

## JSDL Parameter Sweep Job Extension

### Status of This Document

This document provides information to the Grid community about a standardized markup language specifying parameter sweep jobs. Distribution is unlimited.

### Copyright Notice

Copyright © Open Grid Forum 2006-2008. All Rights Reserved.

### Abstract

This document specifies the syntax and semantics of the proposed Parameter Sweep extension to the Job Submission Description Language (JSDL) 1.0 [JSDL].  
The syntax and semantics defined in this document provide an alternative to explicitly submitting thousands of individual JSDL job submissions of the same base job template, but, each with a different parameter set.

### Contents

Abstract.....	1
1. Introduction .....	3
1.1 Notational Conventions .....	3
1.2 XML Namespaces .....	3
2. Parameter Sweep.....	4
2.1 Function.....	4
2.1.1 XML Representation .....	5
2.2 Parameter.....	5
2.2.1 XML Representation .....	5
2.3 Assignment .....	5
2.3.1 XML Representation .....	6
2.3.2 Example .....	6
2.4 Sweep.....	7
2.4.1 Nested Sweep elements.....	7
2.4.2 XML Representation .....	8
2.4.3 Example .....	8
3. Normative Parameter substitute definitions.....	9
3.1 DocumentNode Parameter substitute .....	9
3.1.1 XML Representation .....	9
3.1.2 NamespaceBinding.....	10
3.1.2.1 XML Representaiaon.....	10
3.1.3 Example 1.....	10
3.1.4 Example 2.....	10
3.1.5 Example 3.....	11
3.1.6 Example 4.....	13
3.2 FileSweep Parameter substitute.....	13
3.2.1 FileSweep.....	13
3.2.1.1 XML Representation.....	14
3.2.1.2 Example 1 .....	14
3.2.2 TemplateFile.....	14

3.2.2.1	XML Representation.....	14
3.2.3	Example 1 .....	15
3.2.3.1	Example 2 .....	15
3.2.4	FileToken .....	15
3.2.4.1	XML Representation.....	15
3.2.4.2	Example 1 .....	16
3.2.4.3	Example 2 .....	16
4.	Normative Function substitute definitions.....	16
4.1	Values function .....	16
4.1.1	XML Representation .....	17
4.1.2	Example .....	17
4.2	LoopInteger function.....	17
4.2.1	XML Representation .....	18
4.2.2	Example 1 .....	19
4.2.3	Example 2.....	19
4.2.4	Example 3.....	19
4.3	LoopDouble function.....	19
4.3.1	XML Representation .....	19
4.3.2	Example 1.....	20
4.3.3	Example 2.....	20
5.	Integration with JSDL v1.0.....	20
6.	Examples .....	21
6.1	Example 1 .....	21
6.1.1	Submitted JSDL Job template.....	21
6.1.2	Parameter Sweep individual job templates .....	22
6.2	Example 2 .....	24
6.3	Example 3 .....	25
6.4	Example 4 .....	27
6.5	Example 5 .....	28
6.6	Example 6 .....	29
6.7	Example 7 .....	31
6.8	Example 8 .....	32
7.	Further Examples .....	33
7.1	Example 1 .....	34
7.2	Example 2 .....	35
7.3	Example 3 .....	36
7.4	Example 4 .....	38
7.5	Example 5 .....	40
7.6	Example 6 .....	40
8.	Security Considerations.....	41
9.	Contributors .....	41
10.	Intellectual Property Statement .....	41
11.	Disclaimer .....	41
12.	Full Copyright Notice .....	42
13.	References.....	42

## 1. Introduction

A Parameter Sweep is a job that internally defines a collection of jobs. For each of the jobs in the collection, the value of one or more of the job parameters may be changed in some preordained fashion, from the previous job in the collection. Hence, an array of values may be assigned to a parameter whose value is to be changed to successive elements in that array, for each consecutive job in the collection. This can be done for multiple parameters, so that more than one parameter's value can change between each successive job in the collection.

In the definition of a Parameter Sweep for submission of a collection of jobs, the goal is to capture the array of values for each parameter, whose value is to be changed, in a succinct and systematic manner. For this to be possible, a means to describe Parameter Sweeps for submission are needed (i.e. a Parameter Sweep schema specification).

Given a succinct means to describe a Parameter Sweep (using a suitable schema), a resource management system can then instantiate and manage each job in the job collection defined in that Parameter Sweep.

Hence this document standardises the language for *describing* Parameter Sweeps. This document does not define how to instantiate and manage the described Parameter Sweep, which may be left as an implementation detail, or defined in Profiles on other specifications that deal with processing Job Submissions in Grids.

### 1.1 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 [BRADNER].

This document describes XML Information Sets and inherits the square bracket notation of [INFOSET].

When describing concrete XML schemas [SCHEMA1], [SCHEMA2], this specification uses the notational convention of WS-Security [WSSEC]. Specifically, each member of an element's [children] or [attributes] properties, is described using an XPath-like notation (e.g.: `/x:MyHeader/x:SomeProperty/@value1`). The use of {any} indicates the presence of an element wildcard (`<xsd:any/>`). The use of @{any} indicates the presence of an attribute wildcard (`<xsd:anyAttribute/>`).

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>
```

## 1.2 XML Namespaces

This specification uses a number of namespace prefixes throughout; they are listed in Table 1. Note that the choice of any namespace prefix is arbitrary and not semantically significant (see [BRAY]).

**Table 1** Namespaces and prefixes used in this document

Prefix	Namespace
jsdl	http://schemas.ggf.org/jsdl/2005/11/jsdl
jsdl-posix	http://schemas.ggf.org/jsdl/2005/11/jsdl-posix
sweep	http://schemas.ogf.org/jsdl/2007/04/sweep
sweepfunc	http://schemas.ogf.org/jsdl/2007/04/sweep/functions
file-sweep	http://schemas.ogf.org/jsdl/2008/08/file-sweep
xsd	http://www.w3.org/2001/XMLSchema

## 2. Parameter Sweep

This section defines the core information elements of the Parameter Sweep extension to JSDL 1.0.

### 2.1 Function

The Function element is the source of the values that are assigned to Parameters in a Parameter Sweep, as described below.

A Function is not necessarily a function in a mathematical sense. Rather, it is better described as a mathematical sequence. A Function **MUST** have the following properties:

- A Function yields a finite set of values;
- A Function has a first value;
- A Function has a last value;
- A Function has concepts of “current” and “next” values; and
- A Function yields the same parameter values in the same order every time it is used from the beginning, which is defined by its initial conditions.

The cardinality of a Function is defined as the number of values it yields.

The Function element is defined as an abstract XML Schema element. Hence it serves as an extension point in this Parameter Sweep specification: Profiles on this specification, other standards or end users **MAY** define specialised substitutes for the Function element and use them when necessary. Section 3 defines standard Function substitutes that **MUST** be supported by any implementation of this specification.

The values of a Function substitute are not limited to a particular data type (e.g. integers, doubles, etc.). Instead, values can be of any data type, from primitive scalar values (Boolean, Integers, etc.) to complete XML element structures. Concrete Function substitutes implicitly define the data type of its values. For example, the LoopInteger function defines that all values are of data type `xsd:integer`. The data type of a Function substitute in a XML document **MUST** match the data type of the associated Parameter substitute.

### 2.1.1 XML Representation

The Function element is rendered in XML as:

```
<sweep:Function abstract="true"/>
```

Where:

#### **/sweep:Function**

Represents the Function element. Since the Function element is an abstract XML element it MUST NOT occur in a XML document by itself. It MUST be substituted by a valid Function substitute instead.

### 2.2 Parameter

The Parameter element is the target of the values yielded by a Function substitute.

Like the Function element the Parameter element is defined as an abstract XML Schema element. Hence it serves as an extension point in this Parameter Sweep specification: Profiles on this specification, other standards or end users MAY define specialised substitutes for the Parameter element and use them when necessary. Section 3 defines standard Parameter substitutes that MUST be supported by any implementation of this specification.

Concrete Parameter substitutes define how exactly a value is applied when evaluating the Parameter Sweep.

Parameter substitutes implicitly define a data type. The Parameter substitute's data type MUST match the data type of the associated Function substitute.

### 2.2.1 XML Representation

The Parameter element is rendered in XML as:

```
<sweep:Parameter abstract="true"/>
```

Where:

#### **/sweep:Parameter**

Represents the Parameter element. Since the Parameter element is an abstract XML element it MUST NOT occur in a XML document by itself. It MUST be substituted by a valid Parameter substitute instead.

### 2.3 Assignment

The Assignment element associates one Function substitute with one or more Parameter substitutes.

Each value yielded by a Function substitute MUST be assigned or applied to each associated Parameter substitute in due course during the evaluation of the Parameter Sweep.

For example, if a Function substitute yields five values, then each of the five values must be applied to subsequent five JSDL job submissions.

For evaluation and validation purposes the Assignment element inherits the properties of its child Function element as follows:

- An Assignment element yields a finite set of value assignments;
- An Assignment element has a first value assignment;
- An Assignment element has a last value assignment;
- An Assignment element has concepts of “current” and a “next” value assignments; and
- An Assignment element yields the same value assignments in the same order every time it is used from the beginning, which is defined by its initial conditions.

The cardinality of the Assignment element is defined as the number of value assignments it yields. Hence, the cardinality of an Assignment element is always the same as the cardinality of its child Function element.

Implicit type casting is not allowed. Therefore, the data types of the child Function substitute and the data type(s) of the child Parameter substitute(s) MUST match. If the data types do not match then the JSDL Job Template document MUST be rejected.

### 2.3.1 XML Representation

The Assignment element is rendered in XML as:

```
<sweep:Assignment>
  <sweep:Parameter/>+
  <sweep:Function/>
</sweep:Assignment>
```

Where:

#### **/sweep:Assignment**

Represents the Assignment element. Its XML type is defined as `xsd:complexType` with no content other than XML elements.

#### **/sweep:Assignment/sweep:Parameter**

Represents the Parameter element as defined in section 2.2. At least one Parameter substitute MUST be present in a JSDL Job Template.

#### **/sweep:Assignment/sweep:Function**

Represents the Function element as defined in section 2.1. Exactly one Function substitute MUST be present in a JSDL Job Template.

### 2.3.2 Example

The following example illustrates the use of the Assignment element. It uses normative definitions of (a) a Parameter substitute, “DocumentNode” and (b) a Function substitute, “Values” as defined in sections 3 and 4:

```
<sweep:Assignment>
  <sweep:DocumentNode>
    <sweep:NamespaceBinding
      ns="http://schemas.ggf.org/jsdl/2005/11/jsdl-posix"
      prefix="jsdl-posix" />
    <sweep:Match>
      /*//jsdl-posix:POSIXApplication[1]/jsdl-posix:Argument[2]
    </sweep:Match>
  </sweep:DocumentNode>
  <sweep:DocumentNode>
    <sweep:NamespaceBinding
```

