GWD-I (draft-ggf-ogsa-ems-scenarios-010)
Open Grid Services Architecture
http://forge.gridforum.org/projects/ogsa-wg

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

9 August 2006

OGSA™ EMS Architecture Scenarios

Version 1.0 (draft 010)

Status of This Document

This document provides information to the community on the scenarios used to refine the OGSA™ EMS architecture. This is an early draft and is still in flux. It does not define any standards or technical recommendations. Distribution is unlimited.

Copyright Notice

Trademarks

Abstract

The OGSA EMS 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. ...

Contents

1 INTRODUCTION **3**
2 FUNDAMENTAL JOB EXECUTION **3**
2.1 DIRECT JOB EXECUTION 3
2.2 INDIRECT JOB EXECUTION 4
2.3 JOB TERMINATION 5
3 SELECTED JOB EXECUTION **7**
4 DEPLOYMENT AND CONFIGURATION **8**
4.1 DEPLOYING AN APPLICATION 8
4.2 DEPLOYING AN APPLICATION USING ACS 11
5 DATA TYPES IN SCENARIOS **12**
5.1 JOB 12
5.2 JOBSTATUS 12
5.3 ACTIVITY 12
5.4 ACTIVITYSTATUS 12
5.5 ACTIVITY EXECUTION CANDIDATE 12
6 SECURITY CONSIDERATIONS **13**
7 GLOSSARY **13**
APPENDIX A. RUNNING A BLAST (BASIC LOCAL ALIGNMENT SEARCH TOOL) JOB **16**
A.1. JSDL DOCUMENT SUBMITTED TO THE JOB MANAGER **16**
A.2. CDL DOCUMENT RETRIEVED BY JM TO PREPARE FOR DEPLOYMENT **17**
A.3. CDL DOCUMENT SUBMITTED BY JM TO DS **19**
A.4. CDL DOCUMENT AFTER DS HAS FINISHED DEPLOYMENT **19**
A.5. JSDL DOCUMENT SUBMITTED TO THE BES INSTANCE **19**

1 Introduction

The OGSA EMS architecture 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 and OSG. Scenarios describing the selection of resources for execution as well as the deployment of the necessary software on the resources are also included. Scenarios build on each other and are presented in order. Later versions of this document will expand on other more advanced EMS scenarios.

In all scenarios a list of required specifications and their status is included.

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 monitors a job execution. In the “indirect” scenario the user “delegates” the management of the job to a job manager.

2.1 Direct Job Execution

Using an established BES Container a client (user agent) contacts a BES container directly and starts an activity described by a simple JSDL document. Progress is monitored directly by the user agent.

This scenario requires the BES and JSDL specifications.

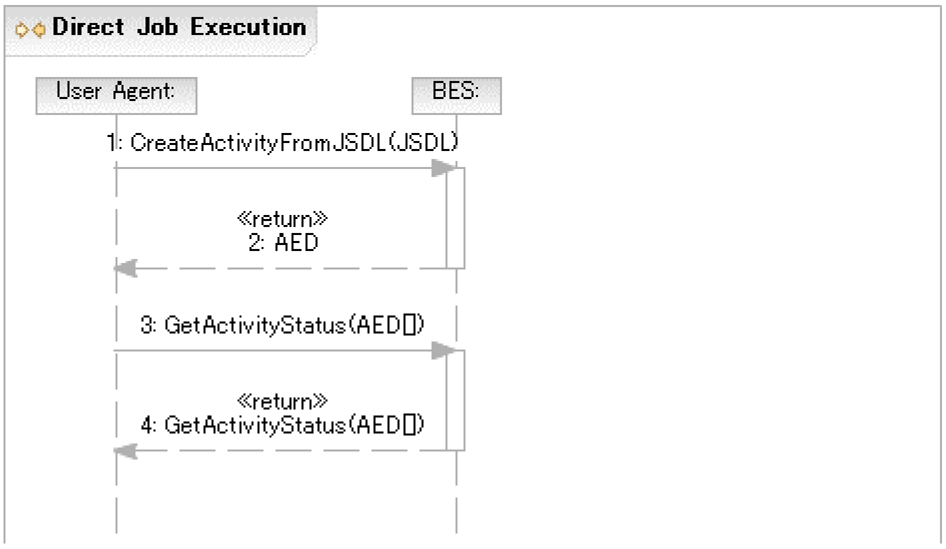
Direct Job Execution
User Agent:
BES:
1: CreateActivityFromJSDL(JSDL)
«return»
2: AED
3: GetActivityStatus(AED[])
«return»
4: GetActivityStatus(AED[])

Figure 1 Direct Job Execution

2.2 Indirect Job Execution

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 established BES container. Once the activity is established it is monitored by the Job Manager.

