GWD-R(draft-ggf-ogsa-bes-spec-003)
Open Grid Services Architecture
http://forge.gridforum.org/projects/ogsa-bes-wg

Authors:
A. Grimshaw, U.Virginia
S. Newhouse, U. Southampton
Darren Pulsipher

May 26, 2005

OGSA Basic Execution Service
Version 1.0

Status of This Memo

This document is in draft stage and should be considered as in flux. Distribution is unlimited.

Abstract

The OGSA V1.0 document section 3.4, pages 17-25, describes an EMS (Execution Management Services) architecture consisting of a number of different services. This document describes one of these services – the "service container" – which is the focus of the Basic Execution Service (BES) specification developed by this working group. BES models execution of services in "containers" that may be implemented in a variety of ways, e.g., by a single Unix or Windows host, by a queuing system, by a hosting environment such as .Net or J2EE, or by more specialized execution containers yet to be invented. BES defines a set of port-types as well as resource properties (attributes) for the simplest – most basic container.

This document defines the scope and motivation for this work followed by an abstract definition of the BES interface. This interface is then rendered using the OGSA profiles into normative text in Appendix A.

This document obviously reflects the decisions taking by the working group as to the essential capabilities of a 'Basic Execution Service'. Where relevant we have also captured the decisions as to the capabilities that are not part of a BES and the reasons behind the decision.

Contents

1. Introduction

The purpose of this document is to describe a web service interface to initiate, monitor & control activity on computational resources. This web service interface enables the creation, destruction and status determination of 'activities' (e.g. jobs, services, resources, …) within a container – an abstract representation of computational capability. Such a container may be a single machine, a cluster managed through a Distributed Resource Manager (DRM) such as Load Leveler, Sun Grid Engine, Portable Batch System, etc. or an interface into a web service hosting environment. Operational differences between container implementations are not expected to be reflected in the service implementation.

Considerable effort has been undertaken within the OGSA-WG EMS (Execution Management System) design team to define the different services and their interactions. The current high-level architecture for the execution of 'legacy' binary applications is encapsulated in the following diagram.

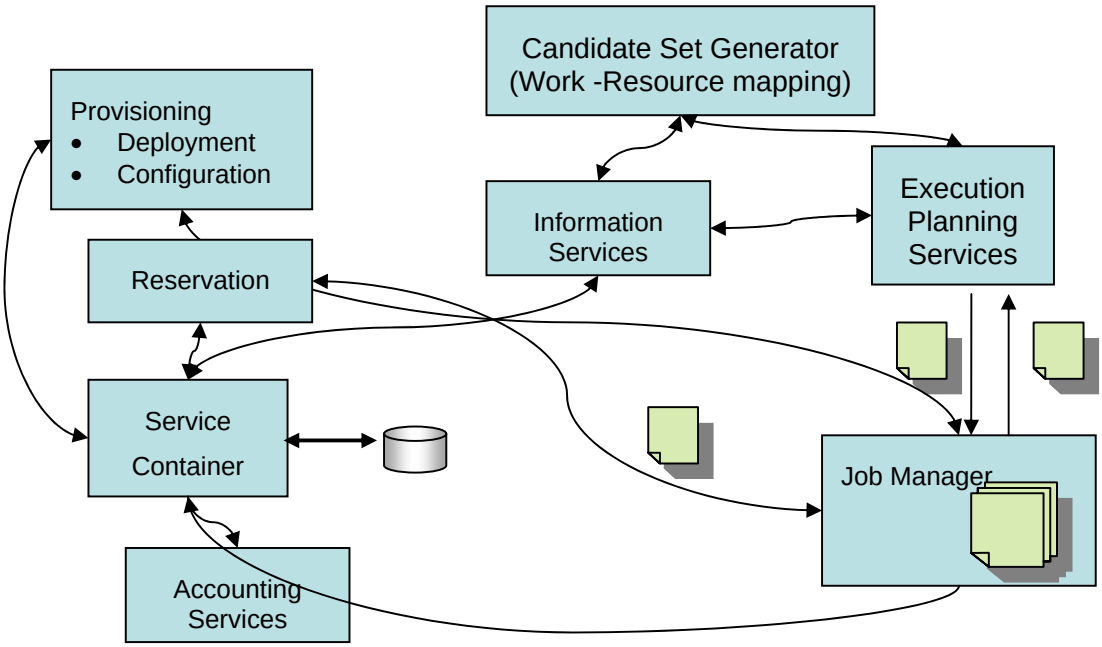


Figure 1: This figure is taken from the OGSA 1.0 informational document and has been modified slightly to reflect the I/O model of JSDL. Specifically, the "data container" has been replaced by a "disk" symbol. There is an assumption in JSDL that the execution context of a "job" as storage that can be accessed with the same path everywhere in the execution context, e.g., there is a shared file system.

