GWD-R (draft-ogf-jsdl-hpcp)
JSDL-WG

Marty Humphrey, UVA
Chris Smith, Platform Computing
Marvin Theimer, Microsoft
Glenn Wasson, UVA
July 14, 2006
Updated: October 2, 2006

JSDL HPC Profile Application Extension, Version 1.0

Status of This Document

This document provides information to the Grid community regarding the specification of the HPC Profile Application. Distribution is unlimited.

Abstract

This document specifies the semantics and structure of the HPC Profile Application. The HPC Profile Application is an extension to JSDL 1.0 [JSDL10] that is used to describe an executable running as an operating system process. The document includes the normative XML Schema for the HPC Profile Application, along with examples of documents based on this schema.

Contents

1. Introduction

The Job Submission Description Language version 1.0 (JSDL 1.0) is a language for describing the requirements of computational jobs for submission to resources, particularly in Grid environments, though not restricted to the latter. The JSDL language contains a vocabulary and normative XML Schema that facilitate the expression of those requirements as a set of XML elements.

JSDL 1.0 defines a normative extension for describing “Executables on POSIX Conformant Hosts” in order to allow the specification of executable aspects of a job submission (executable to run, program arguments, environment variables, etc). While this extension is sufficient for executing jobs in typical UNIX environments, the extension supports features (for example, POSIX shell limits) that are not present on other operating systems such as the Windows Operating Environment, and as such, limit the ability for implementations of JSDL on those systems to be fully interoperable.

The purpose of this specification is to define an extension to JSDL 1.0 for describing a simple HPC application that is made up of an executable file running within an operating system process. It shares much in common with the JSDL POSIXApplication, but removes some of the features that present barriers to interoperability.

2. 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 [BRADNER1]

Pseudo-schemas are provided for each component, before the description of the component. They use BNF-style conventions for attributes and elements: ‘?’ denotes zero or one occurrences; ‘*’ denotes zero or more occurrences; ‘+’ denotes one or more occurrences. Attributes are conventionally assigned a value which corresponds to their type, as defined in the normative schema.

```
<!-- sample pseudo-schema -->
<defined_element
      required_attribute_of_type_string="xsd:string"
      optional_attribute_of_type_int="xsd:int"? >
   <required_element />
   <optional_element />?
   <one_or_more_of_this_element />+
</defined_element>
```

This specification uses namespace prefixes throughout; they are listed in Table 2-1. Note that the choice of any namespace prefix is arbitrary and not semantically significant.

Table 2-1: Prefixes and namespaces used in this specification.

Prefix	Namespace
xsd	http://www.w3.org/2001/XMLSchema
jsdl-hpcpa	http://schemas.ggf.org/jsdl/2006/07/jsdl-hpcpa

The terms *JSDL element* and *JSDL attribute* indicate that the corresponding language construct is represented as an XML element and XML attribute in the normative JSDL schema.
The term *JSDL document* refers to a well formed XML document that can be validated against the normative JSDL Schema definition contained in [JSDL10].

The keyword *present*, when used to refer to a JSDL element, means that an instance of that element appears in a JSDL document.

The keyword *support*, with respect to a JSDL consuming system supporting the JSDL specification and the language constructs, refers to the ability of that system to parse a JSDL document. That is, the consuming system MUST be able to interpret the language constructs and assign to them the semantics described in this specification and the values assigned to them in that JSDL document. All JSDL compliant consuming systems must, therefore, support all of the JSDL language constructs.

The keyword *satisfy*, with respect to a consuming system satisfying a JSDL document, means that the system can both support and satisfy all elements present in the document. An element is satisfied when the consuming system can provide an appropriate implementation of the value assigned to that element in the JSDL document.

3. Executables Running as Operating System Processes

This normative extension defines a schema describing a request to run an executable as an operating system process. It is worth noting that if any of the optional sub-elements of HPCProfileApplication are not defined in the JSDL document, then the system consuming the JSDL document may choose appropriate defaults, but that those defaults are not defined within this specification, and thus have no well-defined semantics.

The namespace prefix used for this schema in the specification is “jsdl-hpcpa.” The normative namespace for this schema is given in Table 2-1.