This scenario requires the BES, JSDL, and Job Manager specifications.

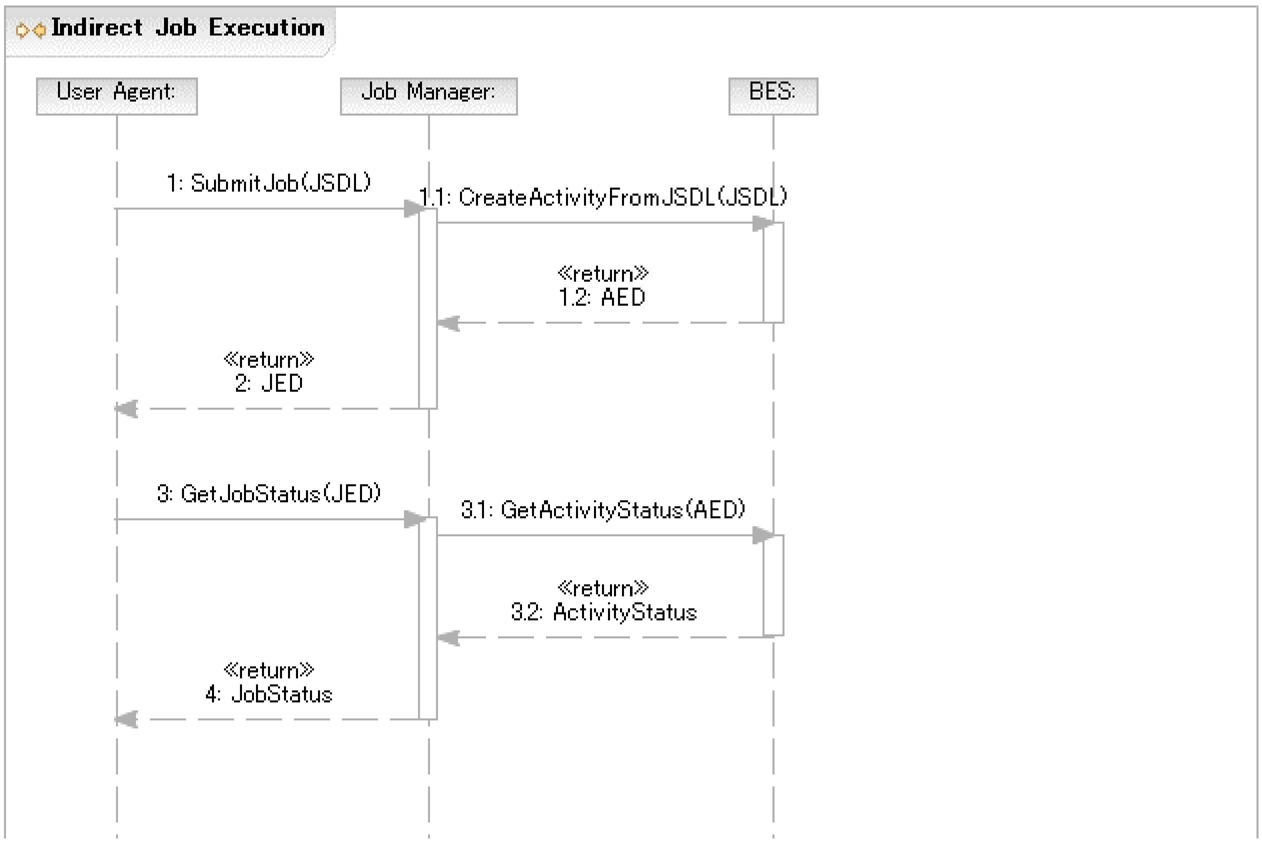

Figure 2 Indirect Job Execution

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

The user agent waits until the activity terminates normally. The User Agent detects this by polling or notification (not shown).

This scenario requires the BES specification.

