BES/JSDL 1.1 Extensions

Status of this Memo

This memo provides information to the Grid community regarding a set of profiles and extensions on the Job Submission Description Language and the OGSA™ Basic Execution Services documents. These extensions are strictly upward compatible, i.e., they are entirely consistent with the existing specifications and are elaborations of XSD:any or new porttypes. Distribution is unlimited.

Copyright Notice

Copyright © Open Grid Forum (2014). All Rights Reserved.

Abstract

The Production Grid Interoperability Working Group identified a number of execution management use cases and requirements in GFD.180. A number of ways to meet these requirements have been extensively discussed. They fall into two categories: 1) define a new set of specifications from scratch to meet the requirements, and 2), profile and minimally extend existing specifications to meet the requirements.

The second approach, combining and refining existing production use specifications, has been embraced by the existing OGSA BES and JSDL communities. The approach combines, extends, and profiles five existing specifications to meet the PGI requirements: WS Addressing EndPoint References, OGSA Basic Execution Services (OGSA\_BES, or BES) [GFD.108], RNS 1.1 OGSA-WSRF Basic Profile 1.0 [GFD.172], WS-Iterator 1.0 [GFD.188], and OGSA-ByteIO WSRF Basic Profile 1.0 [GFD.98].

The BES/JSDL extensions described in this document are a part of the second approach. Along with the BES Basic Directory Profile (BDP) and the Activity Endpoint Profile (AEP) the BES/JSDL extensions meet the PGI requirements. Profiled JSDL extensions and their corresponding additions to BES factory attributes include: incorporation of GLUE2 [] compute resource properties, arbitrary name/value pair matching parameters, additional file system types, and richer file staging options. The profiled BES extensions include a substate model with a set of “Held” substates and pre-and-post-processing substates, a “ResumeActivities” porttype, and clarified JSDL mechanism to subscribe to notifications.

Contents

Table of Contents

[BES/JSDL 1.1 Extensions 1](#_Toc377723941)

[Abstract 1](#_Toc377723942)

[1 Introduction 4](#_Toc377723943)

[2 Notational Conventions 4](#_Toc377723944)

[3 BES State Model Changes 5](#_Toc377723945)

[4 JSDL Profiled Elements 7](#_Toc377723946)

[4.1 Resource Elements 7](#_Toc377723947)

[4.1.1 OperatingSystem\_t 7](#_Toc377723948)

[4.1.2 CPU Type 9](#_Toc377723949)

[4.1.3 Coprocessor 9](#_Toc377723950)

[4.1.4 FileSystem Extensions 10](#_Toc377723951)

[4.1.5 Network Info 10](#_Toc377723952)

[4.1.6 Node Internet Access 11](#_Toc377723953)

[4.1.7 RemoteSessionAccess 11](#_Toc377723954)

[4.1.8 SlotRequirement 11](#_Toc377723955)

[4.1.9 SlotsPerHost 12](#_Toc377723956)

[4.1.10 ExclusiveExecution 12](#_Toc377723957)

[4.1.11 QueueName 12](#_Toc377723958)

[4.1.12 Walltime 12](#_Toc377723959)

[4.1.13 IndividualCPUTime 12](#_Toc377723960)

[4.1.14 ParallelEnvironment 12](#_Toc377723961)

[4.1.15 Discuss 13](#_Toc377723962)

[4.1.16 Run Time Environment – e.g. modules 13](#_Toc377723963)

[4.1.17 Matching Options 13](#_Toc377723964)

[4.2 JobProcessing 15](#_Toc377723965)

[4.2.1 Hold 15](#_Toc377723966)

[4.2.2 Pre and Post Processing 15](#_Toc377723967)

[4.2.3 Pre-cache path for GFFS FIX THIS FOR GENERAL WIDE AREA FILE SYSTEMS 16](#_Toc377723968)

[4.3 File Staging 16](#_Toc377723969)

[4.3.1 Creation Enumeration – extension 16](#_Toc377723970)

[4.3.2 Recursion 16](#_Toc377723971)

[4.3.3 Wild Cards 17](#_Toc377723972)

[5 Extensions and Profiles 17](#_Toc377723973)

[5.1 Profiled BES Factory Attributes 17](#_Toc377723974)

[Supported file staging protocols – in factory attributes 17](#_Toc377723975)

[6 BES porttype extensions 18](#_Toc377723976)

[6.1.1 SupportBESMatching1.1 18](#_Toc377723977)