3.1 HPCProfileApplication Element

3.1.1 Definition

This element describes an application execution request that meets the requirements of the OGSA HPC Profile, including such information as is required to configure the executable instance to run. It contains Executable, Argument, Input, Output, Error, WorkingDirectory, Environment, and UserName elements. If it is present as a sub-element of the JSDL Application element it MUST appear only once.

3.1.2 Multiplicity

The multiplicity of this element (in its intended context) is zero or one.

3.1.3 Type

This is a complex type. It MUST support all of the following elements:

- Executable
- Argument
- Input
- Output
- Error
- WorkingDirectory
- Environment
- UserName

3.1.4 Attributes

The following attributes are defined:

- name—an optional name for the HPCProfileApplication element. Its type is xsd:NCName so that it can be reused and referred to from outside the containing JSDL document.

3.1.5 Pseudo Schema

```
<HPCProfileApplication name=”xsd:NCName”?>
   <Executable ... />?
   <Argument ... />*
   <Input ... />?
   <Output ... />?
   <Error ... />?
   <WorkingDirectory ... />?
   <Environment ... />*
   <UserName ... />?
</HPCProfileApplication>
```

3.2 Executable Element

3.2.1 Definition

This element is a string specifying the command to execute. It is either a full pathname for the executable, or a path that is relative to the WorkingDirectory element (if specified). If Executable is a relative pathname, and WorkingDirectory is not specified, then the directory that is used to resolve the relative pathname is chosen by the consuming system, and is not defined by this specification. If Executable is a filename with no full or relative path, then the consuming system is free to locate the referenced filename using whatever mechanism is supported by the consuming system. For example, the PATH environment variable might be used to look up the location of the supplied filename.

3.2.2 Multiplicity

The multiplicity of this element in the context of an HPCProfileApplication element is zero or one.

3.2.3 Type

The type of this element is xsd:string.

3.2.4 Attributes

No attributes are defined.

3.2.5 Pseudo Schema

```
<Executable> xsd:string </Executable>
```

3.2.6 Examples

Explicitly named executable:

```
<jsdl-hpcpa:Executable>/usr/local/bin/gnuplot</jsdl-hpcpa:Executable>
```

The consuming system could look this up using the PATH environment variable:

```
<jsdl-hpcpa:Executable>gnuplot</jsdl-hpcpa:Executable>
```

3.3 Argument Element

3.3.1 Definition

This element is a string specifying an argument element for the application. Argument elements can be empty, and multiple elements MUST NOT be combined.

3.3.2 Multiplicity

The multiplicity of this element in the context of an HPCProfileApplication element is zero or more.

3.3.3 Type

The type of this element is xsd:string.

3.3.4 Attributes

No attributes are defined.

3.3.5 Pseudo Schema

```
<Argument> xsd:string </Argument>*
```

3.3.6 Examples

Specify “CLASSPATH” and the class to invoke for a Java application:

```
<jsdl-hpcpa:Argument>-cp</jsdl-hpcpa:Argument>
<jsdl-hpcpa:Argument>./example.jar</jsdl-hpcpa:Argument>
<jsdl-hpcpa:Argument>org.example.Main</jsdl-hpcpa:Argument>
```

would be translated to ‘[java] -cp ./example.jar org.example.Main’.

Start the Apache2 service on a Windows box without being Administrator:

```
<jsdl-hpcpa:Argument>/user:Administrator@WORKGROUP</jsdl-
hpcpa:Argument>
<jsdl-hpcpa:Argument>net start Apache2</jsdl-hpcpa:Argument>
```

would be translated to ‘[runas] /user:Administrator@WORKGROUP “net start Apache2”’.