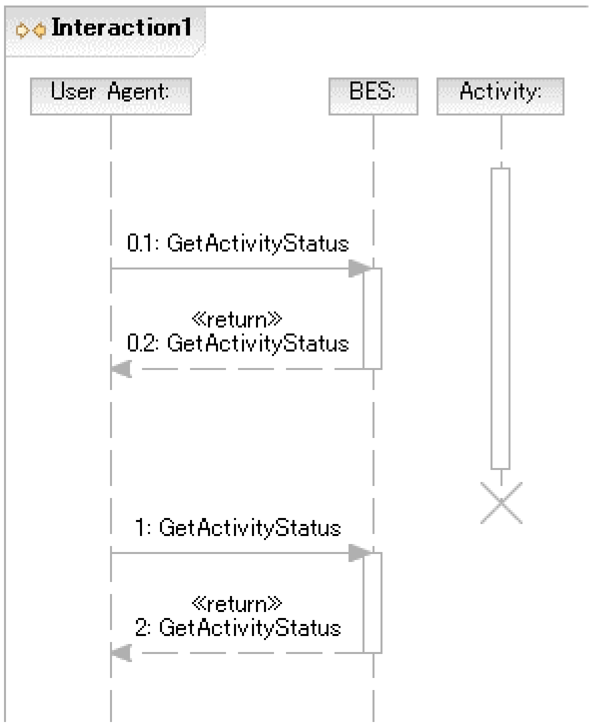

Figure 3 Normal Job Termination

2.3.2 Job killed by User Agent

The user agent requests that the activity is terminated.

This scenario requires the BES specification.

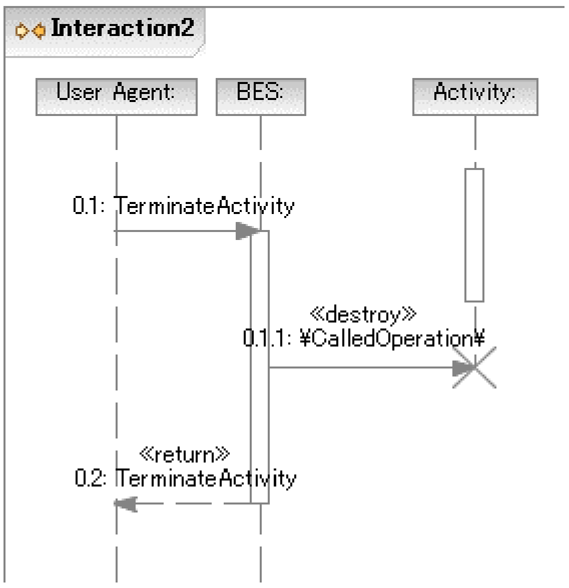

Figure 4 Job killed by User Agent

3 Selected Job Execution

In this scenario an activity (e.g., an invocation of a simple POSIX application (BLAST)) is to be initiated within a BES container. The BES container has not been determined at submission time.

The user agent submits a JSDL document requiring the BLAST application to the Job Manager as in scenario XX. The Job Manager uses the Execution Planning Service (EPS) (part of the Resource Selection Services (RSS)) to select a BES instance. It submits the JSDL document to the EPS and receives a, possibly null, list of activity execution candidates for the job. An activity execution candidate (plan) is made up of at least a JSDL document and a reference to a BES container.

The EPS returns a (possibly refined) JSDL document in the plan 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).

For the purposes of this scenario the first plan returned is considered to be the best choice. The JM starts the requested activity by submitting the (possibly refined) JSDL document to the selected BES instance. The selected BES instance is assumed in this scenario to have BLAST already installed. Later scenarios will describe the necessary deployment steps if the application is not already installed.

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.

Internally, the RSS may use a simple selection algorithm (e.g., number of pre-defined BES instances to round-robin the job request from the job manager to one of these interfaces; or could obtain and select BES instances 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.

This scenario requires the BES, JSDL, JM, and RSS specifications.

