GWD-E
OGSA-RSS-WG
Donal K. Fellows, University of Manchester
February 7, 2006
Updated: October 5, 2007

OGSA® Resource Selection Services: Specification

Status of This Document

This document provides information to the Grid community on the definitions of the OGSA Resource Selection service interfaces. Distribution is unlimited.

Copyright Notice

Trademark

Abstract

The OGSA® Resource Selection Services specification defines an abstract interface for performing queries used to select resources for any purpose. The specification also narrows the abstract interface for the specific purpose of selecting a Basic Execution Service on which to instantiate an activity based upon a Job Submission Description Language document, and defines renderings for such services based on both the WS-Resource Framework and the WS-Transfer draft. The final part of this specification gives a simple language for defining a function capable of ranking one candidate for execution versus another.

Contents

Abstract 1
1. Introduction 3
2. Notational Conventions 3
3. What are the Resource Selection Services? 3
4. Specification of a Candidate Set Generator 4
4.1 CSG Abstract Functional Interface 4
4.2 CSG Basic Input/Output Template Model 5
4.3 Example Request for Candidates 6
4.4 Common CSG Properties 7
5. Specification of the Basic Execution Planning Service Profile 7
5.1 Contents of the *CandidateQualityProperties* 8
5.2 Standard Interpretation of the Candidate Execution Plan 11
5.3 Example Candidate Execution Plan 11
6. Concrete Rendering of the Basic Execution Planning Service 12
6.1 Conceptual Operation of the Basic-EPS 12
6.2 Specialization of the Basic-EPS to WS-RF 12
6.3 Specialization of the EPS to WS-Transfer/WS-Enumeration 18
7. Simple Candidate Ordering Language 19

7.1 Organizing Principles of SCOL for Candidate Generation 20
7.2 SCOL Syntax and Semantics 20
7.3 Example *OrderFunction* 25
7.4 Extending SCOL 26
8. Security Considerations 26
9. Contributors 26
10. Intellectual Property Statement 27
11. Disclaimer 27
12. Full Copyright Notice 27
13. References 27
Appendix A: Normative Schemas 29
A.1 XML Schema for CSG Core 29
A.2 XML Schema for EPS Core 34
A.3 XML Schema for Basic-EPS-RF 39
A.4 WSDL for Basic-EPS-RF 44
A.5 XML Schema for Basic-EPS-XF 50
A.6 WSDL for Basic-EPS-XF 51
A.7 Supplemental XML Schema for WS-Enumeration 53
A.8 XML Schema for SCOL 54

1. Introduction

This document describes the parts of the OGSA® Execution Management architecture[EMSS] that pertain to the selection of resources upon which to execute a job or carry out some other activity. It details a set of interfaces that services that provide resource selection services should implement, so that clients of those services can efficiently choose where to carry out the activity, given that any particular choice is not guaranteed to succeed. This document also specializes those interfaces for the task deciding on what Basic Execution Service[BES] container to execute jobs described in Job Submission Description Language[JSDL] and describes how to render those services as a set of WS-Resources[WSR].

2. Notational Conventions

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

The specification provides a pseudo-schema for each component as part of 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 that 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 /> ?
    <zero_or_more_of_this_element /> *
    <one_or_more_of_this_element /> +
</defined_element>
```

The following namespaces are used in this document, with the prefixes described in this table:

Prefix	Namespace
xsd	http://www.w3.org/2001/XMLSchema
wsen	http://schemas.xmlsoap.org/ws/2004/09/enumeration
jsdl	http://schemas.ggf.org/jsdl/2005/11/jsdl
csg	http://schemas.ogf.org/rss/2007/03/csg
eps-basic	http://schemas.ogf.org/rss/2007/03/eps/basic
eps-wsrf	http://schemas.ogf.org/rss/2007/03/eps/wsrf
eps-xfer	http://schemas.ogf.org/rss/2007/10/eps/transfer
select	http://schemas.ogf.org/rss/2007/05/scol

3. What are the Resource Selection Services?

A Resource Selection Service is a service within the Open Grid Services Architecture® which is used to choose some other service or resource to perform an operation. In particular, the OGSA®

Execution Management Services stack requires a kind of resource selection service called an Execution Planning Service (EPS) whose role it is to take an abstract plan of an execution and suggest particular execution services upon which the plan may be carried out. This is required when the set of services which may provide suggestions is large, when the set of suggestions provided by a particular service is itself large (e.g., because of many possible configurations) or when the decision must be taken autonomously (e.g., when a job fails on one resource and must be rescheduled elsewhere). As such, the EPS forms the core of how to build a higher-level Grid that abstracts away from the details of individual resources.

All suggestions made by an execution planning service are inherently out of date. This is a consequence of the fact that whatever information the EPS uses to make the decision may itself be out of date too, resulting in the suggestions having to always be regarded as being made on a best-effort basis. Only once a reservation has been made can any sort of substantive guarantee (such as a contract) of a particular level of service quality be established; such reservations are outside the scope of this document.

Because of this fundamental lack of any level of guarantee of the quality of any particular suggestion, all resource selection services return sets of suggestions. This means that the caller can, if one suggestion proves impossible to enact, choose another from the set and try again without having to re-query the resource selection service.

However, it is important for the leading suggestions returned by any resource selection service to be of “high quality”. By this, we mean that it should be likely that the first suggestion will satisfy the client’s request, assuming such a request is satisfiable at all. It is important because this means that the client can avoid the majority of network traffic and achieve their goal (e.g., running a job on a Basic Execution Service instance) with a minimum of overhead. The difficulty with this is that the definition of what is “high quality” is itself dependent on what the clients’ goals are. This forces us to consider services where the clients describe to the services how to pick a good plan.

4. Specification of a Candidate Set Generator

A Candidate Set Generator (CSG) is an abstract service that acts as a mapping from an Input Description document to a set of Output Candidate documents. To facilitate efficient implementation, every Candidate Set Generator MUST accept a Candidate Ordering document which imposes a total order on the set of Output Candidates, and MUST return the Output Candidate set as an ordered set, sorted according to that total order. If the client of the service does not supply a Candidate Ordering document, the CSG may use any ordering of the Output Candidates that its implementer desires; if implementations of the CSG specification advertise themselves in some manner, they SHOULD describe what their default ordering is.

4.1 CSG Abstract Functional Interface

Abstractly, the only CSG operation is:

getCandidates: *InputDescription* * optional(*CandidateOrdering*)
→ orderedSet(*OutputCandidate*)

Note that the abstract definition of a CSG does *not* constrain the meaning of the Input Description or Output Candidate documents (other than to state that each Output Candidate should “satisfy” the Input Description in some sense. The only constraint on the Candidate Ordering is that it should abstractly define a (partial) mapping from Output Candidates to floating point values; if a particular Output Candidate is not part of the domain of the function induced by the Candidate Ordering, that Output Candidate MUST NOT be part of the resulting ordered set returned by the CSG.

The core abstract CSG specification does not specify how to render the service on top of a particular web-service stack. That is instead part of the binding of the service.

Every type of Candidate Ordering document language MUST have a unique URI that describes it, and a CSG implementation SHOULD indicate what Candidate Ordering document languages it accepts. If the *getCandidates* operation is invoked with a Candidate Ordering document whose

URI is not understood, the operation MUST fail; all concrete renderings of a CSG MUST describe a mechanism for this failure that is distinguishable from any other failure supported by the concrete rendering.

4.2 CSG Basic Input/Output Template Model

Every CSG must use the following abstract schemas to describe its input and output documents when rendered in XML. This document does not describe non-XML renderings of the CSG.

All named elements in this section MUST use the namespace:

```
http://schemas.ogf.org/rss/2007/03/csg
```

All CSG renderings MUST support a way of reading a property of every CSG instance, *UsageProfile*, whose contents MUST be a set of URIs characterizing the contents of the *InputDescription* and *CandidateDescription* elements (described below) and how they are intended to be used. All CSG renderings MUST support a way of reading a property of every CSG instance, *OrderingLanguages*, whose contents MUST be a set of URIs characterizing the permitted formats of the *CandidateOrdering* elements (described below). Each CSG profile and each ordering language MUST define at least one URI; URIs beneath http://schemas.ogf.org/rss/ are reserved for OGF use when not otherwise described.

4.2.1 Input Description Model

The request element to the CSG MUST be named *RequestCandidates*, MUST contain an *InputDescription* element, and MUST permit the supply of a *UsageProfile* element (as first element), a *CandidateOrdering* element (after the *InputDescription*) and zero or more *PermittedCandidateExtension* elements after that. Profiles of CSG SHOULD augment this with further class-specific parameters.

The contents of the *InputDescription* element MUST be determined by the profile specified in the UsageProfile element, if given; if omitted, the CSG SHOULD use a default profile but MAY instead guess based on the content of the *InputDescription*. The *CandidateOrdering* element MUST have a *dialect* element whose content MUST be an arbitrary URI that describes the content of the element and the manner in which it is to be used to generate the ordered set of candidates.

Each *PermittedCandidateExtension* element must describe (using a QName) the extension element to which it is referring, and SHOULD give (via an optional *type* attribute) where in a candidate the element whose name is the content of the *PermittedCandidateExtension* element may be placed. The *type* attribute MUST be of type *ExtensionType*, which is an enumeration consisting of “functional” and “quality”, defaulting to “functional” if unspecified. Note that the presence of a *PermittedCandidateExtension* SHALL NOT force the processing CSG to generate a *Candidate* that contains such an extension element.

```
<csg:RequestCandidates>
        <csg:UsageProfile> xsd:anyURI </csg:UsageProfile> ?
        <csg:InputDescription> … </csg:InputDescription>
        <csg:CandidateOrdering dialect="xsd:anyURI"> … </csg:CandidateOrdering> ?
        <csg:PermittedCandidateExtension type="csg:ExtensionType"?>
                xsd:QName
        </csg:PermittedCandidateExtension> *
        …
</csg:RequestCandidates>
```

4.2.2 Output Candidate Model

Each candidate returned by the CSG MUST be an element named *Candidate*, whose contents MUST consist of the following three elements, *CandidateDescription*, *CandidateQualityProperties*, *CandidateValidity*, followed by any arbitrary sequence of elements from XML namespaces other than the CSG namespace. Those added elements beyond what is required in the

profile MUST be described by *PermittedCandidateExtension* elements (with type equal to “functional”) in the *RequestCandidates* which caused the containing *Candidate* element to be generated.

The *CandidateDescription* element MUST describe the candidate itself, according to the CSG usage profile which MUST specify the content model of the element. It MUST represent a best-effort suggestion of how to satisfy the request from the *RequestCandidate* element described above.

The *CandidateQualityProperties* element MUST describe the reasons for selecting one candidate over another, e.g. price, reliability. The content model of the *CandidateQualityProperties* element MUST be determined by the CSG usage profile. All extensions to the *CandidateQualityProperties* element beyond what is required in the profile MUST be described by *PermittedCandidateExtension* elements (with type equal to “quality”) in the *RequestCandidates* which caused the containing *Candidate* element to be generated.

The *CandidateValidity* element MUST describe the continuous period of time for which the candidate as a whole is valid, i.e. the range of times which the CSG expects (under best-effort conditions) that the client of the CSG can reasonably act upon the *CandidateDescription* (according to the manner characterized by the profile of the CSG) and get the sort of behavior outlined by the *CandidateQualityProperties* element. The *CandidateValidity* element MUST be empty, and MUST have two attributes, *notBefore* and *notAfter*, both of which MUST be formatted as the XML Schema datatype *dateTime*. The *notBefore* element MUST describe the instant at which the validity period commences, and the *notAfter* element MUST describe the instant at which the validity period ends. Discontinuous validity periods MUST be represented as multiple candidates.

```
<csg:Candidate>
    <csg:CandidateDescription> … </csg:CandidateDescription>
    <csg:CandidateQualityProperties> … </csg:CandidateQualityProperties>
    <csg:CandidateValidity notBefore="xsd:dateTime" notAfter="xsd:dateTime" />
    <xsd:any namespace="##other" processContents="lax" /> *
</csg:Candidate>
```

4.3 Example Request for Candidates

This example is a request for candidates that fits the profile of an Execution Planning Service (the definition of this profile is given in Section 5), and which contains a JSDL document specifying that the candidates are to be for executions of the BLAST application. The order of the candidates is determined by an ordering element (in this case using an arbitrary private dialect), and candidates may optionally include WS-Agreement[WSAG] *Template* elements.

To aid the reading of this example, the bold elements are those parts that are due to this being a document that is sent to a candidate set generator.