```
    ns="http://schemas.ggf.org/jsdl/2005/11/jsdl-posix"  
    prefix="jsdl-posix" />  
  <sweep:Match>  
    /*//jsdl-posix:POSIXApplication[1]/jsdl-posix:Argument[4]  
  </sweep:Match>  
</sweep:DocumentNode>  
<sweepfunc:Values>  
  <sweepfunc:Value>foo</sweepfunc:Value>  
  <sweepfunc:Value>bar</sweepfunc:Value>  
  <sweepfunc:Value>baz</sweepfunc:Value>  
</sweepfunc:Values>  
</sweep:Assignment>
```

In this example, two parameters are associated with one function, which yields three values of type `xsd:string`. Accordingly, the example Assignment yields three value assignments, too. For each value assignment, the respective value must be applied to both referenced parameters.

The two Parameter substitutes are of type `sweep:DocumentNode` (for the definition see section 3), and select the second and fourth `jsdl-Posix:Argument` elements in a hypothetical enclosing JSDL Job Template. The Function substitute is a `sweepfunc:Values` function (for the definition see section 4).

For each value assignment, both parameters will, at the same time, apply the values “foo”, “bar” and “baz” in subsequent JSDL Job submissions.

Chapter 6 provides more detailed examples.

## 2.4 Sweep

The Sweep element describes how to modify an existing JSDL Job Template in order to get a Parameter Sweep JSDL Job Template. More than one Sweep element MAY occur in a JSDL Job Template. In this case, all Sweep elements MUST be evaluated in order of appearance. A Sweep element MUST NOT describe Parameter Sweeps that modify any Sweep element contained in the same XML document, including this Sweep element.

The Sweep element defines the coordination of Assignment element evaluation.

The Sweep element MUST contain at least one Assignment element. It MAY contain nested Sweep elements.

Sibling Assignment child elements MUST have the same cardinality. Otherwise the containing JSDL Job Template MUST be rejected.

All contained Assignment elements MUST be evaluated in parallel: All  $n^{\text{th}}$  value assignments of each contained Assignment element form a “Value Assignment Set”.

Hence the Sweep element inherits the properties of its child Assignment elements as follows:

- A Sweep element yields a finite set of Value Assignment Set;
- A Sweep element has a first Value Assignment Set;
- A Sweep element has a last Value Assignment Set;
- A Sweep element has concepts of “current” and a “next” Value Assignment Set; and
- A Sweep element yields the same Value Assignment Sets in the same order every time it is used from the beginning, which is defined by its initial conditions.

The cardinality of the Sweep element is defined as the number of Value Assignment Sets it

yields. Hence, the cardinality of a Sweep element is always the same as the cardinality of its child Assignment elements.

Each Value Assignment Set **MUST** be applied at once; Each Value Assignment Set is applied to a copy of the original JSDL Job Template. The thus modified job template represents an individual job from the collection of jobs defined by the Parameter Sweep, and concludes one iteration over a Sweep element.

#### 2.4.1 Nested Sweep elements

A Sweep element may contain nested Sweep elements.

The evaluation of nested Sweep elements slightly changes the evaluation rules: For each Value Assignment Set of the parent Sweep element, all nested Sweep elements are fully evaluated anew in order of appearance until all parent Sweep element's Value Assignment Sets are evaluated.

When applying a nested Sweep element's current Value Assignment Set all Value Assignment Sets of all parent Sweep elements are in scope for application to the copy of the JSDL Job template, hence **MUST** be assigned at the same time.

This specification does not restrict the depth of Sweep element nesting, potentially leading to practically indefinite processing time for Parameter Sweeps. Implementations, however, **MAY** limit the nesting for practical reasons. In such circumstances the enclosing JSDL Job Template **MUST** be rejected and an appropriate fault communicated.

#### 2.4.2 XML Representation

The Sweep element is rendered in XML as:

```
<sweep:Sweep>  
  <sweep:Assignment/>+  
  <sweep:Sweep/>*  
</sweep:Sweep>+
```

Where:

##### **/sweep:Sweep**

Represents the Sweep element. Sibling Sweep elements within the same XML document constitute a single Parameter Sweep. If this is not the desired behaviour, higher level XML document partitioning must be used to ensure proper scope of the [sweep] elements in defining multiple [parameter sweep] definitions.

##### **/sweep:Sweep/sweep:Assignment**

Represents the Assignment element as defined in section 2.3. The Assignment element **MUST** appear at least once. It **MAY** appear more than once.

##### **/sweep:Sweep/sweep:Sweep**

Represents a nested Sweep element as defined in this section. It **MAY** appear zero or more times.

#### 2.4.3 Example

The following example illustrates the use of the Sweep information element.

```
<sweep:Sweep>
```

```
jsdl-wg@ogf.org
```



```
<sweep:Assignment>
  <sweep:DocumentNode>
    <sweep:NamespaceBinding
      ns="http://schemas.ggf.org/jsdl/2005/11/jsdl-posix"
      prefix="jsdl-posix" />
    <sweep:Match>
      /*//jsdl-posix:POSIXApplication[1]/jsdl-
posix:Argument[2]
    </sweep:Match>
  </sweep:DocumentNode>
  <sweepfunc:LoopInteger start="1" end="10" />
</sweep:Assignment>
</sweep:Sweep>
```

In this example, one Parameter is associated with one Function: The second `jsdl-posix:Argument` of the first `jsdl-posix:POSIXApplication` element found at an arbitrary depth in the JSDL Template document being addressed, will receive the values 1, 2, 3, 4, etc. up to 10, constituting a collection of jobs defined by this [parameter sweep] definition with 10 individual jobs.

### 3. Normative Parameter substitute definitions

This section defines normative Parameter substitute element that MAY appear in a Parameter Sweep. All substitutes defined in this section MUST be supported by implementations of this specification.

#### 3.1 DocumentNode Parameter substitute

The DocumentNode element selects the contents of an XML element or attribute in the enclosing JSDL Job Template as the target for Parameter Sweep Function values.

The type of the DocumentNode's Match element is an XPath 2.0 expression [XPATH]. The evaluation of the XPath 2.0 expression MUST yield a [sequence]<sup>1</sup> containing exactly one [item]. The [item] can be a [node] or an [atomic value] as per the XPath 2.0 specification.<sup>2</sup> The XPath 2.0 expression MAY select parts of the contents of an XML element or attribute using any of the standard XPATH 2.0 functions, for example "fn:substring()".

Any two DocumentNode elements of the same Sweep element context MUST select truly disjoint [item]s. A consumer of a Parameter Sweep description SHOULD reject a JSDL job template that fails to comply with this requirement<sup>3</sup>. The semantics of the Parameter Sweep and the outcome are undefined if a consumer decides to accept such Parameter Sweeps.

##### 3.1.1 XML Representation

The DocumentNode information element is rendered in XML as:

```
<sweep:DocumentNode>
```

<sup>1</sup> The Information Set elements named [sequence], [item], [node] and [atomic value] are defined in XPath 2.0 [XPATH].

<sup>2</sup> Hence, it is perfectly allowable to select only fractions of the values of the XML elements or XML attributes that are specified in the target JSDL Job Template, e.g. by using the XPath 2.0 function "fn:substring()".

<sup>3</sup> Detecting such circumstances may be difficult or not possible at all hence Parameter Sweep consumers are allowed to accept such documents and "hope for the best".

```
<sweep:NamespaceBinding/>+  
<sweep:Match>xsd:string</sweep:Match>  
</sweep:DocumentNode>
```

Where:

**/sweep:DocumentNode**

Represents the DocumentNode information element. Its XML type is defined as `xsd:complexType`.

**/sweep:DocumentNode/sweep:NamespaceBinding**

Represents an explicit namespace binding used in the following `sweep:Match` XPath expression. This element is defined in section 3.1.2. This element **MUST** appear at least once and **MAY** appear more than once. Any two `sweep:NamespaceBinding` elements within the same `sweep:DocumentNode` element **MAY** map the same namespace to different prefixes. Within the context of one `sweep:DocumentNode`, each bound prefix **MUST** be unique. For each prefix used in the `sweep:Match` element there **MUST** be exactly one sibling `sweep:NamespaceBinding` present that declares that prefix.

**/sweep:DocumentNode/sweep:Match**

Represents the XPath expression that selects that portion of the JSDL Job Template that is target of the enclosing assignment. This element **MUST** appear exactly once. Its XML data type is defined as `xsd:string`.

### 3.1.2 NamespaceBinding

This element, although defined in the context of the `sweep:DocumentNode` **MAY** be used elsewhere if appropriate.

The `NamespaceBinding` element explicitly binds a namespace to a prefix that is used in expressions that are not guaranteed access to implicit namespace binding declarations in a XML document.

#### 3.1.2.1 XML Representaiaon

The `NamespaceBinding` information element is rendered in XML as:

```
<sweep:NamespaceBinding  
  ns="xsd:anyURI"  
  prefix="xsd:NCName" />
```

Where:

**/sweep:NamespaceBinding**

Represents the `NamespaceBinding` element.

**/sweep:NamespaceBinding/@ns**

Represents the namespace that is bound to a prefix. Its type is defined as `xsd:anyURI`.

**/sweep:NamespaceBinding/@prefix**

Represents the prefix that is bound to a given namespace. Its type is defined as `xsd:NCName`.

### 3.1.3 Example 1

The following example illustrates the use of the `DocumentNode` element.

```
<sweep:DocumentNode>  
  <sweep:NamespaceBinding
```

```
ns="http://schemas.ggf.org/jsdl/2005/11/jsdl-posix"  
prefix="jsdl-posix" />  
<sweep:Match>  
  /*//jsdl-posix:POSIXApplication[1]/jsdl-posix:Argument[2]  
</sweep:Match>  
</sweep:DocumentNode>
```

In this example, the XPath 2.0 expression selects the value of the second `jsdl-posix:Argument` of the first `jsdl-posix:POSIXApplication` element found at an arbitrary depth in the JSDL Template document being addressed. For the XPath expression, the namespace “`http://schemas.ggf.org/jsdl/2005/11/jsdl-posix`” is explicitly bound to the prefix “`jsdl-posix`”.

### 3.1.4 Example 2

The following example illustrates the ability to match on part of an element content through the use of the XPath substring function:

```
<jsdl:JobDefinition>  
  <jsdl:JobDescription>  
    <jsdl:Application>  
      <jsdl-posix:POSIXApplication>  
        <jsdl-posix:Executable>/bin/some_exe</jsdl-posix:Executable>  
        <jsdl-posix:Argument>-infile</jsdl-posix:Argument>  
        <jsdl-posix:Argument>in.NNN.dat</jsdl-posix:Argument>  
      </jsdl-posix:POSIXApplication>  
    </jsdl:Application>  
  </jsdl:JobDescription>  
<sweep:Sweep>  
  <sweep:Assignment>  
    <sweep:DocumentNode>  
      <sweep:NamespaceBinding>  
        ns="http://schemas.ggf.org/jsdl/2005/11/jsdl-posix"  
        prefix="jsdl-posix" />  
      <sweep:Match>  
        substring(/*//jsdl-posix:Argument[2], 4, 3)  
      </sweep:Match>  
    </sweep:DocumentNode>  
    <sweepfunc:Values>  
      <sweepfunc:Value>001</sweepfunc:Value>  
      <sweepfunc:Value>002</sweepfunc:Value>  
      <sweepfunc:Value>003</sweepfunc:Value>  
      <sweepfunc:Value>004</sweepfunc:Value>  
      <sweepfunc:Value>005</sweepfunc:Value>  
    </sweepfunc:Values>  
  </sweep:Assignment>  
</sweep:Sweep>  
</jsdl:JobDefinition>
```

The above yields the following 5 jobs:

```
/bin/some_exe -infile in.001.dat  
/bin/some_exe -infile in.002.dat  
/bin/some_exe -infile in.003.dat  
/bin/some_exe -infile in.004.dat  
/bin/some_exe -infile in.005.dat
```

### 3.1.5 Example 3

The following example shows a valid use of multiple DocumentNodes matching disjunct [item]s in a single JSDL template element. A nested sweep modifies the output filenames so that the output of each individual sweep instance can be identified.

```
<jSDL:JobDefinition>
  <jSDL:JobDescription>
    <jSDL:Application>
      <jSDL-posix:POSIXApplication>
        <jSDL-posix:Executable>/bin/some_exe</jSDL-posix:Executable>
        <jSDL-posix:Argument>-output</jSDL-posix:Argument>
        <jSDL-posix:Argument>out.XX.YY.ZZ.dat</jSDL-posix:Argument>
      </jSDL-posix:POSIXApplication>
    </jSDL:Application>
  </jSDL:JobDescription>
  <sweep:Sweep>
    <sweep:Assignment>
      <sweep:DocumentNode>
        <sweep:NamespaceBinding
          ns="http://schemas.ggf.org/jSDL/2005/11/jSDL-posix"
          prefix="jSDL-posix" />
        <sweep:Match>
          substring(*//jSDL-posix:Argument[2], 5, 2)
        </sweep:Match>
      </sweep:DocumentNode>
      <sweepfunc:Values>
        <sweepfunc:Value>01</sweepfunc:Value>
        <sweepfunc:Value>02</sweepfunc:Value>
      </sweepfunc:Values>
    </sweep:Assignment>
  </sweep:Sweep>
  <sweep:Sweep>
    <sweep:Assignment>
      <sweep:DocumentNode>
        <sweep:NamespaceBinding
          ns="http://schemas.ggf.org/jSDL/2005/11/jSDL-posix"
          prefix="jSDL-posix" />
        <sweep:Match>
          substring(*//jSDL-posix:Argument[2], 8, 2)
        </sweep:Match>
      </sweep:DocumentNode>
      <sweepfunc:Values>
        <sweepfunc:Value>10</sweepfunc:Value>
        <sweepfunc:Value>11</sweepfunc:Value>
      </sweepfunc:Values>
    </sweep:Assignment>
  </sweep:Sweep>
  <sweep:Sweep>
    <sweep:Assignment>
      <sweep:DocumentNode>
        <sweep:NamespaceBinding
          ns="http://schemas.ggf.org/jSDL/2005/11/jSDL-posix"
          prefix="jSDL-posix" />
        <sweep:Match>
          substring(*//jSDL-posix:Argument[2], 11, 2)
        </sweep:Match>
      </sweep:DocumentNode>
      <sweepfunc:Values>
```

```
        <sweepfunc:Value>20</sweepfunc:Value>
        <sweepfunc:Value>21</sweepfunc:Value>
    </sweepfunc:Values>
</sweep:Assignment>
</sweep:Sweep>
</sweep:Sweep>
</sweep:Sweep>
</jsdl:JobDefinition>
```

The above yields the following 8 jobs:

```
/bin/some_exe -output out.01.10.20.dat
/bin/some_exe -output out.01.10.21.dat
/bin/some_exe -output out.01.11.20.dat
/bin/some_exe -output out.01.11.21.dat
/bin/some_exe -output out.02.10.20.dat
/bin/some_exe -output out.02.10.21.dat
/bin/some_exe -output out.02.11.20.dat
/bin/some_exe -output out.02.11.21.dat
```

### 3.1.6 Example 4

The following example illustrates an invalid use of the DocumentNode element.