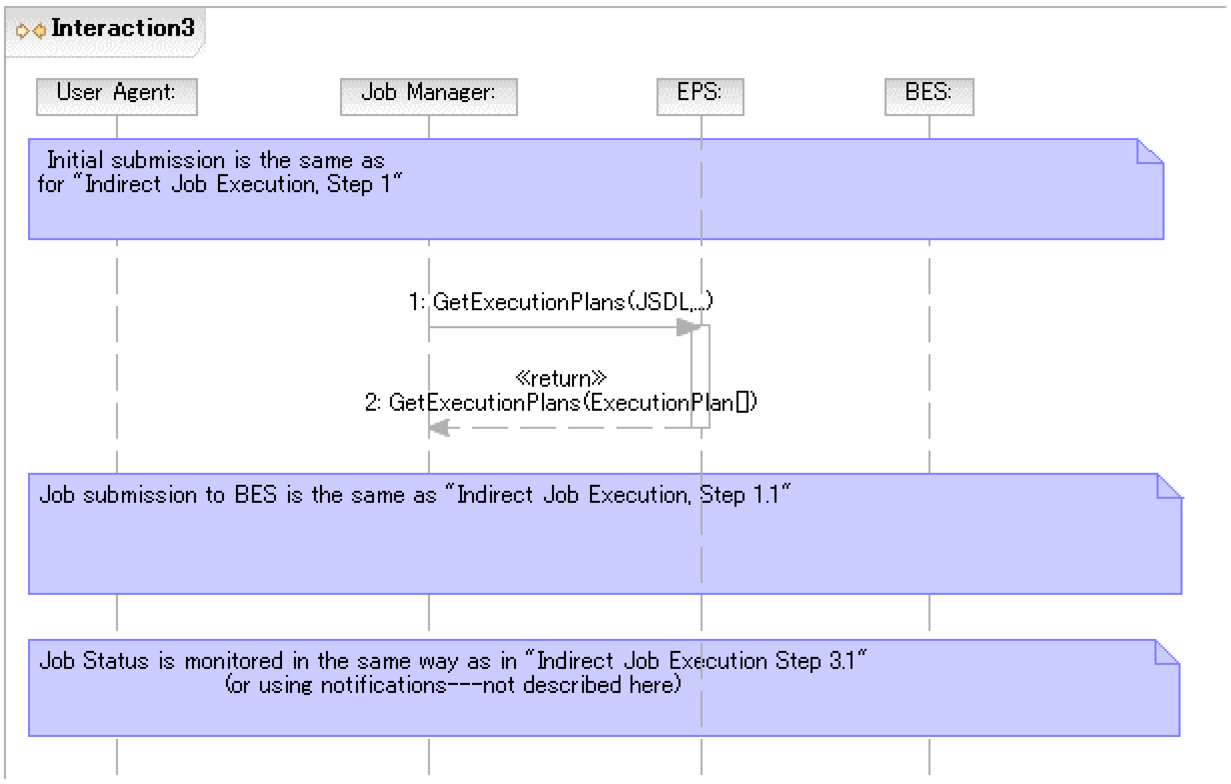
Interaction3
User Agent:
Job Manager:
EPS:
BES:
Initial submission is the same as for "Indirect Job Execution, Step 1"
1: GetExecutionPlans(JSDL,..)
«return»
2: GetExecutionPlans(ExecutionPlan[])
Job submission to BES is the same as "Indirect Job Execution, Step 1.1"
Job Status is monitored in the same way as in "Indirect Job Execution Step 3.1" (or using notifications---not described here)

Figure 5 Selected Job Execution

4 Deployment and Configuration

4.1 Deploying an Application

A simple POSIX application (BLAST) needs to be deployed to support an activity that is to be initiated within a BES container. The BES instance is determined as part of the job submission (but the actual resource, a.k.a. host will be chosen at deployment time) and no BLAST application resides on that platform.

It is assumed that the JM has a suitable BLAST binary, including a BLAST database—possibly created by an earlier preparatory job—as well as a "BLAST Application" CDL document describing the deployment requirements. This scenario does not describe how the JM has obtained these. Scenario XX describes how these can be retrieved from a repository service.

The "BLAST Application" document may have a number of 'lazy elements', e.g., "Hostname" and "PathName." In other words it is a *CDL document template*. Also it can be assumed that the JSDL and CDL documents are created by tooling and are consistent—but not necessarily complete.

Also, for simplicity, it is assumed that the JM knows which deployment service (DS) can be used to deploy to the specified BES instance. In the simplest case the DS may be located on the platform hosting the BES instance, or there is some other well known (pre-configured) relationship. In general the DS 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 DS in a system. It is therefore recommended that a DS advertise (somewhere, in a currently unspecified manner) where it is capable of deploying stuff to, e.g., front end node for a cluster.

The User Agent submits the JSDL document requiring the BLAST application to the JM requesting a specific BES instance. As in previous scenarios it is assumed that the BES instance is specified as an argument to the submission operation and is not included in the JSDL document.

The JM queries the Information Service (IS), or the platform directly, and discovers that the specified BES instance resides on a resource that does not have the BLAST application installed on it.