[6.1.2 SupportBES1States1.1 18](#_Toc377723978)

[6.1.3 SupportBESGLUE2.0 18](#_Toc377723979)

[6.2 PortTypes 18](#_Toc377723980)

[TODO: Add optional WS-Notification subscription in create activity, 18](#_Toc377723981)

[6.2.1 ResumeActivities 19](#_Toc377723982)

[7 Compliance Targets 19](#_Toc377723983)

[8 Security Considerations 19](#_Toc377723984)

[9 Author Information 19](#_Toc377723985)

[10 Contributors 20](#_Toc377723986)

[11 Acknowledgements 20](#_Toc377723987)

[Full Copyright Notice 20](#_Toc377723988)

[Intellectual Property Statement 20](#_Toc377723989)

[Normative References 21](#_Toc377723990)

# Introduction

The Production Grid Interoperability Working Group identified a number of execution management use cases and requirements in GFD.180. The group discussed a number of ways to meet these requirements. These approaches fall into two categories: 1) define an entirely new set of specificationsto meet the requirements, or 2) profile and minimally extend existing specifications to meet the requirements.

The BDP is a part of the second approach; it profiles and extends existing specifications to meet the requirements. It combines, extends, and profiles five existing specifications to meet the PGI requirements: **WS Addressing EndPoint References**;**OGSA Basic Execution Services** (OGSA\_BES, or BES) [GFD.108];**RNS 1.1 OGSA-WSRF Basic Profile 1.0** [GFD.172];**WS-Iterator 1.0** [GFD.188]; and **OGSA-ByteIO WSRF Basic Profile 1.0** [GFD.98].

Profiled JSDL extensions and their corresponding additions to BES factory attributes include: 1) incorporation of GLUE2 [] compute resource properties;2) arbitrary name/value pair matching parameters;3) additional file system types; and 4) richer file staging options. The profiled BES extensions include a substate model with a set of “Held” substates and pre-and-post-processing substates, a ResumeActivities porttype, and clarified JSDL mechanism to subscribe to notifications.

The document is organized as follows.Section 2 discusses notational conventions; section 3 describes the BES state model profile; section 4 details the JSDL extensions; section 5 details the BES factory attributes extensions and the single additional ResumeActivitiesporttype.

# Notational Conventions

The key words “MUST,” “MUST NOT,” “REQUIRED,” “SHALL,” “SHALL NOT,” “SHOULD,” “SHOULD NOT,” “RECOMMENDED,” “MAY,” and “OPTIONAL” are to be interpreted as described in RFC-2119 [RFC 2119].

The document refers to a “BES/JSL Directory Profile compliant system” as a “Compliant system”.

This specification uses namespace prefixes throughout; they are listed in Table 1. Note that the choice of any namespace prefix is arbitrary and not semantically significant.

Table : Prefixes and namespaces used in this specification.

|  |  |
| --- | --- |
| Prefix | Namespace |
| Xsd | <http://www.w3.org/2001/XMLSchema> |
| Wsa | <http://www.w3.org/2005/03/addressing> |
| Rns | <http://schemas.ogf.org/rns/2009/12/rns> |
| Byteio | <http://schemas.ggf.org/byteio/2005/10/byte-io> |
| sbyteio | <http://schemas.ggf.org/byteio/2005/10/streamable-access> |
| rbyteio | <http://schemas.ggf.org/byteio/2005/10/random-access> |
| Bdp | <http://schemas.ogf.org/bdp/2012/03/bdp> |
| Jsdl | http://schemas.ggf.org/jsdl/2005/11/jsdl |
| bes1.1 | http://schemas.ggf.org/BES/2014/bes1.1 |

One of the goals of this document is to bring JSDL and BES into conformance with GLUE2 [GFD.147] nomenclature and syntax, in particular GFD.209, *GLUE v. 2.0 – Reference Realization to XML Schema*. Rather than copying (and possibly miscopying) GLUE2 types and enumerations, we copy them as graphics directly from GFD.209. We include the page number in parenthesis after the GFDnumber so that it is clear exactly what is being copied. Below is an example for the OSFamily\_t, GFD.209(61), i.e. defined on page 61 of GFD.209.



# BES State Model Changes

The BES state model from the original specification contains five states. The BES specification allows the profiling of substates within the five states as long as the state transitions at the top level are not modified.

