATH environment variable -->
          <!-- will be available after deployment.-->
      <TMPDIR cdl:extends="bpc:EnvironmentVariable"/>
          <!--Proper value of TMPDIR environment variable -->
          <!--will be available after deployment.-->
       <HOSTNAME cdl:extends=”bpc:HostName”/>
          <!--Proper value of HOSTNAME ‘variable’-->
          <!--will be filled in before submission.-->
          <!--Note this is not according to the CDDLM spec-->
          <!--and is mainly for illustration.-->
   </Blast>
 </cdl:system>
</cdl:cdl>
```

A.4. CDL Document submitted by Job manager to Deployment service

The CDL template after the Job manager has (possibly) added some values. In this example the Job manager has added the HOSTNAME value (highlighted in blue).

Otherwise the CDL document is identical to the one shown in A.2. (Portions highlighted in yellow have not been resolved yet.)

```
<?xml version="1.0" encoding="UTF-8"?>
<cdl:cdl
   targetNamespace="http://cddlm.org/component-model-example"
   xmlns="http://cddlm.org/component-model-example"
   xmlns:cdl="http://www.gridforum.org/namespaces/2005/02/cddlm/CDL-
1.0"
   xmlns:cmp="http://www.gridforum.org/cddlm/components/2005/02"
   xmlns:bpc="http://www.gridforum.org/ogsa/BasicPosixComponent/2006/
06"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<cdl:system>
   <Blast cdl:extends="bpc:PosixApplication">
      <cdl:documentation>
         Configuration data for BLAST binary deployment.
      </cdl:documentation>
      <BlastApp cdl:extends="bpc:Application">
         <Binary type="tar">http://blast.app.com/blast.tar</Binary>
         <UserName>csmith</UserName>
         <GroupName>bio</GroupName>
         <Mode>755</Mode>
      </BlastApp>
      <TmpFileSystem cdl:extends="bpc:FileSystem">
        <cdl:documentation>
            Configuration data for temporary file system.
        </cdl:documentation>
        <FileSystemType>temporary</FileSystemType>
        <DiskSpace>
          <LowerBoundedRange>10737418240</LowerBoundedRange>
        </DiskSpace>
      </TmpFileSystem>
      <HomeFileSystem cdl:extends="bpc:FileSystem">
        <cdl:documentation>
            Configuration data for home file system.
        </cdl:documentation>
        <FileSystemType>normal</FileSystemType>
        <MountSource>server.acme.com:/home/csmith</MountSource>
        <MountPoint>/home/csmith</MountPoint>
      </HomeFileSystem>
      <DataBaseFileSystem cdl:extends="bpc:FileSystem">
         <cdl:documentation>
            Configuration data for formatted database file system.
         </cdl:documentation>
         <FileSystemType>normal</FileSystemType>
         <MountSource>server.acme.com:/db/ncbiblast</MountSource>
         <MountPoint>/db/ncbiblast</MountPoint>
      </DataBaseFileSystem>
      <PATH cdl:extends="bpc:EnvironmentVariable"/>
          <!--Proper value of PATH environment variable-->
          <!--will be available after deployment.-->
      <TMPDIR cdl:extends="bpc:EnvironmentVariable"/>
          <!--Proper value of TMPDIR environment variable-->
          <!--will be available after deployment.-->
       <HOSTNAME cdl:extends=”bpc:HostName”>node0.cluster1</HOSTNAME>
          <!--Value of HOSTNAME was filled in by Job manager.-->
          <!--Note this is not according to the CDDLM spec -->
          <!--and is mainly for illustration.-->
   </Blast>
 </cdl:system>
```

```
</cdl:cdl>
```

A.5. CDL document after deployment has finished

After deployment the completely resolved CDL document contains no lazy elements. The values for PATH and TMPDIR have now been added (highlighted in blue).