The JM submits the BLAST CDL document to the DS. The DS 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 to the JM providing a reference to the installed software.

The JM updates the IS entry related to the resource with the location of the BLAST application. This prevents the removal of the resource as ‘something’—the BLAST application—is ‘using’ it.

The JM uses the EPR to the installed software to retrieve information it needs to refine the JSDL document. This is effectively an operation to retrieve relevant chunks of the XML deployment tree (i.e. the instantiated CDL document) back to the JM for insertion into JSDL document.

The, now complete, JSDL document is then sent to the BES instance on that platform.

The JM updates the IS entry related to the BLAST application with the information that the BES Activity now has an ‘interest’ in the application. If another BES Activity uses this BLAST application and performs this registration it will prevent un-deployment of the BLAST application even if the initial activity has completed.

Once the BES Activity completes successfully the JM updates the IS entry related to the BLAST application to remove the “interest” of the BES Activity . If no other activities are registering an interest in the BLAST application on this 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.

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

The User Agent is informed that the job is complete.

This scenario requires the JSDL, Job Manager, CDL, Deployment Service and (unspecified) Information Services specifications.

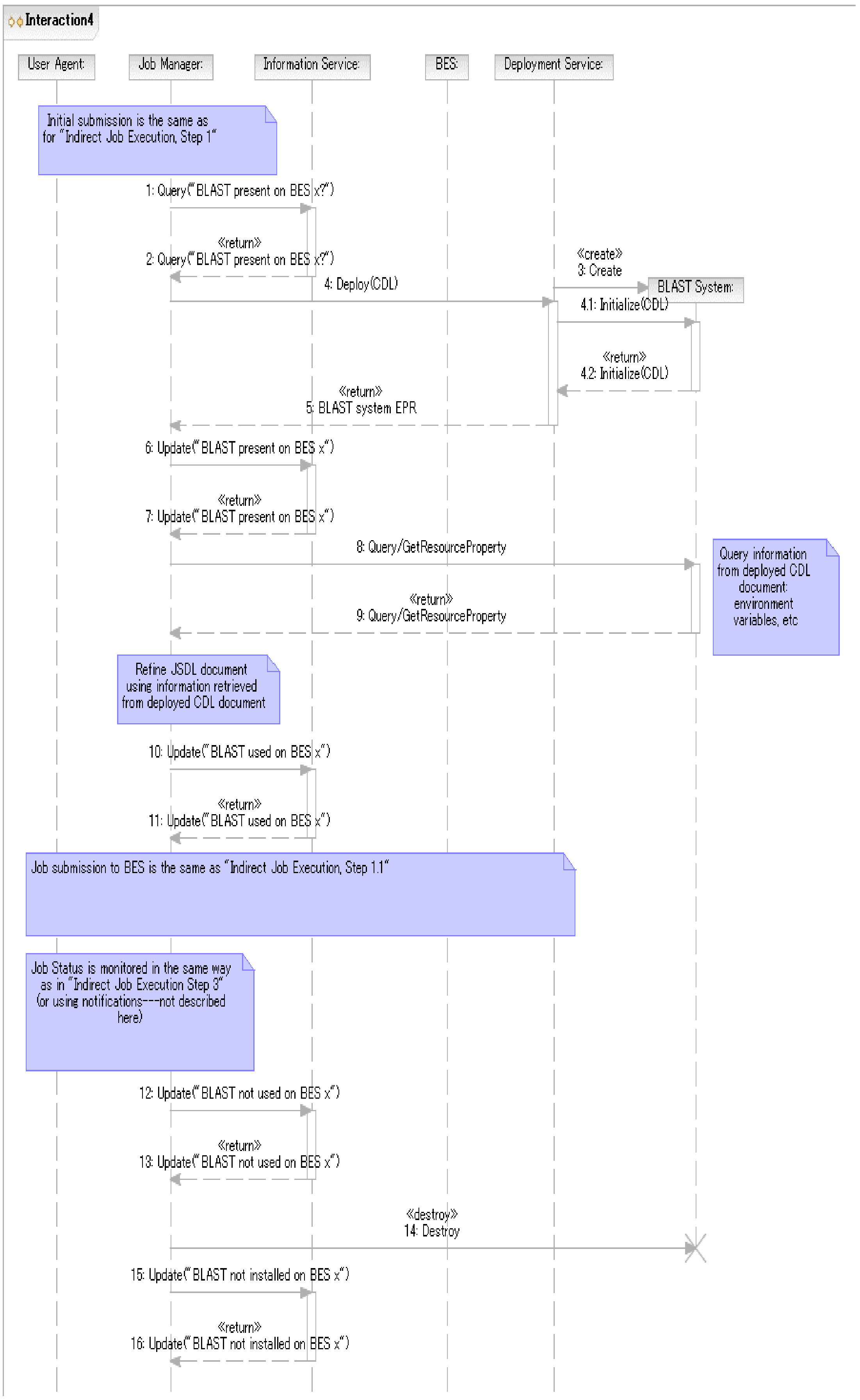
Interaction4
User Agent:
Job Manager:
Information Service:
BES:
Deployment Service:
Initial submission is the same as for "Indirect Job Execution, Step 1"
1: Query("BLAST present on BES x?")
«return»
2: Query("BLAST present on BES x?")
«create»
3: Create
BLAST System:
4: Deploy(CDL)
4.1: Initialize(CDL)
«return»
4.2: Initialize(CDL)
«return»
5: BLAST system EPR
6: Update("BLAST present on BES x")
«return»
7: Update("BLAST present on BES x")
8: Query/GetResourceProperty
Query information from deployed CDL document: environment variables, etc
«return»
9: Query/GetResourceProperty
Refine JSDL document using information retrieved from deployed CDL document
10: Update("BLAST used on BES x")
«return»
11: Update("BLAST used on BES x")
Job submission to BES is the same as "Indirect Job Execution, Step 1.1"
Job Status is monitored in the same way as in "Indirect Job Execution Step 3" (or using notifications---not described here)
12: Update("BLAST not used on BES x")
«return»
13: Update("BLAST not used on BES x")
«destroy»
14: Destroy
15: Update("BLAST not installed on BES x")
«return»
16: Update("BLAST not installed on BES x")