```
<csg:RequestCandidates>
    <csg:UsageProfile> http://schemas.ogf.org/rss/2007/03/eps/basic </csg:UsageProfile>
    <csg:InputDescription>
        <jsdl:JobDefinition>
            <jsdl:JobDescription>
                <jsdl:Application>
                    <jsdl:ApplicationName> BLAST </jsdl:ApplicationName>
                </jsdl:Application>
            </jsdl:JobDescription>
        </jsdl:JobDefinition>
    </csg:InputDescription>
    <csg:CandidateOrdering dialect="urn:private:demo">
        Order by Amortized Price
    </csg:CandidateOrdering>
    <csg:PermittedCandidateExtension type="functional"
```

```
                    xmlns:wsag="http://schemas.ggf.org/graap/2007/03/ws-agreement">
                wsag:Template
        </csg:PermittedCandidateExtension>
</csg:RequestCandidates>
```

4.4 Common CSG Properties

Every CSG must provide some common characteristic properties that describe the CSG so that clients may interact with it successfully. Note that the CSG definition itself does not define how these properties are discovered, but individual profiles or renderings SHOULD do this. All properties in this section MUST use the namespace when described as XML (other renderings are possible, but not described in this document):

```
http://schemas.ogf.org/rss/2007/03/csg
```

4.4.1 The *UsageProfile* Property

Every candidate set generator MUST provide this property as a sequence of one or more elements. Each element MUST be called *UsageProfile*, and have a content model that allows any URI. Each listed URI must describe a profile on the general CSG that describes how the CSG supports a particular mode of operation (e.g., as an Execution Planning Service).

4.4.2 The *SupportedOrderingLanguage* Property

Every candidate set generator MUST provide this property as a sequence of one or more elements. Each element MUST be called *SupportedOrderingLanguage*, and have a content model that allows any URI. Each listed URI must describe an ordering language supported by the CSG (e.g., the Simple Candidate Ordering Language).

5. Specification of the Basic Execution Planning Service Profile

A Basic Execution Planning Service (Basic-EPS) is a profile of the Candidate Set Generator abstract service where the contents of the *InputDescription* element MUST be a JSDL *JobDefinition* document, and the contents of the *CandidateDescription* element must be a sequence of a JSDL *JobDefinition* and an element, called *BESReference*, containing a WS-Addressing[WSA] Endpoint Reference to a service implementing the Basic Execution Service port-type. Within the context of the Basic-EPS profile, the contents of a *Candidate* element may be referred to as a candidate execution plan (CEP). The usage model supported by this usage profile is that the consumer of the candidate execution plan takes action on the CEP by submitting the *JobDefinition* within the CEP to the Basic Execution Service described by the CEP. The *CandidateQualityProperties* element MUST support the elements described further below in this section. Note that the requirements of the CSG *PermittedCandidateExtension* MUST still be respected.

Note that the consumer of the candidate SHOULD ignore any candidate that contains elements outside the *CandidateQualityProperties* that it does not recognize and understand.

The URI for this usage profile is:

```
http://schemas.ogf.org/rss/2007/03/eps/basic
```

The elements defined in this section (including the *BESReference* element outlined above) MUST be in the namespace:

```
http://schemas.ogf.org/rss/2007/03/eps/basic
```

All Basic-EPS instances must support the ordering language with the URI:

```
http://schemas.ogf.org/rss/2006/12/scol
```

Note that a Basic-EPS MAY return a JSDL *JobDefinition* in the *CandidateDescription* that is substantially different to the JSDL *JobDefinition* inside the *InputDescription*. In particular, a different way of specifying what the application is MAY be given for a particular BES, different resources

MAY be selected, or additional file staging requirements MAY be added. This list of ways of variation is not intended to be exhaustive.

The pseudo-schema for *RequestCandidates* under the Basic-EPS profile is as follows:

```
<csg:RequestCandidates>
        <csg:UsageProfile> xsd:anyURI </csg:UsageProfile> ?
        <csg:InputDescription>
                <jsdl:JobDefinition> … </jsdl:JobDefinition>
        </csg:InputDescription>
        <csg:CandidateOrdering> … </csg:CandidateOrdering> ?
        <csg:PermittedCandidateExtension csg:type="csg:ExtensionType"?>
                xsd:QName
        </csg:PermittedCandidateExtension> *
</csg:RequestCandidates>
```

Under this profile, the content of the *UsageProfile* element, if present, MUST be the URI for this profile.

The pseudo-schema for *Candidate* (the meanings of the elements inside the *CandidateQualityProperties* element are discussed below) under the Basic-EPS profile is as follows:

```
<csg:Candidate>
        <csg:CandidateDescription>
                <jsdl:JobDefinition> … </jsdl:JobDefinition>
                <eps-basic:BESReference>
                        ws-a:EndpointReferenceType
                </eps-basic:BESReference>
        </csg:CandidateDescription>
        <csg:CandidateQualityProperties>
                <eps-basic:Prices … /> ?
                <eps-basic:StartTimes … /> ?
                <eps-basic:StartDelays … /> ?
                <eps-basic:EndTimes  … /> ?
                <eps-basic:ExecutionDuration  … /> ?
                <xsd:any namespace="##other" processContents="lax" /> *
        </csg:CandidateQualityProperties>
        <csg:CandidateValidity csg:notBefore="xsd:dateTime" csg:notAfter="xsd:dateTime" />
        <xsd:any namespace="##other" processContents="lax" /> *
</csg:Candidate>
```

5.1 Contents of the *CandidateQualityProperties*

Every CSG that supports the Basic-EPS profile MUST support the following elements (in this order) in the *CandidateQualityProperties* element: *Prices*, *StartTimes*, *StartDelays*, *EndTimes*, and *ExecutionDuration*. The *CandidateQualityProperties* element MAY be empty, or it MAY contain any subset of these elements; those that are present MUST be present in the order given above. It MAY also supply any other element it wants after these, so long as that element describes some aspect of the quality of the candidate other than those described by the standard elements described here; any such element MUST have a namespace other than one defined in this document.

```
<csg:CandidateQualityProperties>
        <eps-basic:Prices … /> ?
        <eps-basic:StartTimes … /> ?
        <eps-basic:StartDelays … /> ?
        <eps-basic:EndTimes  … /> ?
        <eps-basic:ExecutionDuration  … /> ?
```

```
        <xsd:any namespace="##other" processContents="lax" /> *
</csg:CandidateQualityProperties>
```

Any elements added to a *CandidateQualityProperties* element through its content model MUST be safely ignorable by a consumer of the containing *Candidate*. That is, the descriptions in the *CandidateQualityProperties* MUST NOT be used deceptively. Any elements added that are not in the Basic-EPS namespace MUST be permitted by the *PermittedCandidateExtension* elements in the *RequestCandidates* that caused the creation of the containing *Candidate*.

5.1.1 The *Prices* Element

The *Prices* element MUST contain a non-empty list of ranges of prices for the job, each range being described by a *Price* element.

Each *Price* element MUST relate to a single currency, and the actual estimated cost of the job is the sum of each range of the list (i.e., for each *Price* element in the list, pick an arbitrary *PriceRange* element of that *Price* and pick and arbitrary value in that range, and sum all those values after suitable currency conversions). Note that the overall range of values for a particular *Price* MAY be non-contiguous. Individual prices within the range MUST be expressed as floating-point numbers of the given currency unit, and the currency unit of a range MUST be described as a URI. Where a real currency is used, the URI that indicate it MUST be of the form “currency:*ThreeLetterCode*” where *ThreeLetterCode* is any currency code described by ISO 4217. (Thus, US dollars would be written as “currency:USD”.) All prices expressed using other URIs must be in some kind of abstract currency unit, whose specification lies outside the scope of this document.

The content of the *PriceRange* element which MAY consist of mixed strings and elements, if present, MUST be used to describe the origin of the particular charge associated with that *PriceRange*. This specification does not ascribe any particular meaning to this content, and consumers of the candidate MAY ignore it. The *PriceRange* element MUST have two attributes, *from* and *to*, that describe the range of prices in that particular range. The *from* attribute MUST NOT be greater than the *to* attribute. The *from* attribute MAY be equal to the *to* attribute. Either attribute MAY be negative; this MUST describe the case where the client of the BES is paid for their custom.

When a Basic-EPS wishes to return disjoint sets, it MUST do so by returning multiple candidates.

If there is no *Prices* element, this MUST be interpreted as a statement that the price of the job is unknown.

Pseudo-schema for the *Prices* element:

```
<eps-basic:Prices>
        <eps-basic:Price eps-basic:currency="xsd:anyURI">
                <eps-basic:PriceRange eps-basic:from="xsd:double" eps-basic:to="xsd:double">
                        <xsd:any##other> * | xsd:string
                </eps-basic:PriceRange> +
        </eps-basic:Price> +
</eps-basic:Prices>
```

5.1.2 The *StartTimes* Element

The *StartTimes* element that MUST contain a non-empty set of ranges of start times for the job.

Each contiguous component of the range MUST be expressed as a *StartAt* element, which MUST have no content and which MUST have two attributes that describe the endpoints of that particular contiguous range, *from* and *to*. Within a particular *StartAt* element, the *from* attribute MUST NOT specify an instant after the *to* attribute. The overall set of ranges MAY be non-contiguous.

Basic-EPS instances SHOULD take care to not generate *StartTimes*, *StartDelays*, *EndTimes* and *ExecutionDuration* elements that specify inconsistent requirements on the job’s execution times.

Pseudo-schema for the *StartTimes* element:

```
<eps-basic:StartTimes>
        <eps-basic:StartAt eps-basic:from="xsd:dateTime" eps-basic:to="xsd:dateTime" /> +
</eps-basic:StartTimes>
```

5.1.3 The *StartDelays* Element

The *StartDelays* element that MUST contain a non-empty set of ranges of time delays between when the job is submitted to the Basic Execution Service and when it will start executing.

The overall set of ranges MAY be non-contiguous. Individual delay ranges MUST be expressed as *DelayRange* elements, each of which describing a contiguous range of seconds (as floating-point numbers) that the particular delay might last for. Each *DelayRange* element MUST have two attributes, *from* and *to*, that specify the (inclusive) boundaries of the range. The *from* attribute MUST NOT be greater than the *to* attribute.

Basic-EPS instances SHOULD take care to not generate *StartTimes*, *StartDelays*, *EndTimes* and *ExecutionDuration* elements that specify inconsistent requirements on the job's execution times.

Pseudo-schema for the *StartDelays* element:

```
<eps-basic:StartDelays>
        <eps-basic:DelayRange eps-basic:from="xsd:double" eps-basic:to="xsd:double" /> +
</eps-basic:StartDelays>
```

5.1.4 The *EndTimes* Element

The *EndTimes* element that MUST contain a non-empty set of ranges of end times for the job.

Each range MUST be expressed as an *EndAt* element and be contiguous, though the overall set of ranges MAY be non-contiguous. Each *EndAt* element MUST have no content and MUST have two attributes, *from* and *to*, that describe the endpoints of that particular contiguous range. Within a particular *EndAt* element, the *from* attribute MUST NOT specify an instant after the *to* attribute.

Basic-EPS instances SHOULD take care to not generate *StartTimes*, *StartDelays*, *EndTimes* and *ExecutionDuration* elements that specify inconsistent requirements on the job's execution times.

Pseudo-schema for the *EndTimes* element:

```
<eps-basic:EndTimes>
        <eps-basic:EndAt eps-basic:from="xsd:dateTime" eps-basic:to="xsd:dateTime" /> +
</eps-basic:EndTimes>
```

5.1.5 The *ExecutionDuration* Element

The *ExecutionDuration* element that MUST contain a non-empty set of ranges of delays between when the job starts executing and when the job finishes executing.

The set of ranges MAY be non-contiguous. Individual delay ranges MUST be expressed as a *RuntimeRange* element which MUST describe a contiguous range of floating-point numbers of seconds via two attributes, and MUST have no content. Each *RuntimeRange* element MUST have two attributes, *from* and *to*, that specify the (inclusive) boundaries of the range. The *from* attribute MUST NOT be greater than the *to* attribute.

Basic-EPS instances SHOULD take care to not generate *StartTimes*, *StartDelays*, *EndTimes* and *ExecutionDuration* elements that specify inconsistent requirements on the job's execution times.

Pseudo-schema for the *ExecutionDuration* element:

```
<eps-basic:ExecutionDuration>
        <eps-basic:RuntimeRange eps-basic:from="xsd:double" eps-basic:to="xsd:double" /> +
</eps-basic:ExecutionDuration>
```

5.2 Standard Interpretation of the Candidate Execution Plan

A Candidate Execution Plan is intended to be a description of a JSDL job description and where the Basic-EPS believes (in good faith) it to be a good idea to submit the JSDL job description. Moreover, the Basic-EPS believes in good faith (but does not warrant) that if the JSDL job description is submitted within the range of times described by the validity period, then the quality-of-service described within the *CandidateQualityParameters* element will be applied to the job. No guarantee whatsoever is made if the JSDL job description described in the Candidate Execution Plan is submitted to the indicated Basic-EPS outside the given validity period.

If a client cannot understand all the extension *CandidateQualityParameters* elements (it SHOULD understand the standard ones outlined above) it MAY ignore the ones it does not understand. If a client cannot understand all the extension elements in the outer *Candidate* element (it MUST understand the standard ones outlined above) it SHOULD ignore the Candidate Execution Plan.

5.3 Example Candidate Execution Plan

This candidate execution plan describes an offer to run BLAST on the BES at http://some.service.com/Execution (using the back-end service bigiron.service.com) at zero cost for 4-5 minutes with an estimated queuing time of up to 2 minutes, so long as the client submits over the 2006 Christmas period (GMT).

To aid reading of this example, the bold elements are those that are required because this is a candidate produced by a CSG, and the italic elements are those that are because this is a CEP produced by a Basic-EPS.