Pending

Running

Finished

Terminated

Failed

TerminateActivity request

System error/failure event

Successful termination of activity

**Figure 1**. Basic state model from GFD 108.

During a series of meetings within the PGI working group, additional requirements were determined, in particular the ability to suspend an activity before and after execution in order to allow external client-driven interaction with the session directory of the job. This was codified in the European Middleware Initiative interfaces [GFD.210]. In addition, there has been significant feedback from developers on the need to explicitly model pre-and-post-processing steps. This profile defines substates for the Pending and Running states. The profiled substates for Pending and Running are shown in Figures 2 and 3 below.

The “Held” substate of both Pending and Running is a special substate in that it corresponds to a set of substates, e.g., Pending:Created-Held, Pending:Meta-scheduling-Held, etc., where there is a \*-Held substate for each Pending and Running substate shown*except*Running:Queued. The substates of both Pending and Running are listed in order in Table 2. Not all substates are necessarily entered.

Held substates are only entered if the activity JSDL contains a “Hold” element as specified in Figure 2 of this document. Upon entering a Held state, processing of the activity by a compliant BES will stop until a ResumeActivites (Section 6.2.1) is received by the BES or the ResumeActivityporttype is called on the corresponding Activity Endpoint as defined in the AEP. Upon receipt of a resume, the compliant BES will move the activity to the next corresponding state, e.g., Pending:Staging-in from Pending:Scheduled-Held, and resume processing the activity.

Finally, the initial BES state model from GFD.108 has no transition from Pending to Failed. Experience has shown that such a transition is needed. Therefore, compliant implementations MAY transition from Pending to Failed. Note that such a transition is consistent with the BES state model. An external observer may never see the Running state of an activity: the activity might appear to transition directly from Pending to Failed.

Figure 2. BES/JSLD 1.1 substates of Pending and Running. Note that there are many “Held” substates. Only one is being show for each. In Pending it illustrates a Held substate between Scheduled and Staging-in.

Table . Substates for Pending and Running in order that they are observed. Not all substates will be observed.

|  |  |
| --- | --- |
| Pending | Running |
| Created | Executing |
| Created-Held | Executing-Held |
| Meta-scheduling | Post-processing |
| Meta-scheduling-Held | Post-processing-Held |
| Scheduled | Staging-out |
| Scheduled-Held | Staging-out-Held |
| Staging-in |  |
| Staging-in-Held |  |
| Pre-processing |  |
| Pre-processing-Held |  |
| Queued |  |

# JSDL Profiled Elements

JSDL is a very successful standard. When first written,the community envisioned that profiles and updates would be developed over time. In particular, it was anticipated that a congruence with GLUE would be established when the latter stabilized. BES/JSDL standardizes and documents changes and extensions that have been developed over time by the BES and JSDL communities. Many of these changes are borrowed *in toto* from the EMI Execution Service [GFD.210].

The JSDL extensions fall into four broad categories: 1) additional Resources elements or extensions to existing Resource elements; 2) extensions to Data Staging elements; 3) a notion of execution environments for application execution, and 4) a new JobDescription element called JobProcessing.

## Resource Elements

### OperatingSystem\_t

JSDL currently has a non-GLUE2-compliant OperatingSystem element that can be included in the Resourceselement. The OperatingSystem element is a complex type that includes OperatingSystemType, OperatingSystemVersion, and Description. Both OperatingSystemType and OperatingSystemVersion have their own respective, non GLUE12-compliant enumerations.

To bring JSDL and BES into GLUE2 compliance we profile a new Resources element OperatingSystem\_t. This optional complex element specifies the operating system required for the user job. Its type is SoftwareRequirement. Multiplicity is zero or more, where multiple values are interpreted as giving alternatives (i.e. OR semantics are implied). There is no default value of this element. In case of OperatingSystem the Family element of the Software structure embedded in the SoftwareRequirement is open enumeration with values of the GLUE2 OSFamily\_t type, GFD.209(61):



We add two additional OS families to the open enumeration.

<enumeration value=”android”/>

<enumeration value=”iOS”/>

|  |  |
| --- | --- |
| Normative JSDL | Meaning |
| Android | Family of Android systems |
| iOS | Family of Apple iOS based systems |

The corresponding GLUE2 OSName\_t, GFD.209(61)



We add two additional OS name to the open enumeration.