Echo a concatenated list of arguments to stdout:

```
<jsdl-hpcpa:Argument>foo</jsdl-hpcpa:Argument>
<jsdl-hpcpa:Argument>bar</jsdl-hpcpa:Argument>
<jsdl-hpcpa:Argument>baz</jsdl-hpcpa:Argument>
```

would be translated to ‘[echo]foo bar baz’,

```
<jsdl-hpcpa:Argument>foo</jsdl-hpcpa:Argument>
<jsdl-hpcpa:Argument>bar</jsdl-hpcpa:Argument>
<jsdl-hpcpa:Argument></jsdl-hpcpa:Argument>
<jsdl-hpcpa:Argument>baz</jsdl-hpcpa:Argument>
```

would be translated to '[echo]foo bar "" baz', and

```
<jsdl-hpcpa:Argument>foo</jsdl-hpcpa:Argument>
<jsdl-hpcpa:Argument>bar</jsdl-hpcpa:Argument>
<jsdl-hpcpa:Argument>baz</jsdl-hpcpa:Argument>
<jsdl-hpcpa:Argument></jsdl-hpcpa:Argument>
```

would be translated to '[echo]foo bar baz ""'.

3.4 Input Element

3.4.1 Definition

This element is a string specifying the input (Standard Input) for the command. The Input element is an absolute filename or a filename relative to the WorkingDirectory element (if specified). If Input is a relative pathname, and WorkingDirectory is not specified, then the directory that is used to resolve the relative pathname is chosen by the consuming system, and is not defined by this specification. If this element is not present then it is not defined and the consuming system MAY choose any value. The format for this element is not defined in this specification. If the consuming system does not understand the format of this element it SHOULD raise a fault. Furthermore, the consuming system MAY raise a fault specific to this error condition.

3.4.2 Multiplicity

The multiplicity of this element in the context of an HPCProfileApplication element is zero or one.

3.4.3 Type

The type of this element is xsd:string.

3.4.4 Attributes

No attributes are defined.

3.4.5 Pseudo Schema

```
<Input> xsd:string </Input>?
```

3.4.6 Examples

The standard input should be taken from "CWD/input.txt":

```
...
<jsdl-hpcpa:Input>input.txt</jsdl-hpcpa:Input>
...
```

3.5 Output Element

3.5.1 Definition

This element is a string specifying the output (Standard Output) for the command. The Output element is an absolute filename or a filename relative the WorkingDirectory element (if specified). If Input is a relative pathname, and WorkingDirectory is not specified, then the directory that is used to resolve the relative pathname is chosen by the consuming system, and is not defined by this specification. If this element is not present then it is not defined and the consuming system

MAY choose any value. The format for this element is not defined in this specification. If the consuming system does not understand the format of this element it SHOULD raise a fault. Furthermore, the consuming system MAY raise a fault specific to this error condition.

3.5.2 Multiplicity

The multiplicity of this element in the context of an HPCProfileApplication element is zero or one.

3.5.3 Type

The type of this element is xsd:string.

3.5.4 Attributes

No attributes are defined.

3.5.5 Pseudo Schema

```
<Output> xsd:string </Output>?
```

3.5.6 Examples

The standard output should be put in “CWD/output.txt”:

```
...
<jsdl-hpcpa:Output>output.txt</jsdl-hpcpa:Output>
...
```

3.6 Error Element

3.6.1 Definition

This element is a string specifying the error output (Standard Error) for the command. The Error element is an absolute filename or a filename relative to the WorkingDirectory element (if specified). If Input is a relative pathname, and WorkingDirectory is not specified, then the directory that is used to resolve the relative pathname is chosen by the consuming system, and is not defined by this specification. If this element is not present then it is not defined and the consuming system MAY choose any value. The format for this element is not defined in this specification. If the consuming system does not understand the format of this element it SHOULD raise a fault. Furthermore, the consuming system MAY raise a fault specific to this error condition.

3.6.2 Multiplicity

The multiplicity of this element in the context of an HPCProfileApplication element is zero or one.

3.6.3 Type

The type of this element is xsd:string.

3.6.4 Attributes

No attributes are defined.

3.6.5 Pseudo Schema

```
<Error> xsd:string </Error>?
```

3.6.6 Examples

The standard error should be put in “CWD/error.txt”:

```
...
<jsdl-hpcpa:Error>error.txt</jsdl-hpcpa:Error>
...
```

3.7 WorkingDirectory Element

3.7.1 Definition

This element is a string specifying the starting directory required by the job to execute. This element MUST specify an absolute pathname. The format for this element is not defined in this specification. If the consuming system does not understand the format of this element it SHOULD raise a fault. Furthermore, the consuming system MAY raise a fault specific to this error condition. If this element is not present then it is not defined and the consuming system MAY choose any value. If the working directory specified does not exist, then the consuming system SHOULD raise a fault. While the working directory indicates the location where the job defined by the JSDL document is started, this specification defines no semantics about the properties of the working directory. For example, no statement is made about whether the running process may change directory out of the working directory, either to a parent or a sub-directory.