```
<sweep:Sweep>
...
  <sweep:DocumentNode>
    <sweep:NamespaceBinding
      ns="http://schemas.ggf.org/jsdl/2005/11/jsdl-posix"
      prefix="jsdl-posix" />
    <sweep:Match>
      /*//jsdl-posix:POSIXApplication/jsdl-posix:Argument[2]
    </sweep:Match>
  </sweep:DocumentNode>
...
  <sweep:DocumentNode>
    <sweep:NamespaceBinding
      ns="http://schemas.ggf.org/jsdl/2005/11/jsdl-posix"
      prefix="jsdl-posix" />
    <sweep:Match>
      substring(/*//jsdl-posix:POSIXApplication/jsdl-posix:Argument[2],
        1, 1)
    </sweep:Match>
  </sweep:DocumentNode>
...
</sweep:Sweep>
```

The two DocumentNode elements defined in the sample XML above both address the same XML document element value. The first DocumentNode element addresses the value entirely, while the second DocumentNode element addresses only a fraction of it. Both elements cannot be used together in the same Sweep element context, even though both DocumentNode elements are valid on their own.

## 3.2 FileSweep Parameter substitute

The FileSweep element defines the syntax and semantics for Parameter Sweeps that require

fixed input files with incrementally changing contents.

### 3.2.1 FileSweep

The FileSweep element is used to declare file tokens that exist within one or more template files as the target for Parameter Sweep Function values. A FileSweep element thus defines a 'search and replace' operation, where the implementing system replaces selected file tokens within selected template files for each sweep.

A FileSweep element contains one or more nested TemplateFile elements and one or more nested FileToken elements.

Any two FileSweep elements in the same Sweep element context that declare identical FileToken element values for the SAME template file is invalid. A File Sweep defined in that way is invalid and MUST be rejected (see 7.6 for an example).

#### 3.2.1.1 XML Representation

The FileSweep element is a complex type and is rendered in XML as:

```
<file-sweep:FileSweep substitutes="sweep:Parameter">
  <file-sweep:TemplateFile/>+
  <file-sweep:FileToken/>+
</file-sweep:FileSweep>
```

Where:

#### **/file-sweep:FileSweep**

Represents the FileSweep element which MAY appear as a Parameter Sweep Parameter substitute. Its XML type is defined as `xsd:complexType` with no content other than XML elements.

#### **/file-sweep:FileSweep/file-sweep:TemplateFile**

Represents the TemplateFile element as defined in section 3.2.2. Its XML type is defined as `xsd:complexType` with no content other than XML elements. The TemplateFile element MUST appear at least once. It MAY appear more than once.

#### **/file-sweep:FileSweep/file-sweep:FileToken**

Represents the FileToken element as defined in section 3.2.4. The FileToken element MUST appear at least once. It MAY appear more than once.

#### 3.2.1.2 Example 1

```
<file-sweep:FileSweep>
  <file-sweep:TemplateFile>
    <jsdl:FileName>data1.dat</jsdl:FileName>
  </file-sweep:TemplateFile>
  <file-sweep:FileToken value="AOA"/>
  <file-sweep:FileToken value="RE"/>
</file-sweep:FileSweep>
```

In this example, the FileSweep element declares a single template file and two FileToken elements ('AOA' and 'RE') which select the corresponding two file tokens within data1.dat. Sections 3.2.2 and 3.2.4 describe the TemplateFile and FileToken elements respectively.

### 3.2.2 TemplateFile

The TemplateFile element identifies a local text file that must exist on the execution host. A template file contains token values. Template files should be text files only, as binary files may become corrupted.

It is RECOMMENDED that a corresponding `jsdl:DataStaging` element is also defined in the enclosing JSDL job template to guarantee the template file is always available on different execute hosts.

#### 3.2.2.1 XML Representation

The TemplateFile element is a complex type and must support the `jsdl:FileName` and `jsdl:FileSystemName` elements as defined in [JSDL]. The TemplateFile element is rendered in XML as:

```
<file-sweep:TemplateFile>  
  <jsdl:FileName/>  
  <jsdl:FileSystemName/>?  
</file-sweep:TemplateFile>
```

Where:

##### **/file-sweep:TemplateFile/jsdl:FileName**

Represents the `jsdl:FileName` element as defined [JSDL]. The referenced file must be a local text file on the execution host. The `jsdl:FileName` element is MANDATORY and MUST appear only once.

##### **/file-sweep:TemplateFile/jsdl:FileSystemName**

Represents the `jsdl:FileSystemName` element as defined in [JSDL]. The `jsdl:FileSystemName` element is OPTIONAL. If the `jsdl:FileSystemName` is specified, then the `jsdl:FileName` is relative to a `jsdl:FileSystem` declaration that MUST be defined within the enclosing JSDL job template.

### 3.2.3 Example 1

The following examples illustrate the use of the TemplateFile element.

```
<file-sweep:TemplateFile>  
  <jsdl:FileName>data1.dat</jsdl:FileName>  
</file-sweep:TemplateFile>
```

In this example, the template file 'data1.dat' is relative to the working job directory as determined by the consuming system.

#### 3.2.3.1 Example 2

```
<file-sweep:TemplateFile>  
  <jsdl:FileName>data2.dat</jsdl:FileName>  
  <jsdl:FileSystemName>DATADIR</jsdl:FileSystemName>  
</file-sweep:TemplateFile>
```

In this example, the file 'data2.dat' is relative to a file system. A `jsdl:FileSystem` element with the name 'DATADIR' must be defined in the enclosing JSDL job template.

### 3.2.4 FileToken

FileToken elements declare which tokens within a template file should be replaced by the Sweep Function values. Tokens within template files therefore represent placeholder parameters.

Since replacing values within text files can potentially be a very harmful behaviour, a token within a file *MUST* be matched *exactly* with a corresponding token value defined in the `/file-sweep:FileToken/@value` attribute (space and case sensitive). Tokens that exist within a template file which do not have a corresponding FileToken element declaration are left unmodified.

FileToken element values declared within the same FileSweep element *MUST* be unique. A File Sweep defined in that way is invalid and *MUST* be rejected (see 7.5 for an example).

#### 3.2.4.1 XML Representation

The FileToken element is a complex type and must support the `file-sweep:value` and `file-sweep:assignDefault` attributes. The FileToken element is rendered in XML as:

```
<file-sweep:FileToken value="xsd:string" assignDefault="xsd:string"? />
```

Where:

##### **/file-sweep:FileToken/@value**

Represents a unique token value declaration for the enclosing FileSweep element. Corresponding token values should exist within a template file that is also declared within the enclosing FileSweep element. Its XML type is defined as `xsd:string`. The value attribute is **MANDATORY**.

##### **/file-sweep:FileToken/@assignDefault**

Represents a default value that is assigned to the template file token. If specified, the default value overrides the Sweep Function value assignment. Its XML type is defined as `xsd:string`. The value attribute is **OPTIONAL**.

#### 3.2.4.2 Example 1

The following examples illustrate the use of the FileToken element.

```
<file-sweep:FileToken value="AOA" />
```

In this example, the token declaration 'AOA' selects corresponding 'AOA' tokens within a template file as the target for Parameter Sweep Function values.

#### 3.2.4.3 Example 2

```
<file-sweep:FileToken value="TE" assignDefault="6.0" />
```

In this example, the token declaration 'TE' selects corresponding 'TE' tokens within a template file. A default value is also specified which always overrides the Parameter Sweep Function value assignment.



#### 4. Normative Function substitute definitions

This chapter describes the default functions defined for the parameter Sweep specification. These functions are extension functions to the abstract Function element defined earlier in this document.

Each substitute described in this section, when composed with the base Parameter Sweep elements, is an instance of the abstract Function element. In an XML rendering, the Function element itself does not appear. The XML rendering of the selected extension function appears instead. For each appearance of the Function element an extension function can be used instead, either one of the default functions, or a self-defined extension function in a different namespace.

The functions described in this section **MUST** be supported by implementations of this specification.

##### 4.1 Values function

The Values function provides an ordered list of otherwise unconnected, or independent values. When iterated over, the Values function returns the first element, then the second, etc. until it returns the last element.

A Values functions has one or more child Value elements. Each Value child element is returned once and only once in a full iteration over a Values function. However, any two Value child elements **MAY** provide identical values. Within the scope of the parent Values function, all Value element **MUST** have the same data type.

##### 4.1.1 XML Representation

The Values function is rendered in XML as:

```
<sweepfunc:Values substitutes="sweep:Function">
  <sweepfunc:Value/>+
</sweepfunc:Values>
```

Where:

##### **/sweepfunc:Values**

Represents the Values element. It **MUST** contain at least one Value child element. This element **MAY** replace the sweep:Function element in an XML representation of the enclosing Parameter Sweep.

##### **/sweepfunc:Values/sweepfunc:Value**

Represents the Value element. It **MUST** be present at least once. It **MAY** be present more than once. Its XML data type is `xs:anyType`. However, Any two `sweepfunc:Value` elements that are children of the same Values element **MUST** have the same data type.

##### 4.1.2 Example

The following example illustrates the usage of the Values function:

```
<sweepfunc:Values>
  <sweepfunc:Value> The </sweepFunc:Value>
  <sweepfunc:Value> quick </sweepFunc:Value>
  <sweepfunc:Value> brown </sweepFunc:Value>
  <sweepfunc:Value> brown </sweepFunc:Value>
```

```
<sweepfunc:Value> fox </sweepFunc:Value>
<sweepfunc:Value> jumps </sweepFunc:Value>
<sweepfunc:Value> over </sweepFunc:Value>
<sweepfunc:Value> the </sweepFunc:Value>
<sweepfunc:Value> lazy </sweepFunc:Value>
<sweepfunc:Value> dog </sweepFunc:Value>
<sweepfunc:Values>
```

In this example, the Values function yields nine values, which, if concatenated using whitespace, would form the string "The quick brown fox jumps over the lazy dog".

## 4.2 LoopInteger function

The LoopInteger function provides an ordered list of integer values that are calculated based on the previous value beginning with the start value and ending with a final value smaller or equal to the end value.

A LoopInteger function MAY exclude certain values by including Exception Child elements.

This function yields values that are calculated as follows. The first value equals the value of the start value. Any subsequent value is calculated as using the following rule:

$$\text{nextValue} ::= \text{currentValue} + \text{valueOf}(\text{step})$$

If "nextValue" exceeds the value of the end value, then "nextValue" is discarded and the LoopInteger function has reached its end with "currentValue". If a value for "step" is not given, the default value of "1" MUST be used.

If the LoopInteger function contains Exception elements, then the "nextValue" is not returned if "nextValue" equals the value of any of the Exception child elements. Instead, "nextValue" becomes "currentValue", and the new "nextValue" is calculated and tested against all Exception children until no Exception element has the same value as "nextValue". In other words, the Exception child elements act as limiting filters on the list of values returned by the LoopInteger function.

The permissible value range for xsd:integer is defined as being the infinite set {...,-2,-1,0,1,2,...} with a lexical representation consisting of a finite-length sequence of decimal digits with an optional leading sign, e.g. -1, 0, 12678967543233, +100000.

### 4.2.1 XML Representation

The LoopInteger information element is rendered in XML as:

```
<sweepfunc:LoopInteger substitutes="sweep:Function"
    start="xsd:integer"
    end="xsd:integer"
    step="xsd:integer"? >
  <sweepfunc:Exception> xsd:integer </sweepfunc:Exception>*
</sweepfunc:LoopInteger>
```

Where:

#### **/sweepfunc:LoopInteger**

Represents the LoopInteger function. Its XML type is defined as `xsd:complexType` with no content other than XML elements and attributes. This element MAY replace the `sweep:Function` element in an XML representation of the Parameter Sweep.

**/sweepfunc:LoopInteger/@start**

Represents the start attribute of the LoopInteger element. Its XML type is defined as `xsd:integer`. The start attribute is MANDATORY.

**/sweepfunc:LoopInteger/@end**

Represents the end attribute of the LoopInteger element. Its XML type is defined as `xsd:integer`. The end attribute is MANDATORY.

**/sweepfunc:LoopInteger/@step**

Represents the step attribute of the LoopInteger element. Its XML type is defined as `xsd:integer`. The step attribute is OPTIONAL. If no value is defined (i.e. this attribute is not present in the XML representation), then a default value of 1 is assumed for this attribute.

**/sweepfunc:LoopInteger/sweepfunc:Exception**

Represents the Exception child element. Its XML type is defined as `xsd:integer`. Exception elements are OPTIONAL; if no Exception element is given, then the LoopInteger function yields all calculated values as specified above.

4.2.2 Example 1

The following example illustrates the usage of the LoopInteger function:

```
<sweepfunc:LoopInteger start="1" end="10">  
</sweepfunc:LoopInteger>
```

In this example, the LoopInteger function yields the ten values "1", "2", "3", etc. up to "10".

4.2.3 Example 2

The following example illustrates the usage of the LoopInteger function:

```
<sweepfunc:LoopInteger start="1" end="10">  
  <sweepfunc:Exception> 5 </sweepfunc:Exception>  
  <sweepfunc:Exception> 7 </sweepfunc:Exception>  
</sweepfunc:LoopInteger>
```

In this example, the LoopInteger function yields eight values: "1", "2", "3", "4", "6", "8", "9", "10".

4.2.4 Example 3

The following example illustrates the usage of the LoopInteger function:

```
<sweepfunc:LoopInteger start="1" end="10" step="2">  
  <sweepfunc:Exception> 7 </sweepfunc:Exception>  
</sweepfunc:LoopInteger>
```

In this example, the LoopInteger function yields four values: "1", "3", "5", and "9".

4.3 LoopDouble function

The LoopDouble function provides an ordered list of double datatype values (corresponding to the IEEE double-precision 64-bit floating point type) that are calculated based on the previous value, beginning with the start value, and ending with a final value smaller or equal to the end value.

It behaves in a manner, which is functionally identical to the LoopInteger function with the exception of the step attribute which, for LoopDouble, is defined as MANDATORY with no default value.

The permissible value range for `xsd:double` is defined as consisting of "... the values  $m \times 2^e$ , where  $m$  is an integer whose absolute value is less than  $2^{53}$ , and  $e$  is an integer between -1075 and 970, inclusive"<sup>4</sup> In addition the following special values are permissible: positive and negative zero, positive and negative infinity and not-a-number. Lexically, scientific notation may be used consisting of a sequence of a decimal number mantissa value, optionally followed by an 'e' or 'E' character, followed by an integer exponent value. The following are therefore considered valid `xsd:decimal` values : -1E4, 1267.43233E12, 12.78e-2, 12, NaN, -0 and INF .

#### 4.3.1 XML Representation

The LoopDouble information element is rendered in XML as:

```
<sweepfunc:LoopDouble substitutes="sweep:Function"  
    start="xsd:double"  
    end="xsd:double"  
    step="xsd:double" >  
    <sweepfunc:Exception> xsd:double </sweepfunc:Exception>  
</sweepfunc:LoopDouble>
```

Where:

##### **/sweepfunc:LoopDouble**

Represents the LoopDouble element. Its XML type is defined as `xsd:complexType` with no content other than XML elements and attributes. This element MAY replace the `sweep:Function` element in an XML representation of the Parameter Sweep.

##### **/sweepfunc:LoopDouble/@start**

Represents the start attribute of the LoopDouble element. Its XML type is defined as `xsd:double`. The start attribute is MANDATORY.

##### **/sweepfunc:LoopDouble/@end**

Represents the end attribute of the LoopDouble element. Its XML type is defined as `xsd:double`. The end attribute is MANDATORY.

##### **/sweepfunc:LoopDouble/@step**

Represents the step attribute of the LoopDouble element. Its XML type is defined as `xsd:double`. The step attribute is MANDATORY.

##### **/sweepfunc:LoopDouble/sweep:Exception**

Represents the Exception child element. Its XML type is defined as `xsd:double`. Exception elements are OPTIONAL; if no Exception element is given, then the LoopDouble function yields all calculated values as specified above.

#### 4.3.2 Example 1

The following example illustrates the usage of the LoopDouble function:

```
<sweepfunc:LoopDouble start="-1e-4" end="-10e-4" step="-1e-4">
```

<sup>4</sup> <http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/#double>  
jsdl-wg@ogf.org

```
</sweepfunc:LoopDouble>
```

In this example, the LoopDouble function yields the ten values “-1e-4”, “-2e-4”, “-3e-4”, etc. down to “-10e-4”.

### 4.3.3 Example 2

The following example illustrates the usage of the LoopDouble function:

```
<sweepfunc:LoopDouble start="1000.0" end="1400.0" step="50.0">  
  <sweepfunc:Exception>1000.0</sweepfunc:Exception>  
  <sweepfunc:Exception>1100.0</sweepfunc:Exception>  
</sweepfunc:LoopDouble>
```

In this example, the LoopDouble function yields the seven values "1050.0", "1150.0", "1200.0", "1250.0", "1300.0", "1350.0", "1400.0".

## 5. Integration with JSDL v1.0

The information set described in this specification defines, in a compressed format, the individual jobs that span the Parameter Sweep. In JSDL v1.0, the element `jsdl:JobDefinition` has only one defined child; `jsdl:JobDescription`, except providing extension points for both XML attributes and elements. While this is mostly for historical reasons, it is valid to treat the `jsdl:JobDescription` element as the real JSDL Job Template defining the nuts and bolts of the job that needs to be executed, and use the `jsdl:JobDefinition` element as a scoping closure to anything a JSDL Job Template is composed with.

Hence the most appropriate way of composing a JSDL Job Template with a Parameter Sweep definition is to mix-in the Parameter Sweep as a direct child element of the `jsdl:JobDefinition` element.

Having said that, a Parameter Sweep declaration **MUST** be given as follows:

```
<jsdl:JobDefinition>  
  <jsdl:JobDescription/>  
  <sweep:Sweep/>+  
</jsdl:JobDefinition>
```

## 6. Examples

The following examples illustrate the use of the defined JSDL extensions; the information given in this section is purely informational.

All examples given in this section use the same JSDL Job template. This template, when executed, invokes the POSIX application “/usr/bin/echo” with a finite set of command line arguments. When executed correctly, the application would echo the string “The quick brown fox jumps over the lazy dog”. When evaluated using the Parameter Sweep extension as defined in this document, the result will be different.

The first example is fully worked out, i.e. the JSDL Job template is given including the Parameter Sweep elements, and the resulting set of individual jobs are given that constitute the intended Parameter Sweep. Along with each individual job template its expected outcome is given, too.

The following examples use the same JSDL Job template as the first example but illustrate

different use cases for the Parameter Sweep specification. These examples are not worked out as the first example. Instead, only the expected outcome is given for each individual job of the Array Job definition.

As a notational convention, differences compared to the initial JSDL Job template are given in *underlined italic*.

## 6.1 Example 1

This example illustrates the basic use of the Parameter Sweep specification as described in this document.

First the submitted JSDL Job Template is shown. Then the resultant individual jobs are given, together with their expected outcome.

### 6.1.1 Submitted JSDL Job template

The following is the JSDL Job template composed with the Parameter Sweep element, in XML representation:

```
<?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:sweep="http://schemas.ogf.org/jSDL/2007/04/sweep"
  xmlns:sweepfunc="http://schemas.ogf.org/jSDL/2007/04/sweep/functions">
  <jSDL:JobDescription>
    <jSDL:Application>
      <jSDL-posix:POSIXApplication>
        <jSDL-posix:Executable>/bin/echo</jSDL-
posix:Executable>
        <jSDL-posix:Argument>The</jSDL-posix:Argument>
        <jSDL-posix:Argument>quick</jSDL-posix:Argument>
        <jSDL-posix:Argument>brown</jSDL-posix:Argument>
        <jSDL-posix:Argument>fox</jSDL-posix:Argument>
        <jSDL-posix:Argument>jumps</jSDL-posix:Argument>
        <jSDL-posix:Argument>over</jSDL-posix:Argument>
        <jSDL-posix:Argument>the</jSDL-posix:Argument>
        <jSDL-posix:Argument>lazy</jSDL-posix:Argument>
        <jSDL-posix:Argument>dog</jSDL-posix:Argument>
      </jSDL-posix:POSIXApplication>
    </jSDL:Application>
  </jSDL:JobDescription>
  <sweep:Sweep>
    <sweep:Assignment>
      <sweep:DocumentNode>
        <sweep:NamespaceBinding
          ns="http://schemas.ggf.org/jSDL/2005/11/jSDL-posix"
          prefix="jSDL-posix" />
        <sweep:Match>
          //jSDL-posix:Argument[4]
        </sweep:Match>
      </sweep:DocumentNode>
    </sweep:Assignment>
  </sweep:Sweep>
</jSDL:JobDefinition>
```

```
        <sweepfunc:Values>
          <sweepfunc:Value>cat</sweepfunc:Value>
          <sweepfunc:Value>dog</sweepfunc:Value>
          <sweepfunc:Value>bird</sweepfunc:Value>
        </sweepfunc:Values>
      </sweep:Assignment>
    </sweep:Sweep>
  </jsdl:JobDefinition>
```

For brevity reasons the XPath 2.0 expressions are kept very short. It suits the use for this specific document only; in real life applications the DocumentNode expressions are expected to be of much more complexity.

### 6.1.2 Parameter Sweep individual job templates

The following XML representations give each individual job template that altogether constitute the Parameter Sweep submitted as given above.

The **first** individual job template would be:

```
<?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:sweep="http://schemas.ogf.org/jsdl/2007/04/sweep"
  xmlns:sweepfunc="http://schemas.ogf.org/jsdl/2007/04/sweep/functions">
  <jsdl:JobDescription>
    <jsdl:Application>
      <jsdl-posix:POSIXApplication>
        <jsdl-posix:Executable>/bin/echo</jsdl-
posix:Executable>
        <jsdl-posix:Argument>The</jsdl-posix:Argument>
        <jsdl-posix:Argument>quick</jsdl-posix:Argument>
        <jsdl-posix:Argument>brown</jsdl-posix:Argument>
        <jsdl-posix:Argument>cat</jsdl-posix:Argument>
        <jsdl-posix:Argument>jumps</jsdl-posix:Argument>
        <jsdl-posix:Argument>over</jsdl-posix:Argument>
        <jsdl-posix:Argument>the</jsdl-posix:Argument>
        <jsdl-posix:Argument>lazy</jsdl-posix:Argument>
        <jsdl-posix:Argument>dog</jsdl-posix:Argument>
      </jsdl-posix:POSIXApplication>
    </jsdl:Application>
  </jsdl:JobDescription>
</jsdl:JobDefinition>
```

The expected outcome is "The quick brown cat jumps over the lazy dog".

The **second** individual job template would be:

```
<?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:sweep="http://schemas.ogf.org/jsdl/2007/04/sweep"

xmlns:sweepfunc="http://schemas.ogf.org/jsdl/2007/04/sweep/functions">

  <jsdl:JobDescription>
    <jsdl:Application>
      <jsdl-posix:POSIXApplication>
        <jsdl-posix:Executable>/bin/echo</jsdl-
posix:Executable>
        <jsdl-posix:Argument>The</jsdl-posix:Argument>
        <jsdl-posix:Argument>quick</jsdl-posix:Argument>
        <jsdl-posix:Argument>brown</jsdl-posix:Argument>
        <jsdl-posix:Argument>dog</jsdl-posix:Argument>
        <jsdl-posix:Argument>jumps</jsdl-posix:Argument>
        <jsdl-posix:Argument>over</jsdl-posix:Argument>
        <jsdl-posix:Argument>the</jsdl-posix:Argument>
        <jsdl-posix:Argument>lazy</jsdl-posix:Argument>
        <jsdl-posix:Argument>dog</jsdl-posix:Argument>
      </jsdl-posix:POSIXApplication>
    </jsdl:Application>
  </jsdl:JobDescription>

</jsdl:JobDefinition>
```

The expected outcome is "The quick brown dog jumps over the lazy dog".

The **third** individual job template would be:

```
<?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:sweep="http://schemas.ogf.org/jsdl/2007/04/sweep"

  xmlns:sweepfunc="http://schemas.ogf.org/jsdl/2007/04/sweep/functions">

  <jsdl:JobDescription>
    <jsdl:Application>
      <jsdl-posix:POSIXApplication>
        <jsdl-posix:Executable>/bin/echo</jsdl-
posix:Executable>
        <jsdl-posix:Argument>The</jsdl-posix:Argument>
        <jsdl-posix:Argument>quick</jsdl-posix:Argument>
        <jsdl-posix:Argument>brown</jsdl-posix:Argument>
        <jsdl-posix:Argument>bird</jsdl-posix:Argument>
        <jsdl-posix:Argument>jumps</jsdl-posix:Argument>
        <jsdl-posix:Argument>over</jsdl-posix:Argument>
        <jsdl-posix:Argument>the</jsdl-posix:Argument>
        <jsdl-posix:Argument>lazy</jsdl-posix:Argument>
        <jsdl-posix:Argument>dog</jsdl-posix:Argument>
      </jsdl-posix:POSIXApplication>
    </jsdl:Application>
  </jsdl:JobDescription>

</jsdl:JobDefinition>
```



The expected outcome is "The quick brown bird jumps over the lazy dog".

Hence the Parameter Sweep specified in section 6.1.1 yields three individual jobs spanning the one-dimensional parameter space {cat, dog, bird}.

## 6.2 Example 2

This example is a variation of example one in that the parameter space is still one-dimensional: {cat, dog, bird} but its elements are assigned to two arguments to the POSIX application as follows:

```
<?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:sweep="http://schemas.ggf.org/jsdl/2007/04/sweep"
  xmlns:sweepfunc="http://schemas.ggf.org/jsdl/2007/04/sweep/functions">
  <jsdl:JobDescription>
    <jsdl:Application>
      <jsdl-posix:POSIXApplication>
        <jsdl-posix:Executable>/bin/echo</jsdl-
posix:Executable>
        <jsdl-posix:Argument>The</jsdl-posix:Argument>
        <jsdl-posix:Argument>quick</jsdl-posix:Argument>
        <jsdl-posix:Argument>brown</jsdl-posix:Argument>
        <jsdl-posix:Argument>fox</jsdl-posix:Argument>
        <jsdl-posix:Argument>jumps</jsdl-posix:Argument>
        <jsdl-posix:Argument>over</jsdl-posix:Argument>
        <jsdl-posix:Argument>the</jsdl-posix:Argument>
        <jsdl-posix:Argument>lazy</jsdl-posix:Argument>
        <jsdl-posix:Argument>dog</jsdl-posix:Argument>
      </jsdl-posix:POSIXApplication>
    </jsdl:Application>
  </jsdl:JobDescription>
  <sweep:Sweep>
    <sweep:Assignment>
      <sweep:DocumentNode>
        <sweep:NamespaceBinding
          ns="http://schemas.ggf.org/jsdl/2005/11/jsdl-posix"
          prefix="jsdl-posix" />
        <sweep:Match>
          //jsdl-posix:Argument[4]
        </sweep:Match>
      </sweep:DocumentNode>
      <sweep:DocumentNode>
        <sweep:NamespaceBinding
          ns="http://schemas.ggf.org/jsdl/2005/11/jsdl-posix"
          prefix="jsdl-posix" />
        <sweep:Match>
          //jsdl-posix:Argument[9]
        </sweep:Match>
      </sweep:DocumentNode>
    </sweep:Assignment>
  </sweep:Sweep>
</jsdl:JobDefinition>
```

```

        <sweepfunc:Values>
            <sweepfunc:Value>cat</sweepfunc:Value>
            <sweepfunc:Value>dog</sweepfunc:Value>
            <sweepfunc:Value>bird</sweepfunc:Value>
        </sweepfunc:Values>
    </sweep:Assignment>
</sweep:Sweep>
</jsdl:JobDefinition>

```

The Array Job contains again three individual jobs. The expected outcomes of those three jobs would be:

- "The quick brown cat jumps over the lazy cat"
- "The quick brown dog jumps over the lazy dog"
- "The quick brown bird jumps over the lazy bird"

### 6.3 Example 3

This example illustrates how to assign two independent parameter spaces at the same time in a Parameter Sweep. This example performs a two-dimensional parameter sweep, using the following parameter value pairs: {(black,cat), (grey,dog), blue,bird) }.

```

<?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:sweep="http://schemas.ogf.org/jsdl/2007/04/sweep"
  xmlns:sweepfunc="http://schemas.ogf.org/jsdl/2007/04/sweep/functions">
  <jsdl:JobDescription>
    <jsdl:Application>
      <jsdl-posix:POSIXApplication>
        <jsdl-posix:Executable>/bin/echo</jsdl-
posix:Executable>
        <jsdl-posix:Argument>The</jsdl-posix:Argument>
        <jsdl-posix:Argument>quick</jsdl-posix:Argument>
        <jsdl-posix:Argument>brown</jsdl-posix:Argument>
        <jsdl-posix:Argument>fox</jsdl-posix:Argument>
        <jsdl-posix:Argument>jumps</jsdl-posix:Argument>
        <jsdl-posix:Argument>over</jsdl-posix:Argument>
        <jsdl-posix:Argument>the</jsdl-posix:Argument>
        <jsdl-posix:Argument>lazy</jsdl-posix:Argument>
        <jsdl-posix:Argument>dog</jsdl-posix:Argument>
      </jsdl-posix:POSIXApplication>
    </jsdl:Application>
  </jsdl:JobDescription>
  <sweep:Sweep>
    <sweep:Assignment>
      <sweep:DocumentNode>
        <sweep:NamespaceBinding
          ns="http://schemas.ggf.org/jsdl/2005/11/jsdl-posix"
          prefix="jsdl-posix" />
        <sweep:Match>

```

```

        //jsdl-posix:Argument[3]
    </sweep:Match>
</sweep:DocumentNode>
<sweepfunc:Values>
    <sweepfunc:Value>black</sweepfunc:Value>
    <sweepfunc:Value>grey</sweepfunc:Value>
    <sweepfunc:Value>blue</sweepfunc:Value>
</sweepfunc:Values>
</sweep:Assignment>
<sweep:Assignment>
    <sweep:DocumentNode>
        <sweep:NamespaceBinding
            ns="http://schemas.ggf.org/jsdl/2005/11/jsdl-posix"
            prefix="jsdl-posix" />
        <sweep:Match>
            //jsdl-posix:Argument[4]
        </sweep:Match>
    </sweep:DocumentNode>
    <sweepfunc:Values>
        <sweepfunc:Value>cat</sweepfunc:Value>
        <sweepfunc:Value>dog</sweepfunc:Value>
        <sweepfunc:Value>bird</sweepfunc:Value>
    </sweepfunc:Values>
</sweep:Assignment>
</sweep:Sweep>
</jsdl:JobDefinition>

```

The Array Job contains again three individual jobs. The expected outcomes of those three jobs would be:

- “The quick *black cat* jumps over the lay dog”
- “The quick *grey dog* jumps over the lay dog”
- “The quick *blue bird* jumps over the lay dog”

#### 6.4 Example 4

This example illustrates how to assign two otherwise independent parameter spaces to two different Parameter elements.

```

<?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:sweep="http://schemas.ogf.org/jsdl/2007/04/sweep"
    xmlns:sweepfunc="http://schemas.ogf.org/jsdl/2007/04/sweep/functions">
    <jsdl:JobDescription>
        <jsdl:Application>
            <jsdl-posix:POSIXApplication>
                <jsdl-posix:Executable>/bin/echo</jsdl-
posix:Executable>
                <jsdl-posix:Argument>The</jsdl-posix:Argument>
                <jsdl-posix:Argument>quick</jsdl-posix:Argument>
                <jsdl-posix:Argument>brown</jsdl-posix:Argument>
                <jsdl-posix:Argument>fox</jsdl-posix:Argument>

```

```
        <jSDL-posix:Argument>jumps</jSDL-posix:Argument>
        <jSDL-posix:Argument>over</jSDL-posix:Argument>
        <jSDL-posix:Argument>the</jSDL-posix:Argument>
        <jSDL-posix:Argument>lazy</jSDL-posix:Argument>
        <jSDL-posix:Argument>dog</jSDL-posix:Argument>
    </jSDL-posix:POSIXApplication>
</jSDL:Application>
</jSDL:JobDescription>

<sweep:Sweep>
  <sweep:Assignment>
    <sweep:DocumentNode>
      <sweep:NamespaceBinding
        ns="http://schemas.ggf.org/jSDL/2005/11/jSDL-posix"
        prefix="jSDL-posix" />
      <sweep:Match>
        //jSDL-posix:Argument[3]
      </sweep:Match>
    </sweep:DocumentNode>
    <sweepfunc:Values>
      <sweepfunc:Value>black</sweepfunc:Value>
      <sweepfunc:Value>grey</sweepfunc:Value>
      <sweepfunc:Value>blue</sweepfunc:Value>
    </sweepfunc:Values>
  </sweep:Assignment>
</sweep:Sweep>

<sweep:Sweep>
  <sweep:Assignment>
    <sweep:DocumentNode>
      <sweep:NamespaceBinding
        ns="http://schemas.ggf.org/jSDL/2005/11/jSDL-posix"
        prefix="jSDL-posix" />
      <sweep:Match>
        //jSDL-posix:Argument[4]
      </sweep:Match>
    </sweep:DocumentNode>
    <sweepfunc:Values>
      <sweepfunc:Value>cat</sweepfunc:Value>
      <sweepfunc:Value>dog</sweepfunc:Value>
      <sweepfunc:Value>bird</sweepfunc:Value>
    </sweepfunc:Values>
  </sweep:Assignment>
</sweep:Sweep>
</jSDL:JobDefinition>
```

The Array Job contains six individual jobs. The expected outcomes of those six jobs would be:

- “The quick *black* fox jumps over the lay dog”
- “The quick *grey* fox jumps over the lay dog”
- “The quick *blue* fox jumps over the lay dog”
- “The quick brown *cat* jumps over the lay dog”
- “The quick brown *dog* jumps over the lay dog”
- “The quick brown *bird* jumps over the lay dog”

## 6.5 Example 5

This example illustrates the scoping of the Parameter elements. It is very similar to example 4, except that in both Sweep elements, the same XML element is selected in the Parameter element. This is valid because the context of the two Parameter elements is different:

```
<?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:sweep="http://schemas.ogf.org/jsdl/2007/04/sweep"

  xmlns:sweepfunc="http://schemas.ogf.org/jsdl/2007/04/sweep/functions">

  <jsdl:JobDescription>
    <jsdl:Application>
      <jsdl-posix:POSIXApplication>
        <jsdl-posix:Executable>/bin/echo</jsdl-
posix:Executable>
        <jsdl-posix:Argument>The</jsdl-posix:Argument>
        <jsdl-posix:Argument>quick</jsdl-posix:Argument>
        <jsdl-posix:Argument>brown</jsdl-posix:Argument>
        <jsdl-posix:Argument>fox</jsdl-posix:Argument>
        <jsdl-posix:Argument>jumps</jsdl-posix:Argument>
        <jsdl-posix:Argument>over</jsdl-posix:Argument>
        <jsdl-posix:Argument>the</jsdl-posix:Argument>
        <jsdl-posix:Argument>lazy</jsdl-posix:Argument>
        <jsdl-posix:Argument>dog</jsdl-posix:Argument>
      </jsdl-posix:POSIXApplication>
    </jsdl:Application>
  </jsdl:JobDescription>