```
<csg:Candidate
        xmlns:csg="http://schemas.ogf.org/rss/2007/03/csg"
        xmlns:eps-basic="http://schemas.ogf.org/rss/2007/03/eps/basic"
        xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl"
        xmlns:jsdl-hpcp="http://schemas.ggf.org/jsdl/2006/07/jsdl-hpcp"
        xmlns:bes="http://schemas.ggf.org/bes/2006/08/bes-factory"
        xmlns:wsa="http://www.w3.org/2005/08/addressing"
        xmlns:wsaw="http://www.w3.org/2005/03/addressing/wsdl"
        xmlns:serv="http://some.service.com/private">
    <csg:CandidateDescription>
        <jsdl:JobDefinition>
            <jsdl:JobDescription>
                <jsdl:JobIdentification>
                    <jsdl:JobName>C.E.P. Example</jsdl:JobName>
                </jsdl:JobIdentification>
                <jsdl:Application>
                    <jsdl:ApplicationName> BLAST </jsdl:ApplicationName>
                    <jsdl-hpcp:HPCProfileApplication>
                        <jsdl-hpcp:Output>blast_out.txt</jsdl-hpcp:Output>
                        <jsdl-hpcp:Error>blast_err.txt</jsdl-hpcp:Error>
                    </jsdl-hpcp:HPCProfileApplication>
                </jsdl:Application>
            </jsdl:JobDescription>
        </jsdl:JobDefinition>
        <eps-basic:BESReference>
            <wsa:Address> http://some.service.com/Execution </wsa:Address>
            <wsa:Metadata>
                <wsaw:InterfaceName>
                    bes:BESFactoryPortType
                </wsaw:InterfaceName>
            <wsa:Metadata>
            <wsa:ReferenceParameters xmlns:serv="http://some.service.com/private">
```

```
                        <serv:Backend> bigiron.service.com </serv:Backend>
                    </wsa:ReferenceParameters>
                </eps-basic:BESReference>
            </csg:CandidateDescription>
            <csg:CandidateQualityParameters>
                <eps-basic:Prices>
                    <eps-basic:Price eps-basic:currency="currency:USD">
                        <eps-basic:PriceRange eps-basic:from="0" eps-basic:to="0" />
                    </eps-basic:Price>
                </eps-basic:Prices>
                <eps-basic:StartDelay>
                    <eps-basic:DelayRange eps-basic:from="0" eps-basic:to="120" />
                </eps-basic:StartDelay>
                <eps-basic:ExecutionDuration>
                    <eps-basic:RuntimeRange eps-basic:from="240" eps-basic:to="300" />
                </eps-basic:ExecutionDuration>
            </csg:CandidateQualityParameters>
            <csg:CandidateValidity csg:from="2006-12-24T00:00:00Z"
                    csg:to="2006-12-25T23:59:59Z"/>
</csg:Candidate>
```

6. Concrete Rendering of the Basic Execution Planning Service

Although the abstract interface of the Basic-EPS is based on the returning of an ordered set of Candidate Execution Plan documents (CEPs), practical implementations of the Basic-EPS must do this in such a way as to allow potentially large sets of CEPs as a result. This means that all renderings of the service into some concrete binding MUST do so in a way that supports partial transfers of the ordered set of CEPs. This is because it is possible for an Basic-EPS to be acting as a front-end to a cluster of thousands of machines, and the majority of web-service hosting engines are not capable of marshalling arbitrarily large XML documents.

When building a service to bridge between renderings, implementers should remember that it is the abstract interface that must be matched. A direct one-to-one matching of actual calls is not required, and nor is any interoperability of the internal state model of any rendering.

6.1 Conceptual Operation of the Basic-EPS

Conceptually, when the rendering framework does not support partial transfers directly, the rendering of a Basic-EPS SHOULD add this in through the introduction of an Iterable Set Service (ISS). The ISS exists to allow the transfer of large lists of candidates, a segment at a time. The manner in which it does this is required to be specified in the particular rendering.

When a rendering of the Basic-EPS decides to not transfer its entire set of Candidate elements in the initial response to a request, it should instead return a reference to an ISS that manages the transfer. This allows transfers to be done in manageable pieces, much as internet search engines permit going through the results of a search one page at a time.

6.2 Specialization of the Basic-EPS to WS-RF

The basic model of the Basic-EPS as a WS-Resource[WSR] (termed Basic-EPS-RF) shall be that either it responds to the query directly with the ordered set of *Candidate*s that satisfy the query, or it returns an EPR to a separate WS-RF resource that describes an iterable ordered set of CEPs using an ISS (termed ISS-RF). The iterable ordered set resource only requires a single operation, which is used to get the next *n* items from the ordered set, which implies that the ordered set conceptually requires some kind of cursor in its internal state.

All SOAP elements, operations and properties designated with the prefix “eps-wsrf:” shall be qualified names in the following namespace (unless otherwise required by some other standard):

```
http://schemas.ogf.org/rss/2007/03/eps/wsrf
```

Implementations that do not support the ISS interface MUST NOT respond with more than one Candidate Execution Plan to any request.

6.2.1 Operations of the Basic-EPS-RF

The Basic-EPS-RF MUST support the operation *GetCandidates*. The input for the *GetCandidates* operation MUST be a *GetCandidatesRequest* document, and the output for the *GetCandidates* operation MUST be a *GetCandidatesResponse* document.

The Basic-EPS-RF MUST support the read operations described in WS-ResourceProperties[WSRP][1]. Implementations MAY support the operations that update properties but SHOULD check whether the client is authorized to do so if they do support those operations. See §0 for a description of the properties model that MUST be used by the properties operations.

The *GetCandidatesRequest* document MUST contain a *RequestCandidates* element that follows one of the profiles supported by the Basic-EPS (minimally the Basic-EPS profile itself) and MAY contain a *MaxCandidatesPerTransfer* element, a *MaxCandidatesTotal* element, a *MaxResponseOctets* element, and an *IterableSetLifetime* element. Of the elements in addition to *RequestCandidates*, the *MaxCandidatesPerTransfer* and *IterableSetLifetime* MUST be supported, and implementations MUST NOT fault if the others are present, but MAY ignore them.

- The *CandidateOrdering* element inside the *RequestCandidates* element MUST have a "*dialect*" attribute (with a content model permitting any URI) and MUST allow any single element as content. The *dialect* describes which ordering language is to be used to order the set of candidates, and the content of the *CandidateOrdering* element MUST follow the requirements of the language described by the *dialect*; the SCOL (see Section 0) is the only language that MUST be supported.
- The *MaxCandidatesPerTransfer* element MUST have a positive-integer content model, and MUST describe the maximum number of Candidate Execution Plans that the user wishes to receive per SOAP response. If not present, the Basic-EPS-RF service SHOULD use the *MaxCandidatesPerTransfer* property to define the limit.
- The *MaxCandidatesTotal* element MUST have a positive-integer content model, and MUST describe the maximum number of Candidate Execution Plans that the user wishes to receive overall (permitting implementations to clean up resources more efficiently when that many CEPs have been retrieved). If not present, the Basic-EPS-RF SHOULD NOT place any limit on the number of CEPs to be returned, and transfers MUST be done in stages using an ISS-RF unless the total number of responses is less than the value described within the *MaxCandidatesPerTransfer* element.
- The *MaxResponseOctets* element MUST have a positive integer content model, and MUST describe the maximum number of octets (bytes) that the user wishes to receive in any SOAP response that returns more than one *Candidate* elements. If not present, the Basic-EPS-RF service SHOULD use the *MaxResponseOctets* property of itself to define the limit, if present. If that is also absent, no statement is made by this specification about the number of octets in any response. It is RECOMMENDED that neither the Basic-EPS-RF nor the ISS-RF transfer very large numbers of Candidate Execution Plans in a single message in order that clients of the service can process the messages using implementations based on W3C DOM.
- The *IterableSetLifetime* element MUST have an XML Schema time duration content model, and MUST describe the preferred lifespan of the ISS-RF resource if one is cre-

[1] That is the mandatory operation *GetResourceProperty* MUST be supported, and the operations *GetResourcePropertyDocument*, *GetMultipleResourceProperties* and *QueryResourceProperties* SHOULD be supported.

ated. If unspecified, the EPS SHOULD use the duration from its *DefaultIterableSetLifetime* property.

```
<soap:Envelope>
    <soap:Header>
        <wsa:Action>
            http://schemas.ogf.org/rss/2007/03/eps/wsrf/GetCandidates
        </wsa:Action>
        …
    </soap:Header>
    <soap:Body>
        <eps-wsrf:GetCandidatesRequest>
            <csg:RequestCandidates> … </csg:RequestCandidates>
            <eps-wsrf:MaxCandidatesPerTransfer>
                xsd:positiveInteger
            </eps-wsrf:MaxCandidatesPerTransfer> ?
            <eps-wsrf:MaxCandidatesTotal>
                xsd:positiveInteger
            </eps-wsrf:MaxCandidatesTotal> ?
            <eps-wsrf:MaxResponseOctets>
                xsd:positiveInteger
            </eps-wsrf:MaxResponseOctets> ?
            <eps-wsrf:IterableSetLifetime>
                xsd:duration
            </eps-wsrf:IterableSetLifetime> ?
        </eps-wsrf:GetCandidatesRequest>
    </soap:Body>
</soap:Envelope>
```

The *GetCandidatesResponse* document MUST contain a sequence of zero, one or more *Candidate* elements, each of which MUST match the profile for the input document selected, followed optionally by an *IterableSetReference* element. Clients of the Basic-EPS-RF MUST support the receipt of an *IterableSetReference*.

- There MUST NOT be more candidates than the value described in the *MaxCandidatesPerTransfer* or in the *MaxCandidatesTotal* elements of the *GetCandidatesRequest* document (if either are present). There MUST NOT be more candidates than the value described in the *MaxCandidatesPerTransfer* property of the Basic-EPS-RF (if defined).
- The *IterableSetReference* element MUST contain a WS-Addressing Endpoint Reference to the ISS-RF instance from which to retrieve all remaining Candidate Execution Plans should be retrieved.

```
<soap:Envelope>
    <soap:Header>
        <wsa:Action>
            http://schemas.ogf.org/rss/2007/03/eps/wsrf/GetCandidatesResponse
        </wsa:Action>
        …
    </soap:Header>
    <soap:Body>
        <eps-wsrf:GetCandidatesResponse>
            <csg:Candidate> … </csg:Candidate> *
            <eps-wsrf:IterableSetReference>
                wsa:EndpointReferenceType
            <eps-wsrf:IterableSetReference> ?
        </eps-wsrf:GetCandidatesResponse>
```

```
        </soap:Body>
</soap:Envelope>
```

The following kinds of WS-BaseFaults[WSBF] MUST be supported by the *GetCandidates* operation: *UnsupportedCSGProfileFault*, *UnsupportedOrderingLanguageFault*, *UnableToGenerateCandidatesFault*, and *BadCandidateCardinalityRulesFault*. All must be transferred using the rules set out in WS-BaseFaults. The interpretations of the faults shall be as follows:

- An *UnsupportedCSGProfileFault* MUST be indicated when the client supplies a *RequestCandidates* element that does not match any supported profile. The content model of this fault (other than that it MUST be a WS-BaseFault with a particular name) is not defined.
- An *UnsupportedOrderingLanguageFault* MUST be indicated when client asks to order the Candidate Execution Plans using a language that is not supported or if the content of the Ordering element is malformatted. This fault MUST NOT be indicated if the *Ordering* contains a valid SCOL document. The content model of this fault (other than that it MUST be a WS-BaseFault with a particular name) is not defined.
- A *BadCandidateCardinalityRulesFault* SHOULD be indicated if the client requests a number of rules per transfer that the Basic-EPS-RF does not support. The content model of this fault (other than that it MUST be WS-BaseFault with a particular name) is not defined.
- An *UnableToGenerateCandidatesFault* SHOULD be indicated in other failure cases. The content model of this fault (other than that it MUST be a WS-BaseFault with a particular name) is not defined, but the fault SHOULD describe what the failure was. This fault MUST NOT be generated if there are simply no possible *Candidate*s that match the *RequestCandidates.*
- The Basic-EPS-RF SHOULD throw a *ResourceUnknownFault* (from the standard WS-RF fault set) if it receives a request for a Basic-EPS-RF resource that it does not know about.

6.2.2 Operations of the Iterable Set Service (ISS-RF)

The ISS-RF MUST support the operation *GetMoreCandidates*, MUST support the operations of WS-ResourceLifetime[WSRL], and MUST support the read operations described in WS-ResourceProperties[WSRP]; supporting the update operations described in WS-ResourceProperties is NOT RECOMMENDED.

The input for the *GetMoreCandidates* operation shall be a *GetMoreCandidatesRequest* document, and the output for the operation shall be a *GetMoreCandidatesResponse* document. The operation MAY throw a *BadCandidateCardinalityRulesFault*, but only in the circumstances described below, and SHOULD throw an *UnableToGenerateCandidatesFault* in other cases; both faults MUST be syntactically the same as those faults as thrown by the Basic-EPS-RF.

- The *GetMoreCandidatesRequest* document MUST either contain a *Count* element or be empty.
- The *Count* element MUST have a positive integer content model. It MUST describe the maximum number of *Candidate* elements to retrieve in this operation. If the *Count* exceeds the maximum number of candidates per transfer (as described in the *MaxCandidatesPerTransfer* property) a *BadCandidateCardinalityRulesFault* MUST be thrown; if the property is not present, the operation must not generate this fault. If fewer *Candidate* elements remain in the ordered set than the *Count* requests, all remaining *Candidate* elements MUST be returned (unless a fault is thrown instead).