<enumeration value=”windows7”/>

<enumeration value=”windows8”/>

|  |  |
| --- | --- |
| Normative JSDL | Meaning |
| windows7, windows 8 | As of this writing, the most recent Windows. |

#### Pseudo Schema

<Operatingsystem\_t>

<OSFamily\_t>?

<OSName\_t>?

</Operatingsystem\_t>

### CPU Type

JSDL currently has a non GLUE2-compliant ProcessorArchitecture element with a corresponding enumeration. To bring JSDL and BES into GLUE2 compliance we profile a new optional Resources element Platform\_t that specifies the platform architecture required for the user job. Multiplicity is zero or one. There is no default value of this element. Its is an open enumeration with a values of the GLUE2 Platform\_t type, GFD.209(61):



We add one additional CPU name to the open enumeration.

<enumeration value=”arm”/>

|  |  |
| --- | --- |
| Normative JSDL | Meaning |
| ARM | ARM family – most prevalent processor family in the world. |

### Coprocessor

Neither JSDL 1.0 nor GLUE2 addresses the co-processor issue. We define a CoProcessor element that is an open enumeration that specifies the type of coprocessing unit that is available. Multiplicity is zero or more with an implied OR.

<CoProcessor>

<NVidia\_t>?

<IntelPhi\_t>?

<FPGA\_t>?

<MicronFSA\_t>?

</CoProcessor>

For each subelement we define another open enumeration,

<simpleType name=”Nvidia\_t>

<union memberTypes=”string”>

<simpleType>

<restriction base=”string”>

<enumeration value=”c870”/>

<enumeration value=”d870”/>

<enumeration value=”s870”/>

<enumeration value=”c1060”/>

<enumeration value=”s1070”/>

<enumeration value=”c870”/>

<enumeration value=”c870”/>

…

</restriction>

</simpleType>

</union>

</simpleType>

<simpleType name=”IntelPhi\_t>

<union memberTypes=”string”>

<simpleType>

<restriction base=”string”>

<enumeration value=”pickavalue”/>

…

</restriction>

</simpleType>

</union>

</simpleType>

<simpleType name=”FPGA\_t>

<union memberTypes=”string”>

<simpleType>

<restriction base=”string”>

<enumeration value=”pickavalue”/>

….

</restriction>

</simpleType>

</union>

</simpleType>

<simpleType name=”MicronFSA\_t>

<union memberTypes=”string”>

<simpleType>

<restriction base=”string”>

<enumeration value=”pickavalue”/>

….

</restriction>

</simpleType>

</union>

</simpleType>

|  |  |
| --- | --- |
| Normative JSDL | Meaning |
| NVIDEA | Compute Unified Device Architecture, a parallel computing architecture developed by NVIDIA – open enumeration of versions |
| Phi | Intel Phi – open enumeration of versions |
| FPGA | Field programmable gate array - – open enumeration of versions |
| MicronFSA\_t |  |

### FileSystem Extensions

We define four additional well-known file system types extending jsdl:FileSystemTypeEnumeration.

|  |  |
| --- | --- |
| Normative JSDL Name | Definition |
| GFFS | Defines that a GFFS file system should be mounted. |
| iRODS | iRODs file system |
| HDFS |  |
| S3 | S3 Block Storage File System |
| WebDAV | A WebDAV file system |

Further we define an optional extension unique-id for the SCRATCH file system. This attribute is used to give the BES container a unique name with which to batch the files that are downloaded as part of a job. Because many different users may choose to stage in files of similar names (a.out, input.dat, etc.), into SCRATCH we have to keep those files from conflicting with each other. Unfortunately, one cannot always assume that each user has a unique user id or local home directory. For example:

<FileSystem name="SCRATCH" bes1.1:unique-id="unique-scratch-id">

<FileSystemType>spool</FileSystemType>

</FileSystem>

### Network Info

JSDL currently has a non GLUE2-compliant IndividualNetworkBandwidth element that is defined as a jsdl:RangeValue\_Type. To bring JSDL and BES into GLUE2 compliance we profile a new optional Resources element NetworkInfo\_tthat specifies the network type required for the user job. Multiplicity is zero or one. There is no default value for this element;it is an open enumeration with the values of the GLUE2 NetworkInfo\_t type:

|  |  |
| --- | --- |
| Normative JSDL | Meaning |
| 100megabitethernet | Network based on 100 MBit/s Ethernet technology |
| Gigabitethernet | Network based on 1 GBit/s Ethernet technology |
| 10gigabitethernet | Network based on 10 GBit/s Ethernet technology |
| Infiniband | Network based on Infiniband technology |
| Myrinet | Network based Myrinet technology |
| Tightly-coupled | The interconnection network is low latency. |
| Loosely | The interconnection network is relatively high latency compared to tightly coupled. |

Note that we have based this set on the GFD.210 extension of GLUE2.