  <sweep:Sweep>
    <sweep:Assignment>
      <sweep:DocumentNode>
        <sweep:NamespaceBinding
          ns="http://schemas.ggf.org/jsdl/2005/11/jsdl-posix"
          prefix="jsdl-posix" />
        <sweep:Match>
          //jsdl-posix:Argument[4]
        </sweep:Match>
      </sweep:DocumentNode>
      <sweepfunc:Values>
        <sweepfunc:Value>spider</sweepfunc:Value>
        <sweepfunc:Value>ant</sweepfunc:Value>
        <sweepfunc:Value>butterfly</sweepfunc:Value>
      </sweepfunc:Values>
    </sweep:Assignment>
  </sweep:Sweep>

  <sweep:Sweep>
    <sweep:Assignment>
      <sweep:DocumentNode>
        <sweep:NamespaceBinding
          ns="http://schemas.ggf.org/jsdl/2005/11/jsdl-posix"
          prefix="jsdl-posix" />
        <sweep:Match>
```

```

        //jsdl-posix:Argument[4]
    </sweep:Match>
</sweep:DocumentNode>
<sweepfunc:Values>
    <sweepfunc:Value>cat</sweepfunc:Value>
    <sweepfunc:Value>dog</sweepfunc:Value>
    <sweepfunc:Value>bird</sweepfunc:Value>
</sweepfunc:Values>
</sweep:Assignment>
</sweep:Sweep>
</jsdl:JobDefinition>

```

The Array Job contains six individual jobs. The expected outcomes of those six jobs would be:

- “The quick brown *spider* jumps over the lay dog”
- “The quick brown *ant* jumps over the lay dog”
- “The quick brown *butterfly* jumps over the lay dog”
- “The quick brown *cat* jumps over the lay dog”
- “The quick brown *dog* jumps over the lay dog”
- “The quick brown *bird* jumps over the lay dog”

## 6.6 Example 6

This example illustrates the use of nested Sweep elements. It performs a two-dimensional parameter sweep over the parameter spaces {black, grey, blue} and {cat, dog, bird}. The resulting Array Job covers the complete set of permutations over the two parameter spaces. The submitted JSDL Job template looks as follows.

```

<?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:sweep="http://schemas.ogf.org/jsdl/2007/04/sweep"
  xmlns:sweepfunc="http://schemas.ogf.org/jsdl/2007/04/sweep/functions">
  <jsdl:JobDescription>
    <jsdl:Application>
      <jsdl-posix:POSIXApplication>
        <jsdl-posix:Executable>/bin/echo</jsdl-
posix:Executable>
        <jsdl-posix:Argument>The</jsdl-posix:Argument>
        <jsdl-posix:Argument>quick</jsdl-posix:Argument>
        <jsdl-posix:Argument>brown</jsdl-posix:Argument>
        <jsdl-posix:Argument>fox</jsdl-posix:Argument>
        <jsdl-posix:Argument>jumps</jsdl-posix:Argument>
        <jsdl-posix:Argument>over</jsdl-posix:Argument>
        <jsdl-posix:Argument>the</jsdl-posix:Argument>
        <jsdl-posix:Argument>lazy</jsdl-posix:Argument>
        <jsdl-posix:Argument>dog</jsdl-posix:Argument>
      </jsdl-posix:POSIXApplication>
    </jsdl:Application>
  </jsdl:JobDescription>
  <sweep:Sweep>

```

```
<sweep:Assignment>
  <sweep:DocumentNode>
    <sweep:NamespaceBinding
      ns="http://schemas.ggf.org/jsdl/2005/11/jsdl-posix"
      prefix="jsdl-posix" />
    <sweep:Match>
      //jsdl-posix:Argument[3]
    </sweep:Match>
  </sweep:DocumentNode>
  <sweepfunc:Values>
    <sweepfunc:Value>black</sweepfunc:Value>
    <sweepfunc:Value>grey</sweepfunc:Value>
    <sweepfunc:Value>blue</sweepfunc:Value>
  </sweepfunc:Values>
</sweep:Assignment>
<sweep:Sweep>
  <sweep:Assignment>
    <sweep:DocumentNode>
      <sweep:NamespaceBinding
        ns="http://schemas.ggf.org/jsdl/2005/11/jsdl-posix"
        prefix="jsdl-posix" />
      <sweep:Match>
        //jsdl-posix:Argument[4]
      </sweep:Match>
    </sweep:DocumentNode>
    <sweepfunc:Values>
      <sweepfunc:Value>cat</sweepfunc:Value>
      <sweepfunc:Value>dog</sweepfunc:Value>
      <sweepfunc:Value>bird</sweepfunc:Value>
    </sweepfunc:Values>
  </sweep:Assignment>
</sweep:Sweep>
</sweep:Sweep>
</jsdl:JobDefinition>
```

The Array Job contains nine individual jobs. The expected outcomes of those nine jobs would be:

- “The quick *black cat* jumps over the lay dog”
- “The quick *black dog* jumps over the lay dog”
- “The quick *black bird* jumps over the lay dog”
- “The quick *grey cat* jumps over the lay dog”
- “The quick *grey dog* jumps over the lay dog”
- “The quick *grey bird* jumps over the lay dog”
- “The quick *blue cat* jumps over the lay dog”
- “The quick *blue dog* jumps over the lay dog”
- “The quick *blue bird* jumps over the lay dog”

## 6.7 Example 7

This example illustrates illegal scoping of the DocumentNode element. It is illegal, because the two infringing DocumentNode elements reside within the same Sweep element even though they are children of different Assignment elements. The incriminating portions of the DocumentNode elements are given in bold red below.

```
<?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:sweep="http://schemas.ogf.org/jsdl/2007/04/sweep"
  xmlns:sweepfunc="http://schemas.ogf.org/jsdl/2007/04/sweep/functions">

  <jsdl:JobDescription>
    <jsdl:Application>
      <jsdl-posix:POSIXApplication>
        <jsdl-posix:Executable>/bin/echo</jsdl-
posix:Executable>
        <jsdl-posix:Argument>The</jsdl-posix:Argument>
        <jsdl-posix:Argument>quick</jsdl-posix:Argument>
        <jsdl-posix:Argument>brown</jsdl-posix:Argument>
        <jsdl-posix:Argument>fox</jsdl-posix:Argument>
        <jsdl-posix:Argument>jumps</jsdl-posix:Argument>
        <jsdl-posix:Argument>over</jsdl-posix:Argument>
        <jsdl-posix:Argument>the</jsdl-posix:Argument>
        <jsdl-posix:Argument>lazy</jsdl-posix:Argument>
        <jsdl-posix:Argument>dog</jsdl-posix:Argument>
      </jsdl-posix:POSIXApplication>
    </jsdl:Application>
  </jsdl:JobDescription>

  <sweep:Sweep>
    <sweep:Assignment>
      <sweep:DocumentNode>
        <sweep:NamespaceBinding
          ns="http://schemas.ggf.org/jsdl/2005/11/jsdl-posix"
          prefix="jsdl-posix" />
        <sweep:Match>
          //jsdl-posix:Argument[3]
        </sweep:Match>
      </sweep:DocumentNode>
      <sweepfunc:Values>
        <sweepfunc:Value>black</sweepfunc:Value>
        <sweepfunc:Value>grey</sweepfunc:Value>
        <sweepfunc:Value>blue</sweepfunc:Value>
      </sweepfunc:Values>
    </sweep:Assignment>
    <sweep:Assignment>
      <sweep:DocumentNode>
        <sweep:NamespaceBinding
          ns="http://schemas.ggf.org/jsdl/2005/11/jsdl-posix"
          prefix="jsdl-posix" />
        <sweep:Match>
          //jsdl-posix:Argument[3]
        </sweep:Match>
      </sweep:DocumentNode>
      <sweepfunc:Values>
        <sweepfunc:Value>cat</sweepfunc:Value>
        <sweepfunc:Value>dog</sweepfunc:Value>
        <sweepfunc:Value>bird</sweepfunc:Value>
      </sweepfunc:Values>
    </sweep:Assignment>
  </sweep:Sweep>
</jsdl:JobDefinition>
```



```

    </sweep:Sweep>
</jsdl:JobDefinition>

```

## 6.8 Example 8

This example illustrates illegal scoping of the DocumentNode element. It is illegal because the two infringing DocumentNode elements reside in the same nesting Sweep element path and are hence sharing the same scope. The incriminating portions of the DocumentNode elements are given in bold red below.

```

<?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:sweep="http://schemas.ogf.org/jsdl/2007/04/sweep"
  xmlns:sweepfunc="http://schemas.ogf.org/jsdl/2007/04/sweep/functions">
  <jsdl:JobDescription>
    <jsdl:Application>
      <jsdl-posix:POSIXApplication>
        <jsdl-posix:Executable>/bin/echo</jsdl-
posix:Executable>
        <jsdl-posix:Argument>The</jsdl-posix:Argument>
        <jsdl-posix:Argument>quick</jsdl-posix:Argument>
        <jsdl-posix:Argument>brown</jsdl-posix:Argument>
        <jsdl-posix:Argument>fox</jsdl-posix:Argument>
        <jsdl-posix:Argument>jumps</jsdl-posix:Argument>
        <jsdl-posix:Argument>over</jsdl-posix:Argument>
        <jsdl-posix:Argument>the</jsdl-posix:Argument>
        <jsdl-posix:Argument>lazy</jsdl-posix:Argument>
        <jsdl-posix:Argument>dog</jsdl-posix:Argument>
      </jsdl-posix:POSIXApplication>
    </jsdl:Application>
  </jsdl:JobDescription>
  <sweep:Sweep>
    <sweep:Assignment>
      <sweep:DocumentNode>
        <sweep:NamespaceBinding
          ns="http://schemas.ggf.org/jsdl/2005/11/jsdl-posix"
          prefix="jsdl-posix" />
        <sweep:Match>
          //jsdl-posix:Argument[3]
        </sweep:Match>
      </sweep:DocumentNode>
      <sweepfunc:Values>
        <sweepfunc:Value>black</sweepfunc:Value>
        <sweepfunc:Value>grey</sweepfunc:Value>
        <sweepfunc:Value>blue</sweepfunc:Value>
      </sweepfunc:Values>
    </sweep:Assignment>
  </sweep:Sweep>
  <sweep:Assignment>
    <sweep:DocumentNode>
      <sweep:NamespaceBinding

```

```
        ns="http://schemas.ggf.org/jsdl/2005/11/jsdl-posix"  
        prefix="jsdl-posix" />  
        <sweep:Match>  
            //jsdl-posix:Argument[4]  
        </sweep:Match>  
    </sweep:DocumentNode>  
    <sweepfunc:Values>  
        <sweepfunc:Value>cat</sweepfunc:Value>  
        <sweepfunc:Value>dog</sweepfunc:Value>  
        <sweepfunc:Value>bird</sweepfunc:Value>  
    </sweepfunc:Values>  
    </sweep:Assignment>  
    </sweep:Sweep>  
    </sweep:Sweep>  
    </jsdl:JobDefinition>
```

## 7. FileSweep Examples

In this section, some more detailed examples of the file sweep information set are given that focus on the FileSweep Parameter substitute.

All examples given in this section use the same JSDL Job template. This template, when executed, invokes the POSIX application “/bin/cat” with a finite set of command line arguments. When executed correctly, the application outputs the modified contents of the selected template files to standard output.

All examples given in this section also use the same template files (data1.dat, data2.dat). The unmodified template file contents are shown below. Note that the ‘colour’ token is common in both template files.

### data1.dat file content

data1.dat: The quick colour animal.1a jumps over the lazy animal.1b

### data2.dat file content

data2.dat: The quick colour animal.2a jumps over the lazy animal.2b

The first example (7.1), lists both the JSDL Job template and the file sweep elements. The resulting output of the individual jobs that constitute the Parameter Sweep is also given.

The following examples (7.2 to 7.6) use the same JSDL Job template as the first example but illustrate different use cases for the file sweep information set. Only the file sweep information set and the expected outcome are given.

As a notational convention, differences between the modified template files are shown in *underlined italic*.

### 7.1 Example 1

In this example, the same set of values are assigned to tokens in one template file (data2.dat) with one default value assignment. data1.dat is left unmodified.