```
<soap:Envelope>
    <soap:Header>
        <wsa:Action>
            http://schemas.ogf.org/rss/2007/03/eps/wsrf/GetMoreCandidates
        </wsa:Action>
```

```
            …
        </soap:Header>
        <soap:Body>
            <eps-wsrf:GetMoreCandidatesRequest>
                <eps-wsrf:Count> xsd:positiveInteger </eps-wsrf:Count> ?
            </eps-wsrf:GetMoreCandidatesRequest>
        </soap:Body>
</soap:Envelope>
```

The *GetMoreCandidatesResponse* document MUST contain zero, one or more *Candidate-ExecutionPlan* elements (as described previously).

- The number of *Candidate* elements returned MUST NOT be zero if candidates remain in the underlying ordered set at the point of operation invocation. If the *Count* element was specified in the *GetMoreCandidatesRequest* and was legal and at least that many candidates remain in the ordered set, exactly that many *Candidate* elements SHOULD be in the response (subject to the restriction that the limit on the overall size of response given in the *MaxResponseOctets* property SHOULD NOT be exceeded).

```
<soap:Envelope>
        <soap:Header>
            <wsa:Action>
                http://schemas.ogf.org/rss/2007/03/eps/wsrf/GetMoreCandidatesResponse
            </wsa:Action>
            …
        </soap:Header>
        <soap:Body>
            <eps-wsrf:GetMoreCandidatesResponse>
                <csg:Candidate> … </csg:Candidate> *
            </eps-wsrf:GetMoreCandidatesResponse>
        </soap:Body>
</soap:Envelope>
```

6.2.3 Properties

The following properties SHOULD be supported by the Basic-EPS-RF and the ISS-RF. If any property is supported, it SHOULD be readable using the operations described in WS-ResourceProperties; the *GetResourcePropertyDocument* operation MUST be supported. Implementations MAY support writing of a property using any operation described WS-ResourceProperties except where otherwise stated. The ISS-RF MUST support the properties described by WS-ResourceLifetime. Both the Basic-EPS-RF and the ISS-RF MUST support the properties defined by OGSA WS-RF Basic Profile v1.0[OWSR].

csg:UsageProfile (see §4.4.1)

This Basic-EPS-RF property MUST have a cardinality of one or more, and it MUST have the content model of allowing any URI. Each URI listed describes one CSG usage profile supported by the service; the URI describing the Basic-EPS profile MUST be present, but other profiles MAY also be supported. This property SHOULD NOT be writable or otherwise modifiable.

csg:SupportedOrderingLanguages (see §4.4.2)

This Basic-EPS-RF property MUST have a cardinality of one or more, and it MUST have the content model of allowing any URI. Each URI listed describes one candidate ordering language. Since the SCOL (see Section 0 below) MUST be supported, its URI MUST be the content of one of the elements that describe this property. This property SHOULD NOT be writable or otherwise modifiable.

eps-wsrf:MaxCandidatesPerTransfer

This Basic-EPS-RF and ISS-RF property MUST have a cardinality of one, and it MUST have a positive integer content model. The property MUST describe the maximum number of Candidate Execution Plans that will be sent to the client in any SOAP response. This property MAY be writable, but implementations SHOULD NOT permit clients to increase the value and MAY wish to restrict write permission to the service owner.

eps-wsrf:MaxResponseOctets

This Basic-EPS-RF and ISS-RF property MUST have a cardinality of zero or one, and it MUST have a positive integer content model. The property MUST describe the maximum number of octets (bytes) in any response containing more than one *Candidate* elements by either the Basic-EPS-RF or the ISS-RF. This property MAY be writable, but implementations SHOULD NOT permit clients to increase the value and MAY wish to restrict write permission to the service owner. If not present, the service makes no statement about the maximum size of responses.

eps-wsrf:DefaultIterableSetLifetime

This Basic-EPS-RF property MUST have a cardinality of zero or one, and it MUST have a time duration content model. The property MUST describe the default lifetime of the Iterable Set Service resource. This property MAY be writable, but implementations SHOULD restrict write permission to the service owner. If absent, it indicates that the service implementing the Basic-EPS-RF interface is making no statement about the lifetime of the services implementing the ISS-RF interface that it creates.

eps-wsrf:CandidatesTransferredSoFar

This ISS-RF property MUST have a cardinality of zero or one, and it MUST have a non-negative integer content model. The property MUST, if present, describe the number of Candidate Execution Plans retrieved from the (conceptual) underlying ordered set through the initial call to *GetCandidates* on the Basic-EPS-RF and any subsequent calls to *GetMoreCandidates* on the ISS-RF. This property MUST NOT be writable.

eps-wsrf:AnyMoreCandidatesToTransfer

This ISS-RF property MUST have a cardinality of zero or one, and MUST have a boolean content model. The property MUST, if present, describe whether there exist further *Candidate* elements to be retrieved, *i.e.*, whether further invocations of the *GetMoreCandidates* operation will perform useful work. This property MUST NOT be writable, but SHOULD be modified internally when Candidate elements are retrieved.

The pseudo-schema for the Basic-EPS-RF resource properties is:

```
<csg:UsageProfile> xsd:anyURI </csg:UsageProfile> +
<csg:SupportedOrderingLanguages> xsd:anyURI </csg:SupportedOrderingLanguages> +
<eps-wsrf:MaxCandidatesPerTransfer>
        xsd:positiveInteger
<eps-wsrf:MaxCandidatesPerTransfer>
<eps-wsrf:MaxResponseOctets> xsd:positiveInteger <eps-wsrf:MaxResponseOctets> ?
<eps-wsrf:DefaultIterableSetLifetime> xsd:duration </eps-wsrf:DefaultIterableSetLifetime> ?
```

The pseudo-schema for the ISS-RF resource properties is:

```
<eps-wsrf:MaxCandidatesPerTransfer>
        xsd:positiveInteger
</eps-wsrf:MaxCandidatesPerTransfer>
<eps-wsrf:MaxResponseOctets> xsd:positiveInteger <eps-wsrf:MaxResponseOctets> ?
```

```
<eps-wsrf:CandidatesTransferredSoFar>
        xsd:nonNegativeInteger
</eps-wsrf:CandidatesTransferredSoFar> ?

<eps-wsrf:AnyMoreCandidatesToTransfer>
        xsd:boolean
</eps-wsrf:AnyMoreCandidatesToTransfer> ?
```

6.3 Specialization of the EPS to WS-Transfer/WS-Enumeration

The basic model of the Basic-EPS as a WS-Transfer (as described in the document at http://www.w3.org/Submission/WS-Transfer/) resource (termed Basic-EPS-XF) shall be that it responds to a query by returning an ordered sequence of *Candidate*s that satisfy the query, and that it uses WS-Enumeration (as described in http://www.w3.org/Submission/WS-Enumeration/) to transfer the CEPs.

All SOAP elements, operations and properties designated with the prefix “eps-xfer:” shall be qualified names in the following namespace (unless otherwise required by some other standard):

```
http://schemas.ogf.org/rss/2007/10/eps/transfer
```

6.3.1 Operations of the Basic-EPS-XF

The Basic-EPS-XF MUST support the operation *EnumerateCandidates*. The input for the *EnumerateCandidates* operation MUST be an *EnumerateCandidatesRequest* document, and the output for the *EnumerateCandidates* operation MUST be an *EnumerateResponse* message as defined by WS-Enumeration, where each of the elements that is a child of an *Items* element in a *PullResponse* message (as defined by WS-Enumeration) MUST be a *Candidate*.

The Basic-EPS-XF MUST also support the operation *Get* from WS-Transfer. See §6.3.2 for the description of the representation of the resource that MUST be presented. This specification does not require the support of any other operation defined by WS-Transfer.

The *EnumerateCandidatesRequest* document MUST contain a *RequestCandidates* element that follows one of the profiles supported by the Basic-EPS (minimally the Basic-EPS profile itself) and, following that, MAY contain either of the *EndTo* and *Expires* elements defined by WS-Enumeration, which MUST be interpreted with the same semantics as if they were in an *Enumerate* element (as defined by WS-Enumeration). Note however that the WS-Enumerate *Filter* element MUST NOT be present; *Candidate* filtering MUST be done via the *CandidateOrdering* element inside the *RequestCandidates* element.

```
<soap:Envelope>
    <soap:Header>
        <wsa:Action>
            http://schemas.ogf.org/rss/2007/10/eps/transfer/EnumerateCandidates
        </wsa:Action>
        <wsa:MessageID> xsd:anyURI </wsa:MessageID>
        <wsa:To> xsd:anyURI </wsa:To>
        …
    </soap:Header>
    <soap:Body>
        <eps-xfer:EnumerateCandidatesRequest>
            <csg:RequestCandidates> … </csg:RequestCandidates>
            <wsen:EndTo> wsa:EndpointReferenceType </wsen:EndTo> ?
            <wsen:Expires> [ xsd:dateTime | xsd:duration ] </wsen:Expires> ?
        </eps-xfer:EnumerateCandidatesRequest>
    </soap:Body>
</soap:Envelope>
```

If the *RequestCandidate* element inside the *EnumerateCandidatesRequest* does not match any supported profile (as listed in the *UsageProfile* elements in the Basic-EPS-XF resource representation), an *UnsupportedCSGProfileFault* fault MUST be indicated.

If the *RequestCandidate* element inside the *EnumerateCandidatesRequest* contains a candidate ordering element whose format is not supported (i.e. that is listed in one of the *SupportedOrderingLanguages* elements in the Basic-EPS-XF resource representation) or that is malformatted, an *UnsupportedOrderingLanguageFault* fault MUST be indicated. This fault MUST NOT be indicated if the *Ordering* contains a valid SCOL document.

If the *Expires* element is invalid or unsupported according to the rules outlined in WS-Enumeration, the fault indication rules laid out for that case within WS-Enumeration MUST be followed.

In other cases where candidate generation is impossible for failure reasons (as opposed to being because the number of possible candidates is zero), an *UnableToGenerateCandidatesFault* fault MUST be indicated.

All faults MUST be indicated (using SOAP 1.2[SOAP] rules[2]) via the *s12:Code/s12:Value* field being *s12:Sender*, the *s12:Code/s12:Subcode/s12:Value* field being the qualified name for the fault, and the *s12:Reason/s12:Text* field containing a human-readable description of the detailed reason for the fault. This parallels the encoding used for faults in WS-Enumeration.

6.3.2 Resource Representation of the Basic-EPS-XF

The representation of the Basic-EPS-XF MUST be an XML document that contains the following elements:

- One or more *csg:UsageProfile* elements (see §4.4.1), each with the content model of allowing any URI. Each URI listed describes one CSG usage profile supported by the service; the URI describing the Basic-EPS profile MUST be present, but other profiles MAY also be supported.
- One or more *csg:SupportedOrderingLanguages* elements (see §4.4.2), each with the content model of allowing any URI. Each URI listed describes one candidate ordering language. Since the SCOL (see Section 0 below) MUST be supported, its URI MUST be the content of one of the elements that describe this property.

No other elements are required to be supported.