### Node Internet Access

JSDL has no notion of whether compute nodes can access the internet. We define a new Resource element NodeInternetAcess of NodeInternetAccessEnumeration. The optional element defines the required connectivity of the execution node. Multiplicity is zero or one. If it is not defined, then network connection is not required for the user job. NodeInternetAccessEnumeration contains the following values

|  |  |
| --- | --- |
| Normative JSDL | Meaning |
| Inbound | Inbound network is required for the user job, i.e., the node has a pubic IP address AND a connection can be established from outside the site to a compute node. |
| Outbound | Outbound network connectivity is required for the user job, i.e., programs running on compute nodes can send packets to external public IP addresses. |
| Inoutbound | Both directions are required for the user job |

### RemoteSessionAccess

JSDL has no notion of whether clients can access the working directory of running activities. We define a new Resource element RemoteSessionAccess of xsd:boolean. The optional element specifies whether the client needs to access the activity working directory (sometimes called the session directory). Multiplicity is zero or one.

<RemoteSessionAccess>

<xsd:boolean>

</RemoteSessionAccess>

If RemoteSessionAccessis not defined,the user is not interested in access to the session directory remotely (default is false).

### SlotRequirement

GLUE2 introduces a notion of Slots in addition to the usual notion of sockets, CPUs, cores per CPU, and cores per node. The term Slot is used to denote a logical CPU visible to and allocable by the resource management system. They differentiate this from a core insofar as one may not be guaranteed exclusive access to the entire core[[1]](#footnote-2).

<SlotRequirement>

<xsd:UInt3>

</SlotRequirement>

This optional element specifies the requested count of slots and its distribution for multi-slot jobs. Multiplicity is zero or one.

### SlotsPerHost

This optional integer element specifies the number of slots to be allocated on each single host (node).

<SlotsPerHost>

<xsd:UInt3>

</SlotsPerHost>

### ExclusiveExecution

This optional boolean element specifies whether a host (node) should be allocated for exclusive use by the user job.

<ExclusiveExecution>

<xsd:boolean>

</ExclusiveExecution>

Each site has a default value for this, which should be advertised through GLUE2.

### QueueName

This optional string element defines the name of the preferred queue. Multiplicity is zero or one. There is no default value of this element.

<QueueName>

<xsd:String>

</QueueName>

### Walltime

This optional element is the wall clock time requested for the user job, from the start of the first process until the completion of the last process. Multiplicity is zero or one.

<Walltime>

<xsd:UInt3>

</Walltime>

### IndividualCPUTime

This optional element specifies the number of CPU seconds requested for each slot of the user job. There is no default value of this element. Multiplicity is zero or one.

<IndividualCPUTime>

<xsd:UInt3>

</IndividualCPUTime>

### ParallelEnvironment

The parallel environment is used to specify the execution environment for parallel jobs. Multiplicity is zero or one. If a ParallelEnvironment element is present, the execution service MUST create the correct invocation for the requested parallel environment. The execution service MAY also add environment variables and path settings as appropriate. The parallel environments available at an execution service MUST be advertised through the GLUE2 description of the execution service using ApplicationEnvironment element.

This optional element defines the type of multi-slot application. There is no default value of this element. It is string valued, with the following initial set of values taken from the SPMD extension 61 for the JSDL.

### Discuss with JP, some concern glue does not meet our needs

The following seem redundant to Grimshaw. Should we include them?

9.3.5.16.3 ProcessesPerHost

This optional integer element specifies the number of instances of the executable that the consuming system MUST start on each allocated host. Multiplicity is zero or one. Default value is 1. An optional flag useSlotsPerHost allows to indicate that the value of SlotsPerHost should be used.

9.3.5.16.4 ThreadsPerProcesses

This optional integer element specifies the number of threads per process (i.e., per instance of the executable). There is no default value of this element. Multiplicity is zero or one. An optional flag useSlotsPerHost allows to indicate that the value of SlotsPerHost should be used.

9.3.5.16.2 Version

The optional version of the parallel environment.