Figure 6 Deploying an Application

4.2 Deploying an Application Using ACS

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. The BES instance has already been determined and no BLAST application resides on that platform.

In contrast to the previous scenario the JM has to discover a BLAST binary and corresponding CDL. It does so by using an ACS Repository. It is assumed that an Application Archive (AA)—addressable by an EPR—that contains all information required to install and configure BLAST is already registered in an Application Contents Services (ACS) Repository in the system. For simplicity, it is assumed that there is a single ACS repository in the system.

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

The JM uses the contents of the JSDL *ApplicationName* element to search for a matching AA in the ACS Repository using the *LookupArchives* operation. The EPR of the AA describing the matching BLAST application is returned. (It is also possible for the AA EPR to be provided as part of the submission, e.g., inside the JSDL document). For simplicity it is assumed that name matching between ApplicationName and AA is sufficient. In scenarios using the EPS this search-and-match may be done by the EPS and include a matching of Archive requirements with the job description.

The JM uses the AA *GetContents* operation to retrieve the CDL document describing the BLAST deployment requirements. (Comments made in previous scenarios about this CDL document also apply here.)

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

There is behind-the-scenes interaction (not shown) between the DS and the ACS Repository. For example, the DS retrieves the binary corresponding to the BLAST application from the ACS Repository.

The remaining steps are the same as in previous scenarios.

This scenario requires the JSDL, Job Manager, ACS, CDL, and (unspecified) Information Services specifications.