```
<csg:UsageProfile> xsd:anyURI </csg:UsageProfile> +
<csg:SupportedOrderingLanguages> xsd:anyURI </csg:SupportedOrderingLanguages> +
```

7. Simple Candidate Ordering Language

To facilitate interoperability, every CSG should support at least the simple candidate ordering language described in this section. The Simple Candidate Ordering Language (SCOL) is an XML language that describes how to assign a score to a document. SCOL has the URL:

```
http://schemas.ogf.org/rss/2006/12/scol
```

This is also the URL of the XML namespace of the terms in SCOL.

[2] SOAP 1.1 fault bindings for all faults must indicate that they are client faults using the *faultcode* element and provide an informative textual description in the *faultstring* element. This specification does not further discuss them. Use of SOAP 1.1 is NOT RECOMMENDED.

7.1 Organizing Principles of SCOL for Candidate Generation

When any CSG is asked to generate candidates satisfying some request that contains a *CandidateOrdering* term that is written using the SCOL definition (presented here), the CSG:

1. MUST first internally generate a set of *Candidate* elements that satisfy the *Input-Description* from the request (this step otherwise being not described by SCOL; it is an intentionally implementation-dependent process),
2. MUST then evaluate the SCOL expression (independently, once per *Candidate*) within the context of each *Candidate* to generate a floating-point valuation for each *Candidate*, discarding those *Candidate*s for whom the evaluation fails,
3. MUST then sort the remaining *Candidate*s by the valuations such that the *Candidate*s with the smallest valuations come first; this sorted set of *Candidate*s MUST form the contents of the ordered set that is the result of the *getCandidates* operation. When two *Candidate*s have the same valuation, the implementation MAY pick any ordering of the two so long the following two conditions hold: all *Candidate*s with smaller valuations MUST precede both of them, and all *Candidate*s with larger valuations MUST succeed both of them.

7.1.1 Implementation Notes

It is possible to implement SCOL through translation into another query language, e.g., XQuery[XQ]. When doing this, it should be noted that, provided the translation is correct and all embedded XPath 2.0[XP] terms are themselves syntactically correct, the resulting operation can be guaranteed to be both inherently secure and guaranteed to terminate in finite time.

It should also be noted that SCOL has been designed such that an implementations of a CSG that can also perform partial transfers (such as the Basic-EPS-RF definition entails) can include distributed merge sorts without requiring the branches of the resulting tree of CSGs to maintain a large amount of state.

7.2 SCOL Syntax and Semantics

SCOL defines a set of terms, each of which is an XML element. An XML document matches the SCOL syntax if its outermost element is an *OrderFunction* element whose content matches the syntax of any SCOL term, and the evaluation of the *OrderFunction* (with respect to a context document, *i.e.*, a candidate) is the evaluation of the contained term. The evaluation of a term (within the context defined by the document to have a value assigned to it) is defined as follows. Note that the evaluation of any syntactically invalid SCOL term MUST fail.

Note: SCOL omits terms for performing subtraction and division as these can be constructed from *Sum* and *Negate* or *Product* and *Power* terms respectively.

Syntax:

```
<OrderFunction> <Term ... /> </OrderFunction>
```

Note that the *Term* element is abstract. All concrete terms are subtypes of its type and in the same substitution group as it, so anywhere that accepts *Term* will accept any kind of *Term*.

7.2.1 Term: *Constant*

The *Constant* term MUST contain a single *Value* element. The *Value* element MUST contain a floating-point number. The *Constant* term MUST evaluate to the value of the content of the contained *Value* element. A syntactically-valid *Constant* term MUST NOT evaluate to a failure.

Syntax:

```
<Constant> <Value> xsd:double </Value> </Constant>
```

Example: This value of this *Constant* is 42.

```
<Constant><Value>42</Value></Constant>
```

7.2.2 Term: *Sum*

The *Sum* term MUST contain at least two nested terms. It MUST evaluate to the sum of the evaluations of the nested terms. If any nested term evaluates to a failure, the *Sum* MUST evaluate to a failure.

Syntax:

```
<Sum> <Term … /> + </Sum>
```

Example: The value of this *Sum* is the sum of the values of the three contained *Constant* terms, *i.e.*, 10.

```
<Sum>
    <Constant><Value>2</Value></Constant>
    <Constant><Value>3</Value></Constant>
    <Constant><Value>5</Value></Constant>
</Sum>
```

Example: The value of this term is the difference between the two contained *Constant* terms, *i.e.*, 70.

```
<Sum>
    <Constant><Value>99</Value></Constant>
    <Negate>
        <Constant><Value>29</Value></Constant>
    </Negate>
</Sum>
```

7.2.3 Term: *Product*

The *Product* term MUST contain at least two nested terms. It MUST evaluate to the product of the evaluations of the nested terms. If any nested term evaluates to a failure, the *Product* MUST evaluate to a failure.

Syntax:

```
<Product> <Term … /> + </Product>
```

Example: The value of this *Product* is the product of the values of the three contained *Constant* terms, *i.e.*, 30.

```
<Product>
    <Constant><Value>2</Value></Constant>
    <Constant><Value>3</Value></Constant>
    <Constant><Value>5</Value></Constant>
</Product>
```

Example: The value of this term is the quotient of the two contained *Constant* terms, *i.e.*, 11.

```
<Product>
    <Constant><Value>99</Value></Constant>
    <Power exponent="-1">
        <Constant><Value>9</Value></Constant>
    </Power>
</Product>
```

7.2.4 Term: *Power*

The *Power* term MUST contain one nested term, and MUST have one floating-point attribute, "*exponent*". It MUST evaluate to the value of the nested term raised to the power of the value of the *exponent* attribute. If the nested term evaluates to a failure, the *Power* MUST evaluate to a failure. If the exponent attribute is a non-integral value, the *Power* MUST evaluate to a failure if the nested term evaluates to a negative number.

Syntax:

```
<Power exponent="xsd:double" > <Term … /> </Power>
```

Example: The value of this *Power* is the square of the value of the contained *Constant* term, *i.e.*, 11.

```
<Power exponent="2">
        <Constant><Value>11</Value></Constant>
</Power>
```

Example: The value of this term is the quotient of the two contained *Constant* terms, *i.e.*, 11.

```
<Product>
        <Constant><Value>99</Value></Constant>
        <Power exponent="-1">
                <Constant><Value>9</Value></Constant>
        </Power>
</Product>
```

7.2.5 Term: *Negate*

The *Negate* term MUST contain one nested term. It MUST evaluate to the negation of the value of the nested term. If the nested term evaluates to a failure, the *Negate* MUST evaluate to a failure.

Syntax:

```
<Negate> <Term … /> </Negate>
```

Example: The value of this *Negate* is the negation of the value of the contained *Constant* term, *i.e.*, -42.

```
<Negate>
        <Constant><Value>42</Value></Constant>
</Negate>
```

Example: The value of this term is the difference between the two contained *Constant* terms, *i.e.*, 70.

```
<Sum>
        <Constant><Value>99</Value></Constant>
        <Negate>
                <Constant><Value>29</Value></Constant>
        </Negate>
</Sum>
```

7.2.6 Term: *Log*

The *Log* term MUST contain one nested term, and MUST permit the specification of one floating-point attribute, "*base*". If the *base* attribute is omitted, it MUST be interpreted to be *e*, the base of natural logarithms (or as near an approximation as the implementation can achieve). The *Log* term MUST evaluate to the logarithm (to the given *base*) of the value of the nested term. The *Log*

term MUST evaluate to a failure if the nested term evaluates to a failure or the base is not a positive number.

Syntax:

```
<Log base="xsd:double" ? > <Term … /> </Log>
```

Example: The value of this *Log* is the logarithm (to base 2) of the value of the contained *Constant* term, *i.e.*, 7.

```
<Log base="2">
    <Constant><Value>128</Value></Constant>
</Log>
```

7.2.7 Term: *Exp*

The *Exp* term MUST contain one nested term, and MUST permit the specification of one floating-point attribute, "*base*". If the *base* attribute is omitted, it MUST be interpreted to be *e*, the base of natural logarithms (or as near an approximation as the implementation can achieve). The *Exp* term MUST evaluate to the value of the *base* attribute raised to the power of the value of the nested term. The *Exp* term MUST evaluate to a failure if the nested term evaluates to a failure or if the value of the *base* attribute is negative.

Syntax:

```
<Exp base="xsd:double" ? > <Term … /> </Exp>
```

Example: The value of this *Exp* is the antilog (to base 10) of the value of the contained *Constant* term, *i.e.*, 1000.

```
<Exp base="10">
    <Constant><Value>3</Value></Constant>
</Exp>
```

7.2.8 Term: *Abs*

The *Abs* term MUST contain one nested term. It MUST evaluate to the absolute value of the value of the nested term (*i.e.*, with the sign of the value negated if and only if it is negative). The *Abs* term MUST evaluate to a failure if the nested term evaluates to a failure.

Syntax:

```
<Abs> <Term … /> </Abs>
```

Example: The value of this *Abs* is the positive magnitude of the value of the contained *Constant* term, *i.e.*, 19.

```
<Abs>
    <Constant><Value>-19</Value></Constant>
</Abs>
```

7.2.9 Term: *Select*

The *Select* term MUST contain a single *Path* element, and must permit optional "*baseInstant*" and "*operation*" attributes. The *Path* element MUST contain an XPath 2.0[XP] expression. If present, the "*baseInstant*" attribute MUST contain a reference to a particular time instant. The "*operation*" attribute MUST be one of the strings "*first*", "*count*", "*exists*" and "*total*" and MUST, if absent, be treated as if it was present with the value "*first*". The behavior of the terms according to the operations is as follows:

first If the *operation* is "*first*", the *Select* term MUST evaluate to the value of the text content of the first node selected by applying the XPath expression to the context document (*i.e.*, the Candidate Execution Plan). If the *baseInstant* attribute is present, the value of the

Select MUST be the number of seconds after the *baseInstant* that the time instant value selected refers to. If the *baseInstant* attribute is absent, the value of the *Select* MUST be the floating-point interpretation of the text node. The *Select* term MUST evaluate to a failure if no nodes are selected, if the text content of the first selected node cannot be interpreted as a time instant when the *baseInstant* attribute is present, or if the text content of the selected node cannot be interpreted as a floating-point number when the *baseInstant* attribute is absent.

count If the *operation* is "*count*", the *Select* term MUST evaluate to the number of nodes selected by applying the XPath expression to the context document. The content of those nodes SHOULD be ignored.

exists If the *operation* is "*exists*", the *Select* term MUST evaluate to 1 if the number of nodes selected by applying the XPath expression to the context document is at least 1, and 0 otherwise. The content of those nodes SHOULD be ignored.

total If the *operation* is "*total*", the *Select* term MUST evaluate to the sum of the value of each text node selected by applying the XPath expression to the context document. The value of each selected node MUST be computed according to the same rules as for the "*first*" operation (*i.e.*, as seconds from *baseInstant* if that is present, or as floating-point values if *baseInstant* is absent). The *Select* term MUST evaluate to 0 if no nodes are matched. The *Select* term MUST evaluate to a failure if the text content of any selected node cannot be interpreted as a time instant (when *baseInstant* is present) or a floating-point value (when *baseInstant* is absent).

Select term syntax:

```
<Select operation="first|count|exists|total" ? baseInstant="xsd:dateTime" ? >
        <Bind namespace="xsd:anyURI" prefix="xsd:NCName" /> *
        <Path> xsd:string </Path>
</Sum>
```

Example: The value of this *Select* is the total numeric value of the nodes selected from the context document; the nodes selected are Price nodes directly within a QoS node.

```
<Select operation="total">
        <Path> //*:CandidateQualityProperties/*:Prices/*:Price </Path>
</Select>
```

Example: The value of this *Select* is the numeric value of the first node selected from the context document; the node selected is an exact count of CPUs requested in a JSDL job description.

```
<Select operation="first">
        <Bind namespace="http://schemas.ggf.org/jsdl/2005/11/jsdl" prefix="jsdl" />
        <Path>
                //jsdl:JobDescription/jsdl:Resources/jsdl:TotalCPUCount/jsdl:Exact[not(@epsilon)]
        </Path>
</Select>
```

7.2.10 Term: *Bound*

The *Bound* term MUST contain one nested term, and MUST permit either or both of the attributes "*lowerBound*" and "*upperBound*". Both the *lowerBound* and the *upperBound* MUST, if present, be interpreted as floating-point values. The *Bound* term MUST evaluate to the value that the nested term evaluates to unless the *lowerBound* (if present) is a larger value or the *upperBound* (if present) is a smaller value. The *Bound* term MUST evaluate to a failure if the nested term evaluates to a failure, if the *lowerBound* (when present) is larger than the value that the nested term evaluates to, or if the *upperBound* (when present) is smaller than the value that the nested term evaluates to.

Syntax:

```
<Bound lowerBound="xsd:double" ? upperBound="xsd:double" ? > <Term … /> </Bound>
```

Example: The value of this *Bound* is the value of the *Sum* unless that is either less than 1 or more than 42.