The purpose of this service interface is to tackle the issues surrounding the 'Service Container' and to form part of the OGSA Basic Execution Profile. Other services described within the above architecture are considered to be out of scope of this activity.

2. Assumptions

The word 'activity' will be used frequently from this point in the document. Within the context of this document an activity could be the execution of a legacy binary or the initiation of a web service within a container. From the perspective of an external observer this is a self contained operation. If it can be further decomposed, such decomposition does not visible through the service interface.

By the time we reach the point of invoking the Basic Execution Service (BES) we assume that the following issues have been resolved:

- The placement decision (where the 'activity' is going to run) has already been determined by some unspecified means. This may have used some a set of 'resource attributes' associated with the BES instance in a registry or used to annotate the BES instance itself. [At GGF 14 it was decided that specifying these attributes was out of scope of the BES specification at this time. It was noted that work with CIM and the GLUE schema going on elsewhere within the Grid community would be relevant to BES in the future.]
- Naming. An essential pre-requisite to many distributed computing activities is a 'naming' scheme that provides a globally unique identifier in time and space. BES expects to use the WS-Name schema being defined within the OGSA-Naming-WG. It is not our intention to duplicate work in this area.
- Job Submission Description Language (JSDL). BES will use JSDL. There are several assumptions, some explicit, and some implicit. One of the most relevant here is that JSDL assumes a stage-in/stage-out model and associated "local" file system capability in which data is copied to/from the locus of execution. The presumption is that there is a local file system that is visible everywhere within the locus of execution – whether it is a single host or a large cluster.
- The 'activity' that is going to be initiated within the container has already been determined elsewhere and is fully specified in the JSDL document. Therefore we assume that the JSDL document presented to the service is 'concrete' in nature. This does not preclude the JSDL document being initially specified using, for example, logical file names but that these logical file locations have been replaced by real file locations by the time the document reaches the BES.

3. Service Interface

The interface port-types for the OGSA-BES service container are described in this section. The port-types are described using a combination of English and IDL. In Appendix A the port-types are rendered using the definitions found in the OGSA WSRF Base Profile document, i.e., conformant with the OGSA WSRF Base Profile.

3.1 CreateActivityFromJSDL

This operation is used to initiate a new activity within the BES container as specified in a JSDL document.

Input(s):

- JSDLDocument jobDescriptionDocument Accepts an XML document conforming to JSDL 1.0 describing a single activity that is to be started within the BES container. [At GGF 14 we discussed supporting multiple job

Output(s):

- WS-Name activityIdentifier On success a WS-Name identifying the requested activity is returned from the service.

Fault(s):

- NotAcceptingNewActivities: The BES is not accepting any new activities at this state.
- BadlyFormedJSDLDocumentFault: The XML contained within the JSDL document is badly formed, i.e. it is not good XML. The relevant parsing error is returned in the body of the fault.

- UnsupportedFeatureFault: The well formed JSDL 1.0 document contains an operation or a non-standard extension that is not supported by the BES. The feature that is not supported by the BES is returned in the body of the fault.
- JobSpecificationFault: Despite a well formed JSDL document that is understood by the BES the specified activity cannot be undertaken, e.g. mis-specified executable location or ftp file transfer. The unsupported JSDL elements are returned with the body of the fault.

3.2 GetActivityStatus

The activity specified within the JSDL document may contain many different operations – staging in, job execution, staging out, etc. To capture the state of this potentially complex activity within the job we define an operation that allows the state of several of these activities to be retrieved from the BES.

Input(s):

- WS-Name[] activityIdentifiers Passes in a vector of WS-Names (generated from the createActivityFromJSDL operation) which relate to the activities from which we require state information.

Output(s):

- ActivityStatus[] activityStatus An XML document containing a vector of ActivityStatus elements.

Fault(s):

- UnknownActivityIdentifierFault: Before the BES attempts to return information on the specified activities it will attempt to verify that these activities exist. If any activity specified by the WS-Name does not exist within the container then this fault is thrown. Its contents will be the WS-Names for which no information is known within the BES. If any WS-Name does not exist then no output is returned from the operation.