3.7.2 Multiplicity

The multiplicity of this element in the context of an HPCProfileApplication element is zero or one.

3.7.3 Type

The type of this element is xsd:string.

3.7.4 Attributes

No attributes are defined.

3.7.5 Pseudo Schema

```
<WorkingDirectory> xsd:string </WorkingDirectory>?
```

3.7.6 Examples

The job’s working directory must be the directory /home/username:

```
<WorkingDirectory>/home/username</WorkingDirectory>
```

3.8 Environment Element

3.8.1 Definition

This element specifies the name and value of an environment variable that will be defined for the job in the execution environment.

3.8.2 Multiplicity

The multiplicity of this element in the context of an HPCProfileApplication element is zero or more.

3.8.3 Type

The type of this element is xsd:string. This is the literal value of the environment variable. It MUST NOT be empty.

3.8.4 Attributes

The following attributes are defined:

- name—the name of the environment variable.

3.8.5 Pseudo Schema

```
<Environment name=”xsd:NCName”> xsd:string </Environment>*
```

3.8.6 Examples

Set the SHELL to bash:

```
<Environment name=”SHELL”>/bin/bash</Environment>
```

3.9 UserName Element

3.9.1 Definition

This element is a string that defines the user name to be used when executing the application. If present, the consuming system MUST ensure that it is compatible with any available security context, and MUST reject the request of such compatibility cannot be ensured. If it is not present then it is not defined and the consuming system MAY select a user name based on implementation.

3.9.2 Multiplicity

The multiplicity of this element in the context of an HPCProfileApplication element is zero or one.

3.9.3 Type

The type of this element is xsd:string.

3.9.4 Attributes

No attributes are defined.

3.9.5 Pseudo Schema

```
<UserName> xsd:string </UserName>?
```

3.9.6 Examples

A UNIX or Windows user name:

```
<jsdl-hpcpa:UserName>frank</jsdl-hpcpa:UserName>
```

A Windows domain qualified user name:

```
<jsdl-hpcpa:UserName>MYDOMAIN\frank</jsdl-hpcpa:UserName>
```

3.9.7 Security Considerations

It is intended that this element be used to select the user name to run with in circumstances where the prevailing security context can map the user's identity to multiple user names. When it is known that the security context has a simple mapping, it is RECOMMENDED that the creator of the HPCProfileApplication element omits the UserName element.
The ability to describe a desired UserName does not guarantee that the system consuming the document will allow or be able to run a job under the requested context; any job whose UserName element is inconsistent with the context MUST NOT be executed in a way inconsistent with the context. The semantics of how UserName are used are out of scope of this document.

4. Security Considerations

This specification defines a language for describing the requirements of computational jobs for submission to Grids and other job management systems. It is assumed that job submission must be secured, but such mechanisms are out of scope of this specification.

Security considerations specific to a particular element are listed under the specification of the particular element.

5. Contributors

Author Information:

Marty Humphrey
University of Virginia
Email: humphrey@cs.virginia.edu

Chris Smith
Platform Computing, Inc.
Email: csmith@platform.com

Marvin Theimer
Microsoft Corporation
Email: theimer@microsoft.com

Glenn Wasson
University of Virginia
Email: wasson@virginia.edu

We gratefully acknowledge the contributions made to this specification by Donal Fellows.

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) Karl Czajkowski, Peter Lane, and Andreas Savva.

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 the readability and accuracy of this document.

6. 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.

7. Disclaimer

This document and the information contained herein is provided on an “As Is” basis and the OGF 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.