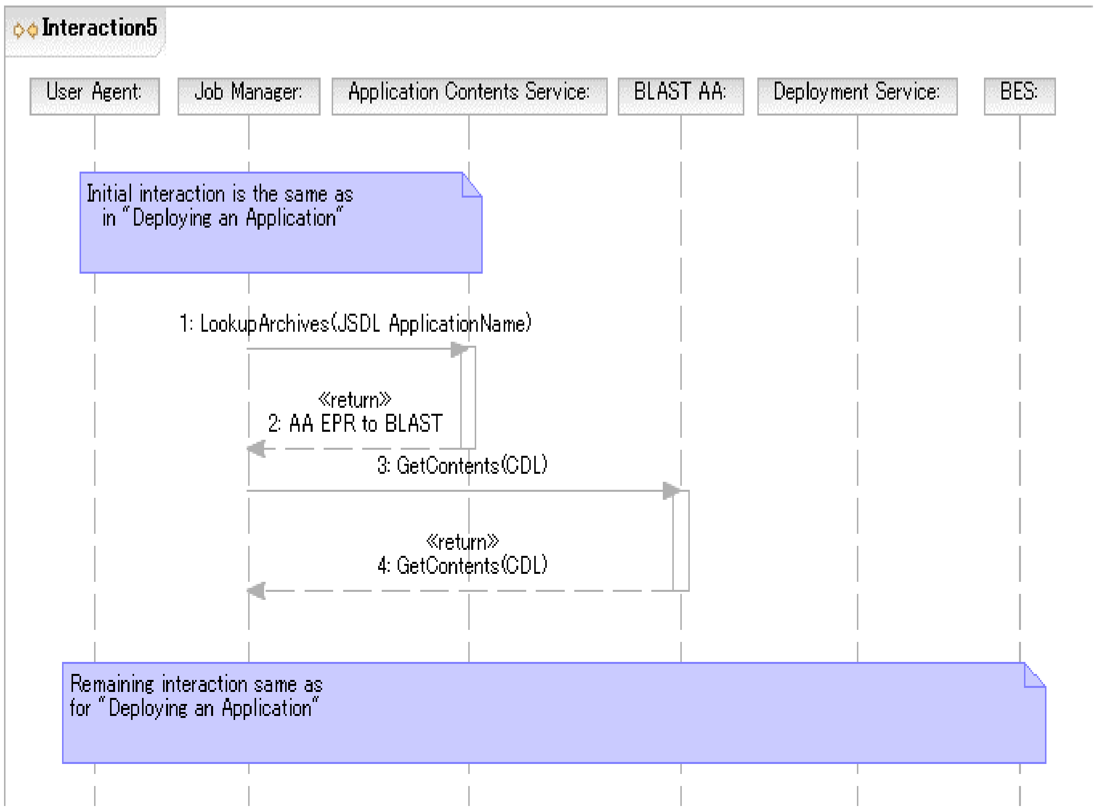
Interaction5
User Agent:
Job Manager:
Application Contents Service:
BLAST AA:
Deployment Service:
BES:
Initial interaction is the same as in "Deploying an Application"
1: LookupArchives(JSDL ApplicationName)
«return» 2: AA EPR to BLAST
3: GetContents(CDL)
«return» 4: GetContents(CDL)
Remaining interaction same as for "Deploying an Application"

Figure 7 Deploying an Application using ACS

5 Data Types in Scenarios

5.1 Job

Job Endpoint Descriptor (JED)

A Job should have some operation to get the current activities.

5.2 JobStatus

This is the status of the Job. This has not been defined yet. But should be defined by the Job Manager group.

5.3 Activity

Activity Endpoint Descriptor (AED)

This is defined in the BES specification.

5.4 ActivityStatus

This is the ActvityStatus and is defined in the BES specification.

5.5 Activity Execution Candidate

This contains a JSDL document, EPR or path of a BES Container, Rank = Number Optional

```
<ActivityExecutionCandidate>
  <JSDL Document/>
  <BES EPR/>
  <Rank/>?
  <CDL Document/>
  <DeploymentService EPR/>
  <xsd:any/>*
</ActivityExecutionCandidate/>+
```

6 Security Considerations

…

Editor Information

…

Contributor

…

Acknowledgements

…

7 Glossary

For the purposes of this document we define:

- Life Cycle: Both provisioning and deployment actions have a life cycle – they are created, used, and eventually destroyed. A mechanism is needed for activities to 'register' their use and interest in a particular activity. Before an activity is destroyed there have to be no outstanding references. [Look at EGA provisioning state model.]
- Provisioning: The instantiation of an environment on a resource (i.e. a deployment activity) that may be used by more than one activity. Generally these 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 manual job submission) or as a result of general response (as defined by a policy) to the state of current resources in the system.
    - For example, "if my available free RHEL4 resources have dropped to 5%, dynamically provision new nodes until the total free nodes have increased to 15%."
- Deployment: The instantiation of an environment on a resource to meet the needs of a specific activity such as a submitted job. This is generally a 'lightweight' action in that may just provide a binary, dataset, etc., onto the resource. 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: download and install the required version for this platform.
- Activity: The smallest part of possibly a larger sequence of activities generated from a single submitted job.

- Job Manager (JM): The entity that accepts a 'job' (defined by a JSDL document) from the user agent. The JM coordinates further invocations and interactions with other elements of the EMS architecture.
- JSDL: Job Submission Description Language (GFD.56)
- Deployment Service (DS): The entity that accepts a 'deployment' action (defined by a CDL document) to instantiate an environment on the resource to support a BES job. The DS provides the bootstrapping for all deployment/provisioning actions. (GFD.69)
- CDL: Configuration Description Language
- CDL document template: A CDL document that still contains unresolved type references, e.g., lazy elements. (Note that this is not a CDDLM defined term. It is used for convenience.)
- Basic Execution Service (BES): The entity that accepts a JSDL document and instantiates an Activity. BES does not do provisioning/deployment.