```
<Bound lowerBound="1" upperBound="42">
    <Sum>…</Sum>
</Bound>
```

7.2.11 Term: *OneOf*

The *OneOf* term MUST contain at least one nested term. The nested terms MUST be evaluated in document order, and the value of the *OneOf* term MUST be the value of the first nested term that does not evaluate to failure; all subsequent terms MAY be ignored. The *OneOf* term MUST evaluate to failure if and only if all the nested terms evaluate to failure.

Syntax:

```
<OneOf> <Term … /> + </OneOf>
```

Example: The value of this *OneOf* is the value of the first contained *Bound* that is satisfied.

```
<OneOf>
    <Bound … />
    <Bound … />
</OneOf>
```

Example: The value of this *OneOf* is the value of the *Bound* if that is satisfied, or the value of the *Constant* otherwise. This never fails (as long as the *Constant* is syntactically correct).

```
<OneOf>
    <Bound … />
    <Constant … />
</OneOf>
```

7.3 Example *OrderFunction*

This example (which should be evaluated within the context of a Candidate Execution Plan) defines the valuation of a candidate to be the total price per CPU (which must be in US dollars per CPU) which must be kept within acceptable bounds (no more than $42/CPU). The example includes extra spaces for greater clarity.

```
<OrderFunction>
    <Bound upperBound="42">
        <Product>
            <Select operation="total">
                <!-- Gets a price, assuming we've got an exact one. -->
                <Bind namespace="http://schemas.ogf.org/rss/2007/03/csg"
                        prefix="csg"/>
                <Bind namespace="http://schemas.ogf.org/rss/2007/03/eps/basic"
                        prefix="eps"/>
                <Path>
                    // csg:CandidateQualityProperties
                    // eps:Price[@currency="currency:USD"]
                    /  eps:PriceRange[@from=@to] @from
                </Path>
            </Select>
            <Power exponent="-1">
                <Select operation="first">
                    <!-- Gets the # of CPUs to be allocated to the job. -->
```

```
                                <Bind namespace="http://schemas.ogf.org/rss/2007/03/csg"
                                        prefix="csg"/>
                                <Bind namespace="http://schemas.ggf.org/jsdl/2005/11/jsdl"
                                        prefix="jsdl"/>
                                <Path>
                                    // csg:CandidateDescription
                                    // jsdl:Resources
                                    /  jsdl:TotalCPUCount
                                    /  jsdl:Exact
                                </Path>
                            </Select>
                        </Power>
                    </Product>
                </Bound>
</OrderFunction>
```

7.4 Extending SCOL

Uses of SCOL where an extension is used MUST use a different URI to describe the language. All extensions MUST be done by introducing new elements. All extension term elements MUST be members of the *Term* substitution group, and the type of every extension term element MUST be an extension of the *Term* type, and hence must be complex (*i.e.*, element) content. All extensions MUST also introduce a new top-level element which must be used instead of the *OrderFunction* element.

8. Security Considerations

Since execution plans are intended for use by a specific user, implementations of the EPS SHOULD permit the specification of a security context that includes the identity of the final user as part of any invocation of the *GetCandidates* operation. They SHOULD (where possible) verify that, for each Candidate Execution Plan, the Basic Execution Service referenced by it will accept the JSDL *JobDefinition* described by the plan under the identity of the final user. This is particularly important when the EPS is delegating the generation of some Candidate Execution Plans to other EPS instances; in that case, it SHOULD also delegate the users' identities to the inner EPS instances.

It is also commonly the case that information systems provide different responses to different users. Because of this, it is important for the identity provided within the security context to be used when accessing information as part of creating a Candidate Execution Plan.

As there may be user-specific information within a Candidate Execution Plan, implementations SHOULD take steps to ensure the confidentiality of the ordered set of plans. This may be by only permitting access to the ISS from the same security context that performed the initial invocation of the *GetCandidates* operation on the EPS, and ensuring that communications between the client and the EPS and ISS go over a secured channel.

This specification does *not* describe what the form of users' identities look like, how to delegate them to other services, or how to communicate a security context to the EPS or the ISS.

9. Contributors

Donal K. Fellows
University of Manchester
Oxford Road
Manchester, M13 9PL
U. K.
email: donal.k.fellows@manchester.ac.uk

The author would like to thank the following people (in alphabetical order) for their invaluable assistance in the design and preparation of this specification: Mathias Dalheimer, Andrew Grim-

shaw, Soonwook Hwang, Stephen Pickles, Andreas Savva, Jay Unger, Philipp Wieder, Ramin Yahyapour.

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

13. References

[BES] Foster, I., Grimshaw, A., Lane, P., Lee, W., Morgan, M., Newhouse, S., Pickles, S., Pulsipher, D., Smith, C., Theimer, M., *GFD.108: OGSA® Basic Execution Service Version 1.0*, Open Grid Forum, August 2007.

[Brad] Bradner, S., *RFC 2119: Key Words for Use in RFCs to Indicate Requirement Levels*, IETF, March 1997.

[EMSS] Savva, A., Kishimoto, H., Newhouse, S., Pulsipher, D., *GFD.106: OGSA® EMS Architecture Scenarios, Version 1.0*, Open Grid Forum. May 2007.

[JSDL] Anjomshoaa, A., Brisard, F., Drescher, M., Fellows, D., Ly, A., McGough, S., Pulsipher, D., Savva, A., *GFD.56: Job Submission Description Language, v1.0*, Global Grid Forum. November 2005.

[OWSR] Foster, I., Maguire, T., Snelling, D., *GFD.72: OGSA® WSRF Basic Profile 1.0*, Open Grid Forum, September 2006.

[SOAP] Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J., Nielsen, H., Karmarkar, A., Lafon, Y., *SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)*, W3C, April 2007.

[WSA] Gudgin, M., Hadley, M., Rogers, T., *Web Services Addressing 1.0 – Core*, W3C, May 2006.

[WSAG] Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Kakata, T., Pruyne, J., Rofrano, J., Tuecke, S., Xu, M., *GFD.107: Web Services Agreement Specification (WS-Agreement)*, Open Grid Forum, May 2007.

[WSBF] Liu, L., Meder, S., *Web Services Base Faults 1.2 (WS-BaseFaults)*, OASIS, April 2006.

[WSR] Graham, S., Karmarkar, A., Mischkinsky, J., Robinson, I., Sedukhin, I., *Web Services Resource 1.2 (WS-Resource)*, OASIS, April 2006.

[WSRL] Srinivasan, L., Banks, T., *Web Services Resource Lifetime 1.2 (WS-ResourceLifetime)*, OASIS, April 2006.

[WSRP] Graham, S., Treadwell, J., *Web Services Resource Properties 1.2 (WS-ResourceProperties)*, OASIS, April 2006.

[XQ] Boag, S., Chamberlin, D., Fernández, M., Florescu, D., Robie, J., Siméon, J., *XQuery 1.0: An XML Query Language*, W3C, January 2007.

[XP] Berglund, A., Boag, S., Chamberlin, D., Fernández, M., Kay, M., Robie, J., Siméon, J., *XML Path Language (XPath) 2.0*, W3C, January 2007.

Appendix A: Normative Schemas

A.1 XML Schema for CSG Core

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://schemas.ogf.org/rss/2007/03/csg"
   xmlns:csg="http://schemas.ogf.org/rss/2007/03/csg"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   attributeFormDefault="unqualified" elementFormDefault="qualified"
   xml:lang="en">

   <xsd:annotation>
      <xsd:documentation>
         This describes the basic elements used in all candidate set
         generators.

         Copyright © 2007, Open Grid Forum
      </xsd:documentation>
      <!-- Need to check the copyright and get the full copyright statement in here. -->
   </xsd:annotation>

   <xsd:element name="RequestCandidates">
      <xsd:annotation>
         <xsd:documentation>
            This describes an entire request for candidate generation. The
            input document MUST have a description of the input task (the
            thing that the requestor is proposing to do, and which the CSG
            should generate candidates for how to carry out) and SHOULD
            have a description of what profile is being used, how to order
            the set of candidates generated, and a list of all extension
            elements that the candidate consumer expects to be able to
            deal with. Anything else is completely optional from the point
            of view of this specification, SHOULD be described by the
            UsageProfile, but MUST result in a fault if the particular CSG
            does not understand it.
         </xsd:documentation>
      </xsd:annotation>
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element ref="csg:UsageProfile" maxOccurs="1"
               minOccurs="0" />
            <xsd:element ref="csg:InputDescription" />
            <xsd:element ref="csg:CandidateOrdering" minOccurs="0" />
            <xsd:element ref="csg:PermittedCandidateExtension"
               maxOccurs="unbounded" minOccurs="0" />
            <xsd:any namespace="##other" minOccurs="0"
               processContents="lax" maxOccurs="unbounded" />
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>

   <xsd:element name="UsageProfile" type="xsd:anyURI">
      <xsd:annotation>
         <xsd:documentation>
            This element is used to declare exactly what profile (among
            those supported by the CSG) is being used for this particular
```

```
            request for candidates.

            When used as a property of a CSG, the usage profile declares
            what profiles of that CSG (i.e. what input descriptions they
            accept and candidates they produce) are actually implemented.
            Each particular profile has its own unique "well-known" URI.
            The set of profiles supported should be modelled by the
            service as some kind of property; the manner in which this is
            done is defined by the service rendering. Every CSG must
            support at least one usage profile.
         </xsd:documentation>
      </xsd:annotation>
   </xsd:element>

   <xsd:element name="InputDescription">
      <xsd:annotation>
         <xsd:documentation>
            This element describes what the action the requestor wants to
            carry out, and what they want candidate strategies for
            performing.
         </xsd:documentation>
      </xsd:annotation>
      <xsd:complexType mixed="true">
         <xsd:sequence>
            <xsd:any namespace="##other" processContents="lax"
               minOccurs="1" maxOccurs="1" />
         </xsd:sequence>
         <xsd:anyAttribute namespace="##other" processContents="lax">
            <xsd:annotation>
               <xsd:documentation>
                  Any attribute from namespaces other than the current
                  one MAY be added by the requesting agent, but the CSG
                  MUST fault when faced with an attribute it does not
                  understand.
               </xsd:documentation>
            </xsd:annotation>
         </xsd:anyAttribute>
      </xsd:complexType>
   </xsd:element>

   <xsd:element name="CandidateOrdering">
      <xsd:annotation>
         <xsd:documentation>
            This describes how to order (and possibly how to go about
            generating) the candidates in response to the request
            document. The content of this element is any sequence of XML
            elements (this schema does not define any scheme) but the
            dialect attribute MUST be given so that agents processing this
            element can determine whether they know how to correctly
            process the contents of this element at an early stage.
         </xsd:documentation>
      </xsd:annotation>
      <xsd:complexType mixed="true">
         <xsd:sequence>
            <xsd:any namespace="##other" processContents="lax"
               minOccurs="0" maxOccurs="unbounded" />
```

```
        </xsd:sequence>
        <xsd:attribute name="dialect" type="xsd:anyURI" use="required">
          <xsd:annotation>
            <xsd:documentation>
              This element describes (via a URI) what the contents
              of the CandidateOrdering element are.
            </xsd:documentation>
          </xsd:annotation>
        </xsd:attribute>
        <xsd:anyAttribute namespace="##other" processContents="lax">
          <xsd:annotation>
            <xsd:documentation>
              Attributes from namespaces other than the current one
              MAY also be added. This schema does not constrain
              them.
            </xsd:documentation>
          </xsd:annotation>
        </xsd:anyAttribute>
      </xsd:complexType>
    </xsd:element>

    <xsd:element name="Candidate">
      <xsd:annotation>
        <xsd:documentation>
          This element is one of the ordered set returned in response to
          a request for candidates. It contains three mandatory parts:
          the description of what the candidate is, a best-guess
          estimate of how good a candidate it is, and the (contiguous)
          period of time for which the candidate is valid. The arbitrary
          elements after the CandidateValidity element MUST all be
          understood by any consumer of the Candidate; it MUST ignore
          any Candidate that contains an element as a direct child of
          the Candidate element that it does not understand.
        </xsd:documentation>
      </xsd:annotation>
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element ref="csg:CandidateDescription" />
          <xsd:element ref="csg:CandidateQualityProperties" />
          <xsd:element ref="csg:CandidateValidity" />
          <xsd:any minOccurs="0" maxOccurs="unbounded"
            namespace="##other" processContents="lax" />
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>

    <xsd:element name="CandidateDescription">
      <xsd:annotation>
        <xsd:documentation>
          This states what the candidate is. It should describe how to
          carry out the action described in the InputDescription. It MAY
          contain a copy of part or all of the InputDescription; that
          depends on the profile. Consumers of the Candidate MUST
          understand the contents of this element.
        </xsd:documentation>
      </xsd:annotation>
```

```
    <xsd:complexType>
      <xsd:sequence>
        <xsd:any minOccurs="1" maxOccurs="unbounded"
          namespace="##other" processContents="lax" />
      </xsd:sequence>
      <xsd:anyAttribute namespace="##other" processContents="lax" />
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="CandidateQualityProperties">
    <xsd:annotation>
      <xsd:documentation>
        This describes &amp;quot;how good&amp;quot; a candidate we've got.
        Consumers of a Candidate MAY ignore anything inside this
        element; none of it needs to be understood.
      </xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
      <xsd:sequence>
        <xsd:any maxOccurs="unbounded" minOccurs="0"
          namespace="##other" processContents="lax" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="CandidateValidity">
    <xsd:annotation>
      <xsd:documentation>
        This element describes the contiguous period of time for which
        the containing Candidate is valid. The CSG need not warrant
        anything at all about what happens if a Candidate is actioned
        (in whatever way makes sense according to the profile) outside
        the validity period; the CandidateQualityProperties might be
        totally meaningful then, the Candidate might be impossible to
        action, or anything else in the spectrum of things between
        those two extremes. If the consumer of the Candidate tries to
        action the Candidate within the validity period, it should be
        at least possible (within the bounds of best-effort semantics)
        to action it, and in that case there should be a reasonable
        expectation that the quality described in the
        CandidateQualityProperties be achieved. If non-contiguous
        period of validity is desired (or if the description or
        quality should vary over time) then multiple Candidates should
        be generated. This element MUST be empty.
      </xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
      <xsd:sequence maxOccurs="0" minOccurs="0" />
      <xsd:attribute name="notBefore" type="xsd:dateTime"
        use="required">
        <xsd:annotation>
          <xsd:documentation>
            This describes the first instant of the validity
            period.
          </xsd:documentation>
        </xsd:annotation>
```

```
        </xsd:attribute>
        <xsd:attribute name="notAfter" type="xsd:dateTime"
          use="required">
          <xsd:annotation>
            <xsd:documentation>
              This describes the last instant of the validity
              period.
            </xsd:documentation>
          </xsd:annotation>
        </xsd:attribute>
      </xsd:complexType>
    </xsd:element>

    <xsd:element name="PermittedCandidateExtension">
      <xsd:annotation>
        <xsd:documentation>
          This element is used to describe which extension elements may
          be added to a candidate by the CSG: the CSG SHOULD NOT return
          any candidate that contains an extension element that is not
          described by a PermittedCandidateExtension element in the
          request.
        </xsd:documentation>
      </xsd:annotation>
      <xsd:complexType>
        <xsd:attribute name="type" type="csg:ExtensionType" use="optional"
          default="functional" />
      </xsd:complexType>
    </xsd:element>

    <xsd:simpleType name="ExtensionType">
      <xsd:annotation>
        <xsd:documentation>
          Used to describe the type of an extension element. Functional
          extension elements are direct children of the Candidate
          element and MUST be understood by the consuming system.
          Quality extension elements are children of the
          CandidateQualityProperties element, and MAY be ignored by the
          consuming system.
        </xsd:documentation>
      </xsd:annotation>
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="functional">
          <xsd:annotation>
            <xsd:documentation>
              Describes an element that must be understood if
              present in a Candidate, and which is present as a
              direct child of the Candidate element.
            </xsd:documentation>
          </xsd:annotation>
        </xsd:enumeration>
        <xsd:enumeration value="quality">
          <xsd:annotation>
            <xsd:documentation>
              Describes an element that may be ignored if present in
              a Candidate, and which may only be present in a
              Candidate as a child of the CandidateQualityProperties
```

```
                element.
              </xsd:documentation>
            </xsd:annotation>
          </xsd:enumeration>
        </xsd:restriction>
      </xsd:simpleType>

      <!-- This, along with UsageProfile, are properties common to all Candidate Set Generators -->

      <xsd:element name="SupportedOrderingLanguages" type="xsd:anyURI">
        <xsd:annotation>
          <xsd:documentation>
            The supported ordering languages gives a scheme for allowing
            implementations of a CSG to declare what they accept in the
            CandidateOrdering element. Each particular language has its
            own unique "well-known" URI. The set of languages supported
            should be modelled by the service as some kind of property;
            the manner in which this is done is defined by the service
            rendering. Every CSG must support at least one ordering
            language.
          </xsd:documentation>
        </xsd:annotation>
      </xsd:element>

</xsd:schema>
```