### Matching Options

The ability to match jobs to resources is fundamental to Grids. This requirement goes both ways: jobs must be able to specify what they need in a BES and the BES must be able to specify attributes that the job must possess. This can be accomplished with *Matching Options*. A Matching Option is an arbitrary name/value pair that a job can use to indicate a property that it requires or supports (required properties have a value of **requires:**value and supported properties have a value of **supports:**value). It is a sub-element of JSDL:Resources. Compliant BESs likewise can advertise the Matching Options that it supports or requires and they are matched to these JSDL matching parameters in the obvious way. For this JSDL extension, the element is defined as

<bes1.1:Matching name=” *xsd:string* “ value=” *xsd:string*“/>**\***

and it occurs 0 or more times inside of JSDL:Resources.

These matching options are different from the existing mechanisms built into the JSDL specification in that they allow for a user to make requests for scheduling or matching based on completely arbitrary scheduling parameters that are agreed on outside of the JSDL specification. In theory, this could also have been done by adding new XML elements into the resources section of the JSDL, but by doing it this way, the code can be written to do the simple matching without knowing what the possible space of parameter types and values is ahead of time.

Activities that **require** a value get matched against BESes that **support** or **require** it, and jobs that **support** a value are allowed to run on BESes that **require** it. The full table of matching is given below.

|  |  |  |  |
| --- | --- | --- | --- |
|  | BES Supports  Parameter | BES Requires  Parameter | BES Does Not  Advertize Parameter |
| Activity Supports  Parameter | Allowed | Allowed | Allowed |
| Activity Requires  Parameter | Allowed | Allowed | Not Allowed |
| Activity JSDL Does Not Mention  Parameter | Allowed | Not Allowed | Allowed |

Example JSDL Pseudo Schema

<bes1.1:Matchingvalue="Some Value" name="requires:Matching Parameter 1"/>>

<bes1.1:Matching value="Another Value" name="requires:Matching Parameter 1"/>>

Profiled Matching options

|  |  |
| --- | --- |
| value | Meaning |
| GFFSAvailable | A value of “true” means that the GFFS can be mounted on nodes where the job will be executed. A value of “false” means it cannot. If it is not specified it is assume that the GFFS cannot be mounted on the execution nodes. |
| SupportedFileSystems | The “value” indicates which file system types are supported, e.g., SCRATCH, GFFS, HOME, TMP. |
| MPIVersions | The “value” indicates the name and type of MPI supported. |
| InstalledApplications | The “value” indicates the an installed applicatiom .e.g. BLAST1.4 |
| ActivityEndpointSupport | The client requires EPRs returned from CreateActivity to be AEP compliant. |
| HoldActivitiesSupported | The client requires the BES to support activity hold states. |
| NodeInternetAccess | The “value” indicates the type of internet access required. |
| Allocation | The “value” indicates the allocation name, e.g., Kraken, indicates a Kraken allocation. |
| ModulesSupported | The “value” indicates the a supported module, e.g., Blast |
| PrePostProcessing | This can be “true” or “false”. |
| StagingProtocls | Indicates a staging protocol supported, e.g., http, GFFS, GridFTP |
| GridPipesSupported | Indicates that GFFS Grid pipes to standard input, output, and error are supported. |
| GenesisIIAvailable | Indicates whetherGenesisII been installed and isaccessible from execution nodes |
|  |  |

## JobProcessing

BES/JSDL 1.1 introduces a new sub element of JSDL:JobDescription, JDSL:JobProcessing. Sub-elements of JobProcessing indicate additional steps that the BES MUST or MAY perform while processing the activity. There are four sub-elements currently defined:

* Hold
* PreCache
* PreProcesses
* PostProcess
* Notification

Note that the order that these appear in the application section is irrelevant.

### Hold

The Hold sub-element of the JobProcessing element indicates when the BES should stop processing the activity and wait for a ResumeActivities as described in §3. The multiplicity is zero or more.

<HoldEnumeration>\*

HoldEnumeration

|  |  |
| --- | --- |
| Pending:Created | Running:Executing |
| Pending:Meta-scheduling | Running:Post-processing |
| Pending:Scheduled | Running:Staging-out |
| Pending:Staging-in |  |
| Pending:Pre-processing |  |

The semantics dictate that the activity move into the appropriate Held state AFTER exiting the specified state. Upon resumption, the activity will move into the subsequent state in the state model.