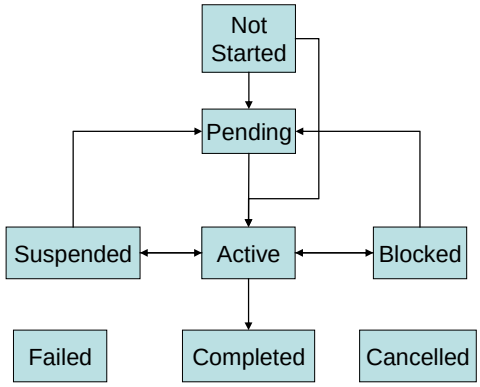
The JSDL document comprises three sets of possible actions:

- Staging-in: Each staging-in action will have its own state.
- Execution: The single execution action will have its own state.
- Staging-out: Each staging-out action will have its own state.

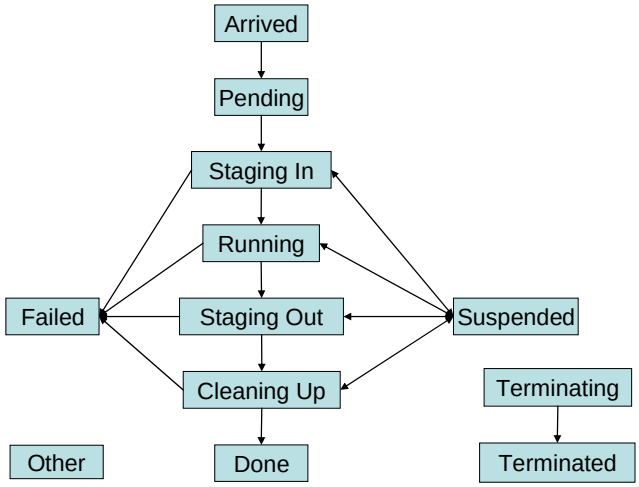
The state of each action is defined as follows, and the transition between these states in the following state diagram:

- NotStarted: The action has been recognised by the BES but has not progressed any further in terms of activity within the container.
- Pending: The action has entered into a state that is external to the service but still internal to the container. For instance, the executable may have entered into a queue within a batch scheduling system or the file transfer may be queued into some service.
- Active: The action is underway.
- Blocked: The action cannot proceed any further at this time due to the unavailability of some resource external to the BES.
- Suspended: The action will not progress any further at this time at the instigation of the BES.
- Completed: The action is complete.

- Failed: This state can be reached from any state except Completed. This state is triggered by the BES due to un-recoverable external action or event.
- Cancelled: This state can be reached from any state except Completed. This state is triggered by the BES due to other operations on the BES.



The proceeding text describes the state of each action initiated by the BES from the JSDL document. It is also important to gain an overall view of the state of a particular activity. However this 'meta-state' is dependent on the state of the actions specified within the BES.



These states are defined as follows:

- Arrived: The JSDL document specified by the 'createActivityFromJSDL' operation has been parsed and accepted as by the BES. The 'staging-in' actions are all in the 'not-started' state.
- Pending: One or more of the 'staging-in' actions have moved beyond the 'not-started' state into the 'queued' state.
- Staging-In/Staging-Out: One or more of the 'staging-in/staging-out' actions are in the 'active', 'blocked' or 'suspended' states. It is possible that one or more of these actions may still be in the 'not-started' or 'queued' states.
- Running: All of the 'staging-in' actions have reached the 'completed' state and the execution action has moved beyond the 'not-started' state.
- Cleaning-Up: All of the 'staging-out' actions have reached the 'completed' state.

- Done: All of the ‘staging-out’ actions have reached the ‘completed’ state.
- Failed: If any of the actions enter the ‘failed’ state then the whole activity enters the ‘failed’ state.
- Suspended: If any of the actions enter the ‘suspended’ state the whole activity enters the ‘suspended’ state.
- Terminating: This is a state that the activity enters into following a call to the ‘terminateActivites’ operation. Following such a call the BES will attempt to cancel any activity that has not already ‘failed’ or ‘completed’.
- Terminated: All of the actions have entered either a ‘failed’, ‘completed’ or ‘cancelled’ state.
- Other: An activity within the BES can transition to the ‘other’ state from any other state. Effectively this is a catchall state equivalent to say an unknown state.

The state of each specified action within an activity, and the overall state of the activity are returned to the requester within an ActivityStatus element with the following structure:

<ActivityStatus>

<ActivityIdentifier>WS-Name</ActivityIdentifier>

<OverallStatus state=”(enumerated state)” />

<StageInStatus id=”file string” state=”(enumerated state)” />*

<ExecutableStatus state=”(enumerated state)” />

<StageOutStatus id=”file string” state=”(enumerated state)” />*

</ActivityStatus>*

A * indicates zero or more elements of this type.

At GGF 14 several other items where identified that could be returned within the ActivityStatus element:

- <ResourceAllocation>: Lists the resources that have been allocated to this activity. The allocated resources may differ from those requested by the activity within the JSDL document.
- <UsageRecord>: Use the UsageRecord schema to record the resource that has been consumed so far by the activity.