A.2 XML Schema for EPS Core

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://schemas.ogf.org/rss/2007/03/eps/basic"
    xmlns:eps="http://schemas.ogf.org/rss/2007/03/eps/basic"
    xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl"
    xmlns:wsa="http://www.w3.org/2005/03/addressing"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    attributeFormDefault="unqualified" elementFormDefault="qualified"
    xml:lang="en">

    <xsd:annotation>
        <xsd:documentation>
            This describes the elements in candidate set generators that are
            also execution planning service interfaces.

            Copyright © 2007, Open Grid Forum
        </xsd:documentation>
        <!-- Need to check the copyright and get the full copyright statement in here. -->
    </xsd:annotation>

    <xsd:import namespace="http://www.w3.org/2005/03/addressing"
        schemaLocation="http://www.w3.org/2005/03/addressing" />
    <xsd:import namespace="http://schemas.ggf.org/jsdl/2005/11/jsdl"
        schemaLocation="http://schemas.ggf.org/jsdl/2005/11/jsdl" />

    <xsd:element name="BESReference" type="wsa:EndpointReferenceType">
        <xsd:annotation>
            <xsd:documentation>
                This element describes a reference to a basic execution
```

```
            service that the JSDL document (in the same candidate) should
            be submitted to in order to enact the candidate.
         </xsd:documentation>
      </xsd:annotation>
   </xsd:element>
   <xsd:complexType name="EPSCandidateDescriptionType">
      <xsd:annotation>
         <xsd:documentation>
            csg:CandidateDescription elements produced by the EPS MUST
            match this type.
         </xsd:documentation>
      </xsd:annotation>
      <xsd:sequence>
         <xsd:element ref="jsdl:JobDefinition" />
         <xsd:element ref="eps:BESReference" />
      </xsd:sequence>
   </xsd:complexType>
   <xsd:element name="Prices">
      <xsd:annotation>
         <xsd:documentation>
            The Prices is used to describe the amount charged for a job
            execution carried out by a BES. It is describes a sum of
            charges within one or more currencies; note that where a
            charge is made in multiple currencies, it is in general
            non-trivial to convert into a charge in a single currency as
            it depends on many factors outside the reasonable control of
            any planning service. Sources of such problems may include the
            degree of currency uncertainty (as moderated by the clients'
            currency hedges!) and the fact that with some virtual
            currencies the actual rate may be not determined at all until
            substantially after the charge was made (it might depend on
            the overall usage of a particular resource over some period of
            time). As such, this specification states that such
            conversions should not normally be done.
         </xsd:documentation>
      </xsd:annotation>
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element ref="eps:Price" maxOccurs="unbounded"
               minOccurs="1" />
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
   <xsd:element name="Price">
      <xsd:annotation>
         <xsd:documentation>
            The Price element describes a price for resources and services
            calculated using a single currency, which may be a virtual or
            real currency. Real currencies MUST be denoted using the
            pseudo-URI &amp;quot;currency:TLC&amp;quot; where TLC is the ISO 4217
            three-letter code for that particular currency. The Price
            element must contain one or more PriceRange elements, each of
            which describes a range of prices that may be charged; the use
            of multiple ranges allows for finer estimates and
            trace-ability of billing than a single range. The overall
            price within a given currency is given by the sum of the
```

```
            values selected from within each of the contained ranges.
          </xsd:documentation>
        </xsd:annotation>
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element ref="eps:PriceRange" minOccurs="1"
              maxOccurs="unbounded" />
          </xsd:sequence>
          <xsd:attribute name="currency" type="xsd:anyURI" use="required" />
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="PriceRange">
        <xsd:annotation>
          <xsd:documentation>
            The PriceRange element describes an individual contiguous
            range of possible prices. The actual price charged (due to the
            cause described by the element or string content) SHOULD be a
            value in the inclusive range defined by the from and to
            attributes; the range is a singleton if the from value is the
            same as the to value. The from value MUST NOT be greater than
            the to value. A reverse direction charge (where the client of
            the EPS is paid for their custom) MUST be expressed using a
            PriceRange whose from and to attributes are negative.
          </xsd:documentation>
        </xsd:annotation>
        <xsd:complexType mixed="true">
          <xsd:sequence maxOccurs="1" minOccurs="1">
            <xsd:any namespace="##other" processContents="lax"
              minOccurs="0" maxOccurs="unbounded" />
          </xsd:sequence>
          <xsd:attribute name="from" type="xsd:double" use="required" />
          <xsd:attribute name="to" type="xsd:double" use="required" />
        </xsd:complexType>
      </xsd:element>
      <xsd:complexType name="DelaySpanType">
        <xsd:annotation>
          <xsd:documentation>
            This type describes a contiguous range of durations, in
            seconds. Elements of this type MUST NOT have any element or
            string content.
          </xsd:documentation>
        </xsd:annotation>
        <xsd:sequence maxOccurs="0" minOccurs="0" />
        <xsd:attribute name="from" type="xsd:double" />
        <xsd:attribute name="to" type="xsd:double" />
      </xsd:complexType>
      <xsd:complexType name="TimeSpanType">
        <xsd:annotation>
          <xsd:documentation>
            This type describes a contiguous period of time. Elements of
            this type MUST NOT have any element or string content.
          </xsd:documentation>
        </xsd:annotation>
        <xsd:sequence minOccurs="0" maxOccurs="0" />
        <xsd:attribute name="from" type="xsd:dateTime" />
        <xsd:attribute name="to" type="xsd:dateTime" />
```

```
    </xsd:complexType>
    <xsd:element name="StartTimes">
      <xsd:annotation>
        <xsd:documentation>
          This element describes the periods of time when the job may
          start executing. The start times are given as a
          possibly-discontinuous set of times.
        </xsd:documentation>
      </xsd:annotation>
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element ref="eps:StartAt" minOccurs="1"
            maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="StartAt" type="eps:TimeSpanType">
      <xsd:annotation>
        <xsd:documentation>
          This element describes a contiguous period of time during
          which the execution of the job may commence. Both the from and
          the to attribute MUST be present. The from attribute MUST NOT
          specify an instant before the to attribute.
        </xsd:documentation>
      </xsd:annotation>
    </xsd:element>
    <xsd:element name="StartDelays">
      <xsd:annotation>
        <xsd:documentation>
          This element describes the delay between when the job is
          submitted to the BES and when the job may start executing. The
          delay is given as a possibly-discontinuous set of durations.
        </xsd:documentation>
      </xsd:annotation>
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element ref="eps:DelayRange" maxOccurs="unbounded"
            minOccurs="1" />
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="DelayRange" type="eps:DelaySpanType">
      <xsd:annotation>
        <xsd:documentation>
          This element describes a contiguous range of delays, in
          seconds. Two attributes, from and to, MUST be present. The
          value of the from attribute MUST NOT be greater than the value
          of the to attribute.
        </xsd:documentation>
      </xsd:annotation>
    </xsd:element>
    <xsd:element name="EndTimes">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element ref="eps:EndAt" maxOccurs="unbounded"
            minOccurs="1" />
```

```
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="EndAt" type="eps:TimeSpanType">
      <xsd:annotation>
        <xsd:documentation>
          This element describes a contiguous period of time during
          which the execution of the job may finish. Both the from and
          the to attribute MUST be present. The from attribute MUST NOT
          specify an instant before the to attribute.
        </xsd:documentation>
      </xsd:annotation>
    </xsd:element>
    <xsd:element name="ExecutionDuration">
      <xsd:annotation>
        <xsd:documentation>
          This element describes the amount of (wall-clock) time for
          which the job will execute. The run-time is given as a
          possibly-discontinuous set of durations.
        </xsd:documentation>
      </xsd:annotation>
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element ref="eps:RuntimeRange" maxOccurs="unbounded"
            minOccurs="1" />
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="RuntimeRange" type="eps:DelaySpanType">
      <xsd:annotation>
        <xsd:documentation>
          This element describes a contiguous range of runtimes, in
          seconds. Two attributes, from and to, MUST be present. The
          value of the from attribute MUST NOT be greater than the value
          of the to attribute.
        </xsd:documentation>
      </xsd:annotation>
    </xsd:element>
    <xsd:complexType name="EPSCandidateQualityPropertiesType">
      <xsd:annotation>
        <xsd:documentation>
          csg:CandidateQualityProperties elements produced by the EPS
          MUST match or extend this type.
        </xsd:documentation>
      </xsd:annotation>
      <xsd:sequence>
        <xsd:element ref="eps:Prices" maxOccurs="1" minOccurs="0" />
        <xsd:element ref="eps:StartTimes" maxOccurs="1" minOccurs="0" />
        <xsd:element ref="eps:StartDelays" maxOccurs="1" minOccurs="0" />
        <xsd:element ref="eps:EndTimes" maxOccurs="1" minOccurs="0" />
        <xsd:element ref="eps:ExecutionDuration" maxOccurs="1"
          minOccurs="0" />
      </xsd:sequence>
    </xsd:complexType>