8. Full Copyright Notice

9. Normative References

[RFC 2119] Bradner, S. Key Words for Use in RFCs to Indicate Requirement Levels, RFC 2119. March 1997. Available at http://www.ietf.org/rfc/rfc2119.txt

[JSDL10] 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

1. Normative HPC Profile Application XSD

This section contains the full normative XML Schema definition for JSDL HPCProfileApplication 1.0.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd=http://www.w3.org/2001/XMLSchema
   xmlns:jsdl-hpcpa=http://schemas.ogf.org/jsdl/2006/07/jsdl-hpcpa
   targetNamespace=http://schemas.ogf.org/jsdl/2006/07/jsdl-hpcpa
   elementFormDefault="qualified">
   <!--============================================================-->
   <xsd:complexType name="Environment_Type">
      <xsd:simpleContent>
         <xsd:extension base="xsd:string">
            <xsd:attribute name="name" type="xsd:NCName"
               use="required"/>
            <xsd:anyAttribute namespace="##other"
               processContents="lax"/>
         </xsd:extension>
      </xsd:simpleContent>
   </xsd:complexType>
   <!--============================================================-->
   <xsd:complexType name="Argument_Type">
      <xsd:simpleContent>
         <xsd:extension base="xsd:string">
            <xsd:anyAttribute namespace="##other"
               processContents="lax"/>
         </xsd:extension>
      </xsd:simpleContent>
   </xsd:complexType>
   <!--============================================================-->
   <xsd:complexType name="FileName_Type">
      <xsd:simpleContent>
         <xsd:extension base="xsd:string">
            <xsd:anyAttribute namespace="##other"
               processContents="lax"/>
         </xsd:extension>
      </xsd:simpleContent>
   </xsd:complexType>
   <!--============================================================-->
   <xsd:complexType name="DirectoryName_Type">
      <xsd:simpleContent>
         <xsd:extension base="xsd:string">
            <xsd:anyAttribute namespace="##other"
               processContents="lax"/>
         </xsd:extension>
      </xsd:simpleContent>
   </xsd:complexType>
   <!--============================================================-->
   <xsd:complexType name="UserName_Type">
      <xsd:simpleContent>
         <xsd:extension base="xsd:string">
            <xsd:anyAttribute namespace="##other"
               processContents="lax"/>
         </xsd:extension>
      </xsd:simpleContent>
   </xsd:complexType>
   <!--============================================================-->
   <xsd:complexType name="HPCProfileApplication_Type">
      <xsd:sequence>
         <xsd:element ref="jsdl-hpcpa:Executable" minOccurs=”0”/>
```

```
        <xsd:element ref="jsdl-hpcpa:Argument" minOccurs="0"
          maxOccurs="unbounded"/>
        <xsd:element ref="jsdl-hpcpa:Input" minOccurs="0"/>
        <xsd:element ref="jsdl-hpcpa:Output" minOccurs="0"/>
        <xsd:element ref="jsdl-hpcpa:Error" minOccurs="0"/>
        <xsd:element ref="jsdl-hpcpa:WorkingDirectory" minOccurs="0"/>
        <xsd:element ref="jsdl-hpcpa:Environment" minOccurs="0"
          maxOccurs="unbounded"/>
        <xsd:element ref="jsdl-hpcpa:UserName" minOccurs="0"/>
      </xsd:sequence>
      <xsd:attribute name="name" type="xsd:NCName" use="optional"/>
      <xsd:anyAttribute namespace="##other" processContents="lax"/>
   </xsd:complexType>
   <!--============================================================-->
   <xsd:element name="HPCProfileApplication"
      type="jsdl-hpcpa:HPCProfileApplication_Type"/>
   <xsd:element name="Executable" type="jsdl-hpcpa:FileName_Type"/>
   <xsd:element name="Argument" type="jsdl-hpcpa:Argument_Type"/>
   <xsd:element name="Input" type="jsdl-hpcpa:FileName_Type"/>
   <xsd:element name="Output" type="jsdl-hpcpa:FileName_Type"/>
   <xsd:element name="Error" type="jsdl-hpcpa:FileName_Type"/>
   <xsd:element name="WorkingDirectory"
      type="jsdl-hpcpa:DirectoryName_Type"/>
   <xsd:element name="Environment" type="jsdl-hpcpa:Environment_Type"/>
   <xsd:element name="UserName" type="jsdl-hpcpa:UserName_Type"/>
</xsd:schema>
```