### Pre and Post Processing

Often it is desirable to run a script before or after executing an application. For a sequential application this does not represent a problem; a sequential combination of the preprocessing steps, the application itself, and the post-processing steps can be combined into a single script. However, when the application is to be run in a parallel environment, e.g., an MPI application, one cannot simply run the preprocessing on every node. Instead,the preprocessing steps should run on the head or login node whilethe parallel machine should execute the job (usually through a batch scheduler) and then execute the post processing steps.

To support this use case, we profile the JSDL extension to add an optional element to the JobProcessing element. There may be AT MOST ONE of pre and post process.what-to-do-if-it-fails may be either Fail or Continue

<PreExecution>

path=”some path” arguments=”arguments”failure-action=”what-to-do-if-it-fails”

</PreExecution>?

If the pre-processing step fails and the JSDL specifies Fail, then the activity will move to Failed state. If the JSDL specifies Continue on failure, then the activity moves to the Pending:Queued state.

For post-execution

<PostExecution>

path=”some path” arguments=”arguments” failure-action=”what-to-do-if-it-fails”

</PostExecution> ?

If the post-processing step fails and the JSDL specifies Fail, then the activity will move to Failed state. If the JSDL specifies Continue on failure, then the activity moves to the Running:staging-out state.

Example JSDL Pseudo Schema for pre-execution.

<PreExecution>

path=”./setup.sh” arguments=”--all” failure-action=”Fail”

</PreExecution>?

### Pre-cache path for GFFS FIX THIS FOR GENERAL WIDE AREA FILE SYSTEMS

If the GFFS is available and used by applications, the user might want to alert the BES to tell the local GFFS environment to begin pre-fetching and caching portions of the GFFS name space. This is done using the optional GFFSPreCache directive. The BES MAY at its own discretion interact with the local GFFS implementation to begin pre-fetching or ignore the directive. The semantics of execution are unchanged by pre-fetching.

<GFFSPreCache>

<! The GFFS path to pre-cache />

<xsd:String>

</GFFSPreCache>

## File Staging

### Creation Enumeration – extension

JSDL defines *JSDL:CreationFlagEnumeration* flags of overwrite, dontOverwrite, and append. We define two additional members of the enumeration,

<setExecutable>

<xsd:boolean>

</setExecutable>

And

<setUMask>

<xsd:String>

</setUMask>

### Recursion

<source:URI>

<JSDL:RecursiveCopy>

Semantics would be IF and only IF a) the URI protocol supports recursive operations AND the middleware supports recursive operations, copy recursively. Else fault.

### Wild Cards

1. Wild card will be expressed in the URI element.
2. Wild card expansion is the responsibility of the middleware (the BES), if the protocol supports wild cards, and the middle wants to use it, that is ok. Conversely, if the protocol does not support wildcards, the middleware MUST or fault.
3. File & directory structure will be the same.
4. At a minimum the middleware should support “\*” and “?” What escape character if any?

The middleware is free to perform optimizations as possible, e.g., use tar and ssh instead of scp

# Extensions and Profiles

## Profiled BES Factory Attributes

For each of the JSDL things above we need the corresponding thing in the BES factory attributes and/or the Resource Properties.

GLUE 2 computeelements

See also section 8.1 of EMI ES, has elements like number of jobs ….

## Supported file staging protocols – in factory attributes

# BES porttype extensions

This section describes the compliance requirements.

### SupportBESMatching1.1

This Metadata entry is OPTIONAL. If present in the Endpoint Reference for the BES it whether the endpoint is compliant with the BES this profile Profile. The entry’s type is xsd:boolean, and it has a cardinality of exactly 1. A value of “true” indicates compliance, while a value of “false” or the absence of the entry indicates non-compliance.

<wsa:EndpointReference>

...

<wsa:Metadata>

<bdp:whatever>

/xsd:boolean

</bdp:SupportsBESDirectory>

</wsa:Metadata>

</wsa:EndpointReference>

### SupportBES1States1.1

### SupportBESGLUE2.0

## PortTypes

### TODO: Add optional WS-Notification subscription in create activity,

This section is applicable to implementations that implement the WSRF-BP.

Users may want to subscribe to state change and resource exceeded events.

WSRF-BP, section 6.1 mandates the use of WSNotification’sNotificationProducer port-type, which enables a consumer to subscribe to notification topics at any time via the Subscribe message exchange. The topics that are available from the port-type are not mandated, however, so the following topics are OPTIONAL.

Comment2: How do we subscribe within the bes framework? May be through the createactivity method similar to the way genesis does?