```
<?xml version="1.0" encoding="UTF-8"?>  
<jsdl:JobDefinition
```

```
  xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'  
xmlns:sweepfunc='http://schemas.ogf.org/jsdl/2007/04/sweep/functions'  
  xmlns:sweep='http://schemas.ogf.org/jsdl/2007/04/sweep'  
  xmlns:file-sweep='http://schemas.ogf.org/jsdl/2008/08/file-sweep'  
  xmlns:jsdl-posix='http://schemas.ogf.org/jsdl/2005/11/jsdl-posix'  
  xmlns:jsdl='http://schemas.ogf.org/jsdl/2005/11/jsdl'>  
  
  <jsdl:JobDescription>  
    <jsdl:Application>  
      <jsdl-posix:POSIXApplication>  
        <jsdl-posix:Executable>/bin/cat</jsdl-posix:Executable>  
        <jsdl-posix:Argument>data1.dat</jsdl-posix:Argument>  
        <jsdl-posix:Argument>/usr/data/data2.dat  
        </jsdl-posix:Argument>  
      </jsdl-posix:POSIXApplication>  
    </jsdl:Application>  
    <jsdl:Resources>  
      <jsdl:FileSystem name="DATADIR">  
        <jsdl:FileSystemType>normal</jsdl:FileSystemType>  
        <jsdl:Description>Data files</jsdl:Description>  
        <jsdl:MountPoint>/usr/data</jsdl:MountPoint>  
      </jsdl:FileSystem>  
    </jsdl:Resources>  
    <jsdl:DataStaging>  
      <jsdl:FileName>data1.dat</jsdl:FileName>  
      <jsdl:CreationFlag>overwrite</jsdl:CreationFlag>  
      <jsdl>DeleteOnTermination>>true</jsdl>DeleteOnTermination>  
      <jsdl:Source>  
        <jsdl:URI>gsiftp://host:port/dir/data1.dat</jsdl:URI>  
      </jsdl:Source>  
    </jsdl:DataStaging>  
    <jsdl:DataStaging>  
      <jsdl:FileName>data2.dat</jsdl:FileName>  
      <jsdl:FileSystemName>DATADIR</jsdl:FileSystemName>  
      <jsdl:CreationFlag>overwrite</jsdl:CreationFlag>  
      <jsdl>DeleteOnTermination>>true</jsdl>DeleteOnTermination>  
      <jsdl:Source>  
        <jsdl:URI>gsiftp://host:port/dir/data2.dat</jsdl:URI>  
      </jsdl:Source>  
    </jsdl:DataStaging>  
  </jsdl:JobDescription>  
  
  <sweep:Sweep>  
    <sweep:Assignment>  
      <file-sweep:FileSweep>  
        <file-sweep:TemplateFile>  
          <jsdl:FileName>data2.dat</jsdl:FileName>  
          <jsdl:FileSystemName>DATADIR</jsdl:FileSystemName>  
        </file-sweep:TemplateFile>  
        <file-sweep:FileToken value="colour"  
          assignDefault="blue"/>  
        <file-sweep:FileToken value="animal.2a"/>  
        <file-sweep:FileToken value="animal.2b"/>  
      </file-sweep:FileSweep>  
      <sweepfunc:Values>  
        <sweepfunc:Value>cat</sweepfunc:Value>  
        <sweepfunc:Value>dog</sweepfunc:Value>
```

```
        <sweepfunc:Value>bird</sweepfunc:Value>
      </sweepfunc:Values>
    </sweep:Assignment>
  </sweep:Sweep>
</jsdl:JobDefinition>
```

The file sweep job yields three individual jobs. Notice that the same Sweep Function values are assigned to the template file tokens in data2.dat. Also note that the 'colour' token is always assigned a default value of 'blue' whilst data1.dat remains unmodified. Standard output displays:

data1.dat: The quick colour animal.1a jumps over the lazy animal.1b

data2.dat: The quick blue cat jumps over the lazy cat

data1.dat: The quick colour animal.1a jumps over the lazy animal.1b

data2.dat: The quick blue dog jumps over the lazy dog

data1.dat: The quick colour animal.1a jumps over the lazy animal.1b

data2.dat: The quick blue bird jumps over the lazy bird

## 7.2 Example 2

In this example, different Parameter Sweep Function values are assigned to each token in one template file (data2.dat).

```
<sweep:Sweep>
  <sweep:Assignment>
    <file-sweep:FileSweep>
      <file-sweep:TemplateFile>
        <jsdl:FileName>data2.dat</jsdl:FileName>
        <jsdl:FileSystemName>DATADIR</jsdl:FileSystemName>
      </file-sweep:TemplateFile>
      <file-sweep:FileToken value="colour"/>
    </file-sweep:FileSweep>
    <sweepfunc:Values>
      <sweepfunc:Value>blue</sweepfunc:Value>
      <sweepfunc:Value>red</sweepfunc:Value>
      <sweepfunc:Value>green</sweepfunc:Value>
    </sweepfunc:Values>
  </sweep:Assignment>

  <sweep:Assignment>
    <file-sweep:FileSweep>
      <file-sweep:TemplateFile>
        <jsdl:FileName>data2.dat</jsdl:FileName>
        <jsdl:FileSystemName>DATADIR</jsdl:FileSystemName>
      </file-sweep:TemplateFile>
      <file-sweep:FileToken value="animal.2a"/>
    </file-sweep:FileSweep>
    <sweepfunc:Values>
      <sweepfunc:Value>cat</sweepfunc:Value>
      <sweepfunc:Value>dog</sweepfunc:Value>
      <sweepfunc:Value>cow</sweepfunc:Value>
    </sweepfunc:Values>
  </sweep:Assignment>
```

```
<sweep:Assignment>
  <file-sweep:FileSweep>
    <file-sweep:TemplateFile>
      <jsdl:FileName>data2.dat</jsdl:FileName>
      <jsdl:FileSystemName>DATADIR</jsdl:FileSystemName>
    </file-sweep:TemplateFile>
    <file-sweep:FileToken value="animal.2b"/>
  </file-sweep:FileSweep>
  <sweepfunc:Values>
    <sweepfunc:Value>bird</sweepfunc:Value>
    <sweepfunc:Value>spider</sweepfunc:Value>
    <sweepfunc:Value>snake</sweepfunc:Value>
  </sweepfunc:Values>
</sweep:Assignment>
</sweep:Sweep>
```

The file sweep job yields three individual jobs (data1.dat remains unmodified). Notice that different Sweep Function values are assigned to different tokens in the same template file (data2.dat). Standard output displays:

data1.dat: The quick colour animal.1a jumps over the lazy animal.1b  
data2.dat: The quick blue cat jumps over the lazy bird

data1.dat: The quick colour animal.1a jumps over the lazy animal.1b  
data2.dat: The quick red dog jumps over the lazy spider

data1.dat: The quick colour animal.1a jumps over the lazy animal.1b  
data2.dat: The quick green cow jumps over the lazy snake

### 7.3 Example 3

In this example, values are assigned to tokens in both template files. Notice that when template files have common token names (e.g. colour) multiple TemplateFile elements can be declared within the same FileSweep element in order to assign the same values.

```
<sweep:Sweep>
  <sweep:Assignment>
    <file-sweep:FileSweep>
      <file-sweep:TemplateFile>
        <jsdl:FileName>data1.dat</jsdl:FileName>
      </file-sweep:TemplateFile>
      <file-sweep:TemplateFile>
        <jsdl:FileName>data2.dat</jsdl:FileName>
        <jsdl:FileSystemName>DATADIR</jsdl:FileSystemName>
      </file-sweep:TemplateFile>
      <file-sweep:FileToken value="colour"/>
    </file-sweep:FileSweep>
    <sweepfunc:Values>
      <sweepfunc:Value>blue</sweepfunc:Value>
      <sweepfunc:Value>red</sweepfunc:Value>
      <sweepfunc:Value>green</sweepfunc:Value>
    </sweepfunc:Values>
  </sweep:Assignment>

  <sweep:Assignment>
    <file-sweep:FileSweep>
```

```
<file-sweep:TemplateFile>
  <jSDL:FileName>data1.dat</jSDL:FileName>
</file-sweep:TemplateFile>
<file-sweep:FileToken value="animal.1a"/>
<file-sweep:FileToken value="animal.1b"/>
</file-sweep:FileSweep>
<file-sweep:FileSweep>
  <file-sweep:TemplateFile>
    <jSDL:FileName>data2.dat</jSDL:FileName>
    <jSDL:FileSystemName>DATADIR</jSDL:FileSystemName>
  </file-sweep:TemplateFile>
  <file-sweep:FileToken value="animal.2a"/>
</file-sweep:FileSweep>
<sweepfunc:Values>
  <sweepfunc:Value>cat</sweepfunc:Value>
  <sweepfunc:Value>dog</sweepfunc:Value>
  <sweepfunc:Value>bird</sweepfunc:Value>
</sweepfunc:Values>
</sweep:Assignment>

<sweep:Assignment>
  <file-sweep:FileSweep>
    <file-sweep:TemplateFile>
      <jSDL:FileName>data2.dat</jSDL:FileName>
      <jSDL:FileSystemName>DATADIR</jSDL:FileSystemName>
    </file-sweep:TemplateFile>
    <file-sweep:FileToken value="animal.2b"/>
  </file-sweep:FileSweep>
  <sweepfunc:Values>
    <sweepfunc:Value>cow</sweepfunc:Value>
    <sweepfunc:Value>spider</sweepfunc:Value>
    <sweepfunc:Value>snake</sweepfunc:Value>
  </sweepfunc:Values>
</sweep:Assignment>
</sweep:Sweep>
```

The file sweep job yields three individual jobs. Both data1.dat and data2.dat are modified. Standard output displays:

data1.dat: The quick *blue cat* jumps over the lazy *cat*  
data2.dat: The quick *blue cat* jumps over the lazy *cow*

data1.dat: The quick *red dog* jumps over the lazy *dog*  
data2.dat: The quick *red dog* jumps over the lazy *spider*

data1.dat: The quick *green bird* jumps over the lazy *bird*  
data2.dat: The quick *green bird* jumps over the lazy *snake*

#### 7.4 Example 4

This example illustrates the use of nested FileSweep elements. Notice that the outer sweep always assigns the same set of values to tokens in data1.dat while different values are assigned to tokens in data2.dat. The submitted JSDL Job template looks as follows.

```
<sweep:Sweep>
  <sweep:Assignment>
```

```
<file-sweep:FileSweep>
  <file-sweep:TemplateFile>
    <jsdl:FileName>data1.dat</jsdl:FileName>
  </file-sweep:TemplateFile>
  <file-sweep:FileToken value="colour"
    assignDefault="blue"/>
  <file-sweep:FileToken value="animal.1a"/>
  <file-sweep:FileToken value="animal.1b"/>
</file-sweep:FileSweep>
<sweepfunc:Values>
  <sweepfunc:Value>cat</sweepfunc:Value>
  <sweepfunc:Value>dog</sweepfunc:Value>
  <sweepfunc:Value>bird</sweepfunc:Value>
</sweepfunc:Values>
</sweep:Assignment>

<sweep:Sweep>
  <sweep:Assignment>
    <file-sweep:FileSweep>
      <file-sweep:TemplateFile>
        <jsdl:FileName>data2.dat</jsdl:FileName>
        <jsdl:FileSystemName>
          DATADIR
        </jsdl:FileSystemName>
      </file-sweep:TemplateFile>
      <file-sweep:FileToken value="colour"/>
    </file-sweep:FileSweep>
    <sweepfunc:Values>
      <sweepfunc:Value>blue</sweepfunc:Value>
      <sweepfunc:Value>red</sweepfunc:Value>
      <sweepfunc:Value>green</sweepfunc:Value>
    </sweepfunc:Values>
  </sweep:Assignment>
  <sweep:Assignment>
    <file-sweep:FileSweep>
      <file-sweep:TemplateFile>
        <jsdl:FileName>data2.dat</jsdl:FileName>
        <jsdl:FileSystemName>
          DATADIR
        </jsdl:FileSystemName>
      </file-sweep:TemplateFile>
      <file-sweep:FileToken value="animal.2a" />
    </file-sweep:FileSweep>
    <sweepfunc:Values>
      <sweepfunc:Value>cat</sweepfunc:Value>
      <sweepfunc:Value>dog</sweepfunc:Value>
      <sweepfunc:Value>cow</sweepfunc:Value>
    </sweepfunc:Values>
  </sweep:Assignment>
  <sweep:Assignment>
    <file-sweep:FileSweep>
      <file-sweep:TemplateFile>
        <jsdl:FileName>data2.dat</jsdl:FileName>
        <jsdl:FileSystemName>
          DATADIR
        </jsdl:FileSystemName>
      </file-sweep:TemplateFile>
    </file-sweep:FileSweep>
  </sweep:Assignment>
</sweep:Sweep>
```