All implementations of the BES may transition through these states, but not all of these states will always be transitioned. As an action changes state and as the overall activity changes state an event will be dispatched if it has been subscribed to.

ISSUE: Compare these states against CIM state model.

ISSUE: Do we want to be able to apply a ‘suspend’ signal to an activity within the BES. If we do various file transfer activities may go into suspend.

ISSUE: Authorisation – who is able to find out about an activity state

3.3 TerminateActivities

This operation initiates the termination of a set of activities within the BES.

Input(s):

- WS-Name[] activityIdentifiers Vector of WS-Names that are to be terminated.

Output(s):

- None

Fault(s):

- None

The BES attempts to terminate each activity specified in the list. As a consequence of this operation the specified activity moves from its current (presumably non-terminating state) to the 'terminating' state. If termination is successful then the activity enters into a 'terminated' state. Invoking this operation on a 'terminated' activity has no further effect. How long the activity remains in the 'terminated' state before the WS-Name no longer returns a reference to the activity is not defined. The overall success of this operation (i.e. to move the activity into a 'terminated' state') must be determined through other operations or subscribing to any generated events.

ISSUE: Should the operation fail silently if the WS-Name does not exist within the BES OR should it throw a fault.

3.4 StopAcceptingNewActivities

This operation moves the BES into a state where it stops accepting new activities.

Input(s):

- None.

Output(s):

- None.

Fault(s):

- None.

Instead, this should be somewhere else, and we may want a few paragraphs at the beginning saying that management is out-of-scope, but we believe there are some required aspects.

3.5 StartAcceptingNewActivities

This operation moves the BES into a state where it starts accepting new activities.

Input(s):

- None.

Output(s):

- None.

Fault(s):

- None.

3.6 ShutdownContainer

This operation terminates the BES container. The impact on the activities taking place within the container is undefined.

Input(s):

- None.

Output(s):

- None.

Fault(s):

- None.

3.7 GetSubmittedJobDescriptions

This operation returns the submitted JSDL document that is associated with the specified WS-Name. terminates the BES container. The impact on the activities taking place within the container is undefined.

Input(s):

- WS-Name[] activityIdentifiers

Output(s):

- JSDLDocument[] documents.

Fault(s):

ISSUE: The WS-Name needs to be stored somewhere in the returned JSDL documentation.

4. Exposing Container Activity

The BES does not mandate that the activity started within the container provide a web service interface for management or control purposes.

An interface to an activity corresponding to a legacy POSIX-like binary application was proposed at GGF14. An outline of such an interface is presented below:

```
interface Job {
    getJobId         (out string        jobId);
    getJobState      (out JobState      state);
    getJobInfo       (out JobInfo       info);
    getJobDefinition (out JobDefinition jobDef);
    getJobExitStatus (out JobExitStatus exitStatus);
    suspend          (); // A request that may not succeed
    resume           (); // A request that may not succeed
    hold             ();
    release          ();
    restart          (); // Throw fault if not supported
    checkpoint       (); // Throw fault if not supported
    migrate          (in JobDefinition  jobDef);
    terminate        (); // Stop the activity & initiate terminate
action
    signal           (in int            signum); // Common in Windows &
UNIX systems
}
```

ISSUE: Is this within scope of the BES-WG. Does it need to be specified here?

5. Security Considerations

Security considerations are significant in execution management, both in terms of access control (authorization) to the various services, as well as identity mapping issues, e.g., run this activity as “Fred”. Authorization and authentication is outside of the scope of this document and is dependent on the ongoing activity within the OGSA Authorisation Working Group.

6. Authors Information

Andrew Grimshaw

Mark Morgan

Chris Smith

Darren Pulsipher

Steven Newhouse

Add your name here …

7. Contributors

We gratefully acknowledge the contributions made to this document by

8. Acknowledgments

We are grateful to numerous colleagues for discussions on the topics covered in this document, and to the people who provided comments on the public drafts. Thanks in particular to (in alphabetical order, with apologies to anybody we have missed) ….

9. Intellectual Property Statement

10. Full Copyright Notice

11. References

OGSA WSRF Basic Profile definition

OGSA Authorization

WS-Naming

RNS

12. Appendix A – OGSA WSRF Base Profile Rendering

12.1 CreateActivityFromJSDL

<WSDL Goes here >

12.2 getActivityStatus

<WSDL Goes here >

12.3 terminateActivity

<WSDL Goes here >

12.4 StopAcceptingNewActivities

<WSDL Goes here >

12.5 StartAcceptingNewActivities

<WSDL Goes here >

12.6 ShutdownContainer

<WSDL Goes here >