**Topic: State Change**

Notification returns EPR of the activity, old state and new state, and some (optional) description

Topic:

/aep:ActivityStateChanged

/aep:ActivityStateChanged/aep:ReachedFinalState

Returns:

<aep:BESActivityStateChangedContents>

<aep:ActivityEPR>/wsa:EndpointReferenceType</aep:ActivityEPR>

<aep:OldState>bes:ActivityStatus</aep:OldState>

<aep:NewState>bes:ActivityStatus</aep:NewState>

<aep:Description>xsd:string</aep:Description> ?

</aep:BESActivityStateChangedContents>

The notification response message is returned to WSNotification’s Consumer service for which the endpoint address is given at the time of subscription. The Consumer service could be residing at the user facing client side entity, meta-scheduler, or portal application.

### ResumeActivities

This operation requests that a specified set of activities be released from either Pending:HOLD-IN or Running:HOLD-Out state.The BES transitions the specified activities from either the Pending:HOLD-IN state to Pending:Stage-In or the Running:HOLD-OUT state to Running:Stage-Out. If an activity cannot be released immediately, the eventual success of this operation (i.e., to move the activity into the Running:*Hold-OUT* state) must be determined through other operations (e.g.,GetActivityState) or by subscribing to any generated events, if a BES supports subscription.

If a request is successful, then each specifiedactivitywill eventually enter thecorresponding stagingstate. Invoking this operation on an activity not in a HELD state has no effect.

#### Input(s)

* **EPR[] activities**: A vector of zero or more EPRs identifying the activities that are to be resumed.

#### Output(s)

* **ResumeActivityResponseType[] Response**: A vector detailing the responses of the BES to the Resume requests. The *Resume* element is a boolean value indicating whether the BES successfully (true) resumed the activity or not (false). If true is returned, then the associated activity has now exited the HELD state. If false is returned then the activity MAY eventually transition out of the *HELD* state. If an activity specified in the input cannot be located or cannot be resumed for some reason, or is not in a HELD state, then the ResumeResponse MUST contain a SOAP-1.1 fault element instead of a Resumed element.

<ResumeActivityResponse>

<ActivityIdentifier>EPR</ActivityIdentifier>

<Resumed>xsd:boolean</Resumed> ?

|

<SOAP-1.1:fault> ... </SOAP-1.1:fault> ?

</ResumeActivityResponse> \*

#### Fault(s)

* **InvalidRequestMessageFault:** An element in the request message is not recognized. The elements that are not recognized are described in the body of the fault. This does not mean that the element itself is in error, but rather that it specifies a syntactically correct value which does not in fact make sense.

# Compliance Targets

# Security Considerations

Access control is out of scope.

# Author Information

Daniel Dougherty

University of Virginia

Andrew Grimshaw (editor)  
University of Virginia

ShahbazMemnon  
Forschungszentrum Juelich (FZJ)

Bernd Schuller  
Forschungszentrum Juelich (FZJ)

# Contributors

We gratefully acknowledge the contributions made to this specification by [insert names].

# Acknowledgements

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've missed)[insert names].

We would like to thank the people who took the time to read and comment on earlier drafts. Their comments were valuable in helping us improve thereadability and accuracy of this document.

Full Copyright Notice

Copyright © Open Grid Forum (2014). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the OGF or other organizations, except as needed for the 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 assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE OPEN GRID FORUM DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Intellectual Property Statement

The OGF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the OGF Secretariat.

The OGF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this recommendation. Please address the information to the OGF Executive Director (see contact information at OGF website).

Normative References

[RFC 2119]Bradner, S. Key words for use in RFCs to Indicate Requirement Levels. Internet Engineering Task Force, RFC 2119, March 1997.Available at <http://www.ietf.org/rfc/rfc2119.txt>

[JSDL10] Available at <http://www.ggf.org/documents/GFD.136.pdf>

OGF

WS Addressing EndPoint References

OGSA Basic Execution Services (OGSA\_BES, or BES) [GFD.108]

RNS 1.1 OGSA-WSRF Basic Profile 1.0 [GFD.172]

WS-Iterator 1.0 [GFD.188]

OGSA-ByteIO WSRF Basic Profile 1.0 [GFD.98]

GFD.210 European Middleware Initiative Execution Service Version 2.0 M. Riedel, A. Konstantinov

1. GFD.147 pp. 22-23. [↑](#footnote-ref-2)