```
        <file-sweep:FileToken value="animal.2b" />
    </file-sweep:FileSweep>
    <sweepfunc:Values>
        <sweepfunc:Value>bird</sweepfunc:Value>
        <sweepfunc:Value>spider</sweepfunc:Value>
        <sweepfunc:Value>snake</sweepfunc:Value>
    </sweepfunc:Values>
</sweep:Assignment>
</sweep:Sweep>
</sweep:Sweep>
```

The file sweep job yields nine individual jobs. Both data1.dat and data2.dat are modified. Standard output displays:

data1.dat: The quick blue cat jumps over the lazy cat

data2.dat: The quick blue cat jumps over the lazy bird

data1.dat: The quick blue cat jumps over the lazy cat

data2.dat: The quick red dog jumps over the lazy spider

data1.dat: The quick blue cat jumps over the lazy cat

data2.dat: The quick green cow jumps over the lazy snake

data1.dat: The quick blue dog jumps over the lazy dog

data2.dat: The quick blue cat jumps over the lazy bird

data1.dat: The quick blue dog jumps over the lazy dog

data2.dat: The quick red dog jumps over the lazy spider

data1.dat: The quick blue dog jumps over the lazy dog

data2.dat: The quick green cow jumps over the lazy snake

data1.dat: The quick blue bird jumps over the lazy bird

data2.dat: The quick blue cat jumps over the lazy bird

data1.dat: The quick blue bird jumps over the lazy bird

data2.dat: The quick red dog jumps over the lazy spider

data1.dat: The quick blue bird jumps over the lazy bird

data2.dat: The quick green cow jumps over the lazy snake

## 7.5 Example 5

This example shows an illegal use of a FileSweep element. It is illegal because two infringing FileToken elements declare the same FileToken value (colour) within the same FileSweep element. The infringing parts of the example are outlined in bold red font face.

```
<sweep:Sweep>
  <sweep:Assignment>
    <file-sweep:FileSweep>
      <file-sweep:TemplateFile>
        <jsdl:FileName>data1.dat</jsdl:FileName>
      </file-sweep:TemplateFile>
      <file-sweep:FileToken value="colour"/>
      <file-sweep:FileToken value="colour"/>
    </file-sweep:FileSweep>
  </sweep:Assignment>
</sweep:Sweep>
```



```
<sweepfunc:Values>
  <sweepfunc:Value>red</sweepfunc:Value>
  <sweepfunc:Value>blue</sweepfunc:Value>
  <sweepfunc:Value>green</sweepfunc:Value>
</sweepfunc:Values>
</sweep:Assignment>
</sweep:Sweep>
```

## 7.6 Example 6

This example shows an illegal use of a FileSweep element. It is illegal because two FileSweep elements declare the same infringing FileToken value (colour) for the same template file (data1.dat) within the same Sweep element context.

```
<sweep:Sweep>
  <sweep:Assignment>
    <file-sweep:FileSweep>
      <file-sweep:TemplateFile>
        <jsdl:FileName>data1.dat</jsdl:FileName>
      </file-sweep:TemplateFile>
      <file-sweep:FileToken value="colour"/>
    </file-sweep:FileSweep>
    <file-sweep:FileSweep>
      <file-sweep:TemplateFile>
        <jsdl:FileName>data1.dat</jsdl:FileName>
      </file-sweep:TemplateFile>
      <file-sweep:FileToken value="colour"/>
    </file-sweep:FileSweep>
    <sweepfunc:Values>
      <sweepfunc:Value>red</sweepfunc:Value>
      <sweepfunc:Value>blue</sweepfunc:Value>
      <sweepfunc:Value>green</sweepfunc:Value>
    </sweepfunc:Values>
  </sweep:Assignment>
</sweep:Sweep>
```

## 8. Security Considerations

This document defines a language that can be used to declare the member jobs of a Parameter Sweep. It does not define any security related semantics such as format and possible contents of security tokens.

Having said that, security as such is, even though an important part of Grids, orthogonal to the contents of this document.

## 9. Contributors

Michel Drescher  
Fujitsu Laboratories of Europe, Ltd.  
Hayes Park Central, Hayes End Road  
Hayes, Middlesex UB4 8FE  
United Kingdom

Ali Anjomshoa (Independent)

Geoff Williams  
Oxford University Computing Laboratory,  
Wolfson Building, Parks Road,  
Oxford OX1 3QD

David Meredith  
Science and Technology Facilities Council  
e-Science Centre, Daresbury Laboratory  
Warrington, Cheshire, WA4 4AD  
UK

The authors would like to thank KISTI as a whole, and Byungsang Kim and Soonwook Hwang in particular, for their outstanding efforts and contributions to this specification.

The authors also wish to thank everybody who contributed for their valuable comments including, but not limited to and in no particular order, Andreas Savva, Kazushige Saga and the NAREGI project as a whole, David Snelling, Hans-Christian Hoppe, Jay Unger, Steven McGough, Donal Fellows.

## **10. 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.

## **11. 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.

## **12. Full Copyright Notice**

Copyright © Open Grid Forum 2006-2008. 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 assignees.

### 13. References

- [BRADNER] Bradner, S. et al Key Words for Use in RFCs to Indicate Requirement Levels, RFC 2119. March 1997.
- [BRAY] Bray, T. et al Namespaces in XML 1.0, W3C Recommendation. March 1997.
- [INFOSET] Cowan, J. et al XML Information Set (Second Edition), W3C Recommendation. February 2004.
- [JSDL] Anjomshoaa, A. et al Job Submission Description Language (JSDL) Specification, Version 1.0, GFD.56. November 2005.
- [SCHEMA1] Thompson, H. et al XML Schema Part 1: Structures (Second Edition), W3C Recommendation. October 2004.
- [SCHEMA2] Biron, P. et al XML Schema Part 2: Datatypes (Second Edition), W3C Recommendation. October 2004.
- [WSSEC] Nadalin, A. et al Web Services Security: SOAP Message Security 1.1, OASIS Standard. February 2006.
- [XPath] Berglund, A. et al XML Path Language (XPath) 2.0, W3C Recommendation. January 2007.

## Appendix A. Normative XML Schema for the Parameter Sweep Information Set

The following defines the normative XML Schema for the Parameter Sweep Information Set as defined in section 2.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://schemas.ogf.org/jsdl/2007/04/sweep"
  xmlns:sweep="http://schemas.ogf.org/jsdl/2007/04/sweep"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified" version="1">

  <xsd:element name="Function" abstract="true"/>

  <xsd:element name="Parameter" abstract="true" />

  <xsd:element name="NamespaceBinding">
    <xsd:complexType>
      <xsd:attribute name="ns" type="xsd:anyURI"
        use="required" />
      <xsd:attribute name="prefix" type="xsd:NCName"
        use="required" />
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="DocumentNode"
    substitutionGroup="sweep:Parameter">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="sweep:NamespaceBinding"
          minOccurs="1" maxOccurs="unbounded" />
        <xsd:element name="Match" type="xsd:string" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="Assignment">
    <xsd:complexType mixed="false">
      <xsd:sequence>
        <xsd:element ref="sweep:Parameter"
          minOccurs="1" maxOccurs="unbounded" />
        <xsd:element ref="sweep:Function"
          minOccurs="1" maxOccurs="1" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:complexType name="Sweep_Type" mixed="false">
    <xsd:sequence>
      <xsd:element ref="sweep:Assignment"
        minOccurs="1" maxOccurs="unbounded" />
      <xsd:element ref="sweep:Sweep"
        minOccurs="0" maxOccurs="unbounded" />
      <xsd:any namespace="##other" processContents="lax"
        minOccurs="0" maxOccurs="unbounded" />
    </xsd:sequence>
    <xsd:anyAttribute namespace="##other"
      processContents="lax" />
  </xsd:complexType>
</xsd:schema>
```

```
</xsd:complexType>  
  
<xsd:element name="Sweep" type="sweep:Sweep_Type"/>  
  
</xsd:schema>
```

## Appendix B. Normative XML Schema for the Parameter Sweep Functions Information Set

The following defines the normative XML Schema for the Parameter Sweep Functions Information Set as defined in section 3.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
targetNamespace="http://schemas.ogf.org/jsdl/2007/04/sweep/functions"
  xmlns:sweep="http://schemas.ogf.org/jsdl/2007/04/sweep"

xmlns:sweepfunc="http://schemas.ogf.org/jsdl/2007/04/sweep/functions"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified" version="1">

  <xsd:import
    namespace="http://schemas.ogf.org/jsdl/2007/04/sweep"
    schemaLocation="./sweep.xsd"/>

  <xsd:element name="Values" substitutionGroup="sweep:Function">
    <xsd:complexType mixed="false">
      <xsd:sequence>
        <xsd:element name="Value" type="xsd:anyType"
          nillable="false"
          minOccurs="1" maxOccurs="unbounded"/>
      </xsd:sequence>
      <xsd:anyAttribute namespace="##other"
        processContents="lax" />
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="LoopInteger" substitutionGroup="sweep:Function">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Exception" type="xsd:integer"
          maxOccurs="unbounded" minOccurs="0" />
      </xsd:sequence>
      <xsd:attribute name="start" type="xsd:integer"
        use="required"/>
      <xsd:attribute name="end" type="xsd:integer"
        use="required"/>
      <xsd:attribute name="step" use="optional"
        type="xsd:integer" default="1"/>
      <xsd:anyAttribute namespace="##other"
        processContents="lax" />
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="LoopDouble" substitutionGroup="sweep:Function">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Exception" type="xsd:double"
          maxOccurs="unbounded" minOccurs="0" />
      </xsd:sequence>
      <xsd:attribute name="start" type="xsd:double"
        use="required" />
    </xsd:complexType>
  </xsd:element>
```

```
        <xsd:attribute name="end" type="xsd:double"
                      use="required" />
        <xsd:attribute name="step" type="xsd:double"
                      use="required" />
        <xsd:anyAttribute namespace="##other"
                          processContents="lax" />
    </xsd:complexType>
</xsd:element>
</xsd:schema>
```

## Appendix C. Normative XML Schema for the File Sweep Information Set

The following defines the normative XML Schema for the File Sweep Information Set.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns="http://schemas.ogf.org/jsdl/2008/08/file-sweep"
  xmlns:jsdl="http://schemas.ogf.org/jsdl/2005/11/jsdl"
  xmlns:sweep="http://schemas.ogf.org/jsdl/2007/04/sweep"
  xmlns:file-sweep="http://schemas.ogf.org/jsdl/2008/08/file-sweep"
  targetNamespace="http://schemas.ogf.org/jsdl/2008/08/file-sweep"
  elementFormDefault="qualified" version="1">

  <xsd:import namespace="http://schemas.ogf.org/jsdl/2007/04/sweep"
    schemaLocation="./sweep.xsd"/>
  <xsd:import namespace="http://schemas.ogf.org/jsdl/2005/11/jsdl"
    schemaLocation="../jsdl/jsdl.2005_11.xsd"/>

  <xsd:element name="FileSweep" type="FileSweep_Type"
    substitutionGroup="sweep:Parameter"/>

  <xsd:complexType name="FileSweep_Type">
    <xsd:sequence>
      <xsd:element name="TemplateFile" type="TemplateFile_Type"
        minOccurs="1" maxOccurs="unbounded"/>
      <xsd:element name="FileToken" type="FileToken_Type"
        minOccurs="1" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="TemplateFile_Type">
    <xsd:sequence>
      <xsd:element ref="jsdl:FileName"/>
      <xsd:element ref="jsdl:FilesystemName" minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="FileToken_Type">
    <xsd:attribute name="value" type="xsd:string" use="required"/>
    <xsd:attribute name="assignDefault" type="xsd:string" />
  </xsd:complexType>

</xsd:schema>
```