References

BES

JSDL

RSS

Deployment API

CDL

…

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

A.1. JSDL Document Submitted to the Job Manager

According to the scenario above the JSDL document MUST contain at a minimum:

1. The application name

Highlighted text indicates portions that may need refinement or have linkage to CDL.

```
<?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"
  xsi:schemaLocation="http://schemas.ggf.org/jsdl/2005/11/jsdl
jsdl.xsd
http://schemas.ggf.org/jsdl/2005/11/jsdl-posix jsdl-posix.xsd">
  <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>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 BY JM 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: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. CDL Document retrieved by JM to prepare for deployment

The initial CDL template retrieved by the JM.

Add text that it is based on “BasicPosixComponent” not shown here.

```
<?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/C
DL-1.0"

xmlns:cmp="http://www.gridforum.org/cddlm/components/2005/02"

xmlns:bpc="http://www.gridforum.org/ogsa/BasicPosixComponent/2
006/06"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
 <cdl:system>
   <Blast cdl:extends="bpc:PosixApplication">
     <cdl:documenation>
         Configuration data for BLAST binariy deployment.
     </cdl:documenation>
     <BlastApp cdl:extends="bpc:Application">
       <Binary type="tar"> http://blast.app.com/blast.tar
</Binary>
       <UserName> csmith </UserName>
     <GroupName> hpc </GroupName>
     <Mode> 755 </Mode>
     </BlastApp>
     <TmpFileSystem cdl:extends="bpc:FileSystem">
       <cdl:documenation>
           Configuration data for temporary file system.
       </cdl:documenation>
       <FileSystemType>temporary</FileSystemType>
       <DiskSpace>
        <LowerBoundedRange>10737418240</LowerBoundedRange>
       </DiskSpace>
     </TmpFileSystem>
     <HomeFileSystem cdl:extends="bpc:FileSystem">
       <cdl:documenation>
           Configuration data for home file system.
       </cdl:documenation>
       <FileSystemType>normal</FileSystemType>
       <MountSource>server.acme.com:/home/csmith</MountSource>
       <MountPoint>/home/csmith</MountPoint>
     </HomeFileSystem>
     <DataBaseFileSystem cdl:extends="bpc:FileSystem">
        <cdl:documenation>
           Configuration data for formatted database file
system.
        </cdl:documenation>
        <FileSystemType>normal</FileSystemType>

<MountSource>server.acme.com:/db/ncbiblast</MountSource>
        <MountPoint>/db/ncbiblast</MountPoint>
     </DataBaseFileSystem>
     <PATH cdl:extends="bpc:EnvironmentVariable"/>
        <cdl:documenation>
           Proper value of PATH environment variable will be
available after
           deployment.
        </cdl:documenation>
     <TMPDIR cdl:extends="bpc:EnvironmentVariable"/>
        <cdl:documenation>
           Proper value of TMPDIR environment variable will be
```

```
available after
           deployment.
        </cdl:documenation>
   </Blast>
 </cdl:system>
</cdl:cdl>
```

A.3. CDL Document submitted by JM to DS

The CDL template after the JM has (possibly) added some values, e.g., hostnames.

Assume for now that this is identical to the one in A.2

A.4. CDL document after DS has finished deployment

The completely resolved CDL document (no lazy elements).

This one is needed.

A.5. JSDL Document submitted to the BES Instance

The final, refined, JSDL document submitted to BES for execution. Refinements of the initial document are highlighted.

```
<?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"
  xsi:schemaLocation="http://schemas.ggf.org/jsdl/2005/11/jsdl
jsdl.xsd
http://schemas.ggf.org/jsdl/2005/11/jsdl-posix jsdl-posix.xsd">
  <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>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: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>
```

ISSUES

1. Terminology needs fixing, e.g., job/activity.
2. Make a new sub-section for the Assumptions of each scenario
3. Refer to UML sequence steps in the explanation
4. Get latest CDL and include in Appendix
5. Cross-check JSDL and CDL examples for consistency
6. How to refer to JSDL/CDL documents in the scenarios
7. References section
8. …