```
<?xml version="1.0" encoding="UTF-8"?>
<cdl:cdl
   targetNamespace="http://cddlm.org/component-model-example"
   xmlns="http://cddlm.org/component-model-example"
   xmlns:cdl="http://www.gridforum.org/namespaces/2005/02/cddlm/CDL-
1.0"
   xmlns:cmp="http://www.gridforum.org/cddlm/components/2005/02"
   xmlns:bpc="http://www.gridforum.org/ogsa/BasicPosixComponent/2006/
06"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<cdl:system>
   <Blast cdl:extends="bpc:PosixApplication">
      <cdl:documentation>
         Configuration data for BLAST binary deployment.
      </cdl:documentation>
      <BlastApp cdl:extends="bpc:Application">
         <Binary type="tar">http://blast.app.com/blast.tar</Binary>
         <UserName>csmith</UserName>
         <GroupName>bio</GroupName>
         <Mode>755</Mode>
      </BlastApp>
      <TmpFileSystem cdl:extends="bpc:FileSystem">
        <cdl:documentation>
            Configuration data for temporary file system.
        </cdl:documentation>
        <FileSystemType>temporary</FileSystemType>
        <DiskSpace>
          <LowerBoundedRange>10737418240</LowerBoundedRange>
        </DiskSpace>
      </TmpFileSystem>
      <HomeFileSystem cdl:extends="bpc:FileSystem">
        <cdl:documentation>
            Configuration data for home file system.
        </cdl:documentation>
        <FileSystemType>normal</FileSystemType>
        <MountSource>server.acme.com:/home/csmith</MountSource>
        <MountPoint>/home/csmith</MountPoint>
      </HomeFileSystem>
      <DataBaseFileSystem cdl:extends="bpc:FileSystem">
         <cdl:documentation>
            Configuration data for formatted database file system.
         </cdl:documentation>
         <FileSystemType>normal</FileSystemType>
         <MountSource>server.acme.com:/db/ncbiblast</MountSource>
         <MountPoint>/db/ncbiblast</MountPoint>
      </DataBaseFileSystem>
      <PATH cdl:extends="bpc:EnvironmentVariable">
          /usr/bin:/usr/local/bin:/usr/local/bio/bin</PATH>
      <TMPDIR cdl:extends="bpc:EnvironmentVariable">
          /tmp</TMPDIR>
       <HOSTNAME cdl:extends=”bpc:HostName”>
          node0.cluster1</HOSTNAME>
```

```
   </Blast>
 </cdl:system>
</cdl:cdl>
```

A.6. JSDL document submitted to BES

The final, refined, JSDL document submitted to BES for execution. The JSDL document was refined using information extracted from the deployed system CDL. Refinements of the initial document are highlighted (in blue).

```
<?xml version="1.0" encoding="UTF-8"?>
<jsdl:JobDefinition
xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl"
xmlns:jsdl-posix="http://schemas.ggf.org/jsdl/2005/11/jsdl-posix"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <jsdl:JobDescription>
      <jsdl:JobIdentification>
         <jsdl:JobName>Blast1</jsdl:JobName>
         <jsdl:Description>Blast query number 1</jsdl:Description>
         <jsdl:JobProject>BlastProject</jsdl:JobProject>
      </jsdl:JobIdentification>
      <jsdl:Application>
         <jsdl:ApplicationName>BlastN</jsdl:ApplicationName>
         <jsdl:ApplicationVersion>2.2.13</jsdl:ApplicationVersion>
         <jsdl:Description>BlastN performs nucleotide similarity
            searching</jsdl:Description>
         <jsdl-posix:POSIXApplication>
            <jsdl-posix:Executable>/usr/local/bin/blastall
               </jsdl-posix:Executable>
            <jsdl-posix:Argument>-p</jsdl-posix:Argument>
            <jsdl-posix:Argument>blastn</jsdl-posix:Argument>
            <jsdl-posix:Argument>-d</jsdl-posix:Argument>
            <jsdl-posix:Argument>est</jsdl-posix:Argument>
            <jsdl-posix:Argument>-T</jsdl-posix:Argument>
            <jsdl-posix:Argument>T</jsdl-posix:Argument>
            <jsdl-posix:Input filesystemName="HOME">
               sequences1.txt</jsdl-posix:Input>
            <jsdl-posix:Output filesystemName="HOME">
               sequences1.html</jsdl-posix:Output>
            <jsdl-posix:Error filesystemName="HOME">
               sequences1.err</jsdl-posix:Error>
            <jsdl-posix:WorkingDirectory filesystemName="HOME">
               blastqueries</jsdl-posix:WorkingDirectory>
            <jsdl-posix:Environment name="PATH">
               /usr/bin:/usr/local/bin:/usr/local/bio/bin
            </jsdl-posix:Environment>
            <jsdl-posix:Environment name="TMPDIR"
                  filesystemName="TMP"/>
            <jsdl-posix:WallTimeLimit>60</jsdl-posix:WallTimeLimit>
            <jsdl-posix:FileSizeLimit>1073741824
            </jsdl-posix:FileSizeLimit>
            <jsdl-posix:CoreDumpLimit>0</jsdl-posix:CoreDumpLimit>
            <jsdl-posix:DataSegmentLimit>32768
               </jsdl-posix:DataSegmentLimit>
            <jsdl-posix:LockedMemoryLimit>8388608
               </jsdl-posix:LockedMemoryLimit>
            <jsdl-posix:MemoryLimit>67108864</jsdl-posix:MemoryLimit>
            <jsdl-posix:OpenDescriptorsLimit>16
               </jsdl-posix:OpenDescriptorsLimit>
```

```
            <jsdl-posix:PipeSizeLimit>512</jsdl-posix:PipeSizeLimit>
            <jsdl-posix:StackSizeLimit>1048576
               </jsdl-posix:StackSizeLimit>
            <jsdl-posix:CPUTimeLimit>30</jsdl-posix:CPUTimeLimit>
            <jsdl-posix:ProcessCountLimit>8
               </jsdl-posix:ProcessCountLimit>
            <jsdl-posix:VirtualMemoryLimit>134217728
               </jsdl-posix:VirtualMemoryLimit>
            <jsdl-posix:ThreadCountLimit>8
               </jsdl-posix:ThreadCountLimit>
            <jsdl-posix:UserName>csmith</jsdl-posix:UserName>
            <jsdl-posix:GroupName>bio</jsdl-posix:GroupName>
         </jsdl-posix:POSIXApplication>
      </jsdl:Application>
      <jsdl:Resources>
         <jsdl:CandidateHosts>
            <jsdl:HostName>node0.cluster1</jsdl:HostName>
         </jsdl:CandidateHosts>
         <jsdl:FileSystem name="TMP">
            <jsdl:FileSystemType>temporary</jsdl:FileSystemType>
            <jsdl:Description>Well-known 'name' for temporary space
               that does not necessarily persist after the job
               terminates.
            </jsdl:Description>
            <!--Mount point could also be defined in the initial
               JSDL. Added here because this value may be needed to
               construct full paths for other elements.-->
            <jsdl:MountPoint>/tmp</jsdl:MountPoint>
            <jsdl:DiskSpace>
               <jsdl:LowerBoundedRange>10737418240.0
                   </jsdl:LowerBoundedRange>
            </jsdl:DiskSpace>
         </jsdl:FileSystem>
         <jsdl:FileSystem name="HOME">
            <jsdl:FileSystemType>normal</jsdl:FileSystemType>
            <jsdl:Description>Chris's home directory
               </jsdl:Description>
            <!--Mount point could also be defined in the initial JSDL
               Added here because this value may be needed to
               construct full paths for other elements.-->
            <jsdl:MountPoint>/home/csmith</jsdl:MountPoint>
         </jsdl:FileSystem>
         <jsdl:FileSystem name="DB">
            <jsdl:FileSystemType>normal</jsdl:FileSystemType>
            <jsdl:Description>Formatted BLAST database
               </jsdl:Description>
            <jsdl:MountSource>server.acme.com:/db/ncbiblast
               </jsdl:MountSource>
            <jsdl:MountPoint>/db/ncbiblast</jsdl:MountPoint>
         </jsdl:FileSystem>
         <jsdl:ExclusiveExecution>true</jsdl:ExclusiveExecution>
         <jsdl:TotalCPUCount>
            <jsdl:Exact>1.0</jsdl:Exact>
         </jsdl:TotalCPUCount>
      </jsdl:Resources>
      <jsdl:DataStaging>
         <jsdl:FileName>blastqueries/sequences1.txt</jsdl:FileName>
         <jsdl:FilesystemName>HOME</jsdl:FilesystemName>
         <jsdl:CreationFlag>overwrite</jsdl:CreationFlag>
         <jsdl:Source>
            <jsdl:URI>
```

```
            http://csmith.otherhost.com/blastqueries/sequences1.txt
            </jsdl:URI>
         </jsdl:Source>
      </jsdl:DataStaging>
      <jsdl:DataStaging>
         <jsdl:FileName>blastqueries/sequences1.html</jsdl:FileName>
         <jsdl:FilesystemName>HOME</jsdl:FilesystemName>
         <jsdl:CreationFlag>overwrite</jsdl:CreationFlag>
         <jsdl:Target>
            <jsdl:URI>
            http://csmith.otherhost.com/blastqueries/sequences1.html
            </jsdl:URI>
         </jsdl:Target>
      </jsdl:DataStaging>
      <jsdl:DataStaging>
         <jsdl:FileName>blastqueries/sequences1.err</jsdl:FileName>
         <jsdl:FilesystemName>HOME</jsdl:FilesystemName>
         <jsdl:CreationFlag>append</jsdl:CreationFlag>
         <jsdl:Target>
            <jsdl:URI>
            http://csmith.otherhost.com/blastqueries/sequences1.err
            </jsdl:URI>
         </jsdl:Target>
      </jsdl:DataStaging>
   </jsdl:JobDescription>
</jsdl:JobDefinition>
```