</xsd:schema>
```

A.3 XML Schema for Basic-EPS-RF

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://schemas.ogf.org/rss/2007/03/eps/wsrf"
   xmlns:csg="http://schemas.ogf.org/rss/2007/03/csg"
   xmlns:eps-basic="http://schemas.ogf.org/rss/2007/03/eps/basic"
   xmlns:eps-wsrf="http://schemas.ogf.org/rss/2007/03/eps/wsrf"
   xmlns:ogsa-bp="http://schemas.ggf.org/ogsa/2006/05/wsrf-bp"
   xmlns:ws-a="http://www.w3.org/2005/03/addressing"
   xmlns:wsrf-bf="http://docs.oasis-open.org/wsrf/bf-1"
   xmlns:wsrf-rl="http://docs.oasis-open.org/wsrf/rl-1"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   attributeFormDefault="unqualified" elementFormDefault="qualified"
   xml:lang="en">

   <xsd:annotation>
      <xsd:documentation>
         This schema describes the elements that are only used when
         implementing an EPS on top of WS-RF.

         Copyright © 2007, Open Grid Forum
      </xsd:documentation>
      <!-- Need to check the copyright and get the full copyright statement in here. -->
   </xsd:annotation>

   <xsd:import namespace="http://schemas.ogf.org/rss/2007/03/csg"
      schemaLocation="csg.xsd" />
   <xsd:import namespace="http://schemas.ogf.org/rss/2007/03/eps/basic"
      schemaLocation="eps.xsd" />
   <xsd:import namespace="http://docs.oasis-open.org/wsrf/bf-1"
      schemaLocation="http://docs.oasis-open.org/wsrf/bf-1" />
   <xsd:import namespace="http://www.w3.org/2005/03/addressing"
      schemaLocation="http://www.w3.org/2005/03/addressing" />
   <xsd:import namespace="http://docs.oasis-open.org/wsrf/rl-1"
      schemaLocation="http://docs.oasis-open.org/wsrf/rl-1" />
   <xsd:import namespace="http://schemas.ggf.org/ogsa/2006/05/wsrf-bp"
      schemaLocation="http://schemas.ggf.org/ogsa/2006/05/wsrf-bp.xsd" />

   <xsd:element name="GetCandidatesRequest">
      <xsd:annotation>
         <xsd:documentation>
            This element describes the content of the request part of the
            GetCandidates operation of an EPS when implemented using
            WS-RF.
         </xsd:documentation>
      </xsd:annotation>
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element ref="csg:RequestCandidates" />
            <xsd:element ref="eps-wsrf:MaxCandidatesPerTransfer"
               maxOccurs="1" minOccurs="0" />
            <xsd:element ref="eps-wsrf:MaxCandidatesTotal" maxOccurs="1"
               minOccurs="0" />
            <xsd:element ref="eps-wsrf:MaxResponseOctets" maxOccurs="1"
               minOccurs="0">
            </xsd:element>
```

```
            <xsd:element ref="eps-wsrf:IterableSetLifetime" maxOccurs="1"
               minOccurs="0" />
            <xsd:any namespace="##other" processContents="lax"
               minOccurs="0" maxOccurs="unbounded" />
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>

   <xsd:element name="GetCandidatesResponse">
      <xsd:annotation>
         <xsd:documentation>
            The response of the GetCandidates operation is an optional
            sequence of candidates (satisfying the general cadinality
            rules of the service and the requestor) followed by an
            optional reference to a resource which will provide the
            remaining candidates. If there is no reference to an Iterable
            Set Service, the sequence of candidates in this response
            message must be all the candidates that the EPS is prepared to
            give the client. Note that if a reference to an ISS is
            present, no conclusions about the total number of remaining
            candidates may be drawn.
         </xsd:documentation>
      </xsd:annotation>
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element ref="csg:Candidate" maxOccurs="unbounded"
               minOccurs="0" />
            <xsd:element ref="eps-wsrf:IterableSetReference" maxOccurs="1"
               minOccurs="0" />
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>

   <xsd:element name="UnsupportedCSGProfileFault"
      type="wsrf-bf:BaseFaultType">
      <xsd:annotation>
         <xsd:documentation>
            This fault is thrown when the request to the GetCandidates
            operation does not match any profile that is supported by the
            EPS.
         </xsd:documentation>
      </xsd:annotation>
   </xsd:element>

   <xsd:element name="UnsupportedOrderingLanguageFault"
      type="wsrf-bf:BaseFaultType">
      <xsd:annotation>
         <xsd:documentation>
            This fault is thrown when the ordering component of the
            request to the GetCandidates operation is not in a language
            supported by the EPS.
         </xsd:documentation>
      </xsd:annotation>
   </xsd:element>

   <xsd:element name="UnableToGenerateCandidatesFault"
```

```
1902        type="wsrf-bf:BaseFaultType">
1903        <xsd:annotation>
1904          <xsd:documentation>
1905            This fault is thrown when the EPS is unable to generate
1906            candidates for some reason that is described within the fault.
1907          </xsd:documentation>
1908        </xsd:annotation>
1909      </xsd:element>
1910
1911      <xsd:element name="BadCandidateCardinalityRulesFault"
1912        type="wsrf-bf:BaseFaultType">
1913        <xsd:annotation>
1914          <xsd:documentation>
1915            This fault is thrown when the requested candidate cardinality
1916            restrictions are either inconsistent or impossible given the
1917            EPS's general limits.
1918          </xsd:documentation>
1919        </xsd:annotation>
1920      </xsd:element>
1921
1922      <xsd:element name="MaxCandidatesPerTransfer" type="xsd:positiveInteger">
1923        <xsd:annotation>
1924          <xsd:documentation>
1925            This element is used to specify what the client's maximum
1926            number of candidates per message is. Note that if the server's
1927            limits are lower, this element will have no effect.
1928          </xsd:documentation>
1929        </xsd:annotation>
1930      </xsd:element>
1931
1932      <xsd:element name="MaxCandidatesTotal" type="xsd:positiveInteger">
1933        <xsd:annotation>
1934          <xsd:documentation>
1935            This element is used to specify the maximum total number of
1936            candidates that the client is interested in.
1937          </xsd:documentation>
1938        </xsd:annotation>
1939      </xsd:element>
1940
1941      <xsd:element name="MaxResponseOctets" type="xsd:positiveInteger">
1942        <xsd:annotation>
1943          <xsd:documentation>
1944            This element is used to specify the maximum number of bytes in
1945            a response that contains more than one Candidate.
1946          </xsd:documentation>
1947        </xsd:annotation>
1948      </xsd:element>
1949
1950      <xsd:element name="IterableSetLifetime" type="xsd:double">
1951        <xsd:annotation>
1952          <xsd:documentation>
1953            This element is used to specify the initial lifetime of the
1954            ISS resource, in seconds. If unspecified, the EPS should use
1955            its own default value. While clients may extend the lifetime
1956            of the ISS, they cannot request an unbounded lifetime at
1957            creation time and SHOULD NOT request an unbounded lifetime
```

```
            using the interfaces given by WS-Lifetime.
         </xsd:documentation>
      </xsd:annotation>
   </xsd:element>

   <xsd:element name="IterableSetReference"
      type="ws-a:EndpointReferenceType">
      <xsd:annotation>
         <xsd:documentation>
            This element contains a reference to an Iterable Set Service
            resource containing additional candidates generated in
            response to a GetCandidates request.
         </xsd:documentation>
      </xsd:annotation>
   </xsd:element>

   <xsd:element name="GetMoreCandidatesRequest">
      <xsd:annotation>
         <xsd:documentation>
            This is the content of the request to the GetMoreCandidates
            operation of the Iterable Set Service. It contains an optional
            Count element, but is otherwise empty.
         </xsd:documentation>
      </xsd:annotation>
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element ref="eps-wsrf:Count" maxOccurs="1" minOccurs="0" />
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>

   <xsd:element name="GetMoreCandidatesResponse">
      <xsd:annotation>
         <xsd:documentation>
            This is the content of the response to the GetMoreCandidates
            operation of the IterableSetService. It contains zero or more
            candidates; the response SHOULD only contain zero candidates
            when the list of candidates (internal to the ISS) is
            exhausted.
         </xsd:documentation>
      </xsd:annotation>
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element ref="csg:Candidate" maxOccurs="unbounded"
               minOccurs="0" />
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>

   <xsd:element name="Count" type="xsd:positiveInteger">
      <xsd:annotation>
         <xsd:documentation>
            This element describes how many candidates the client wants to
            retrieve from the ISS this time.
         </xsd:documentation>
      </xsd:annotation>
```

```
</xsd:element>

<xsd:element name="EPSProperties">
  <xsd:annotation>
    <xsd:documentation>
      This describes the resource properties document for the
      Execution Planning Service.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="ogsa-bp:ResourcePropertyNames" />
      <xsd:element ref="ogsa-bp:FinalWSResourceInterface" />
      <xsd:element ref="ogsa-bp:WSResourceInterfaces" />
      <xsd:element ref="ogsa-bp:ResourceEndpointReference" />
      <xsd:element ref="csg:UsageProfile" maxOccurs="unbounded"
        minOccurs="1" />
      <xsd:element ref="csg:SupportedOrderingLanguages"
        maxOccurs="unbounded" minOccurs="1" />
      <xsd:element ref="eps-wsrf:MaxCandidatesPerTransfer"
        minOccurs="1" maxOccurs="1" />
      <xsd:element ref="eps-wsrf:MaxResponseOctets"></xsd:element>
      <xsd:element ref="eps-wsrf:DefaultIterableSetLifetime" />
      <xsd:any namespace="##other" minOccurs="0"
        maxOccurs="unbounded" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<xsd:element name="ISSProperties">
  <xsd:annotation>
    <xsd:documentation>
      This describes the resource properties document for the
      Iterable Set Service.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="ogsa-bp:ResourcePropertyNames" />
      <xsd:element ref="ogsa-bp:FinalWSResourceInterface" />
      <xsd:element ref="ogsa-bp:WSResourceInterfaces" />
      <xsd:element ref="ogsa-bp:ResourceEndpointReference" />
      <xsd:element ref="wsrf-rl:CurrentTime" maxOccurs="1"
        minOccurs="1" />
      <xsd:element ref="wsrf-rl:TerminationTime" maxOccurs="1"
        minOccurs="1" />
      <xsd:element ref="eps-wsrf:MaxCandidatesPerTransfer" />
      <xsd:element ref="eps-wsrf:MaxResponseOctets" maxOccurs="1"
        minOccurs="0">
      </xsd:element>
      <xsd:element ref="eps-wsrf:CandidatesTransferredSoFar"
        maxOccurs="1" minOccurs="0" />
      <xsd:element ref="eps-wsrf:AnyMoreCandidatesToTransfer"
        maxOccurs="1" minOccurs="0" />
    </xsd:sequence>
  </xsd:complexType>
```

```
    </xsd:element>

    <xsd:element name="DefaultIterableSetLifetime" type="xsd:duration">
      <xsd:annotation>
        <xsd:documentation>
          This describes the default lifetime of Iterable Set Service
          resources created by the Execution Planning Service.
        </xsd:documentation>
      </xsd:annotation>
    </xsd:element>

    <xsd:element name="CandidatesTransferredSoFar"
      type="xsd:nonNegativeInteger">
      <xsd:annotation>
        <xsd:documentation>
          This describes the number of candidates that have already been
          transferred from the (conceptual) underlying ordered set,
          including any returned in the result of the initial call to
          GetCandidates.
        </xsd:documentation>
      </xsd:annotation>
    </xsd:element>

    <xsd:element name="AnyMoreCandidatesToTransfer" type="xsd:boolean">
      <xsd:annotation>
        <xsd:documentation>
          This describes whether there are any more candidates in the
          underlying (conceptual) ordered set. Support for it is not
          required, but if it is present then no candidates should ever
          be returned after the value of this property becomes true.
          Implementations may choose to release internal resources at
          the same time as setting this property to true.
        </xsd:documentation>
      </xsd:annotation>
    </xsd:element>

</xsd:schema>
```

A.4 WSDL for Basic-EPS-RF

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<wsdl:definitions name="eps-wsrf" xml:lang="en"
    targetNamespace="http://schemas.ogf.org/rss/2007/03/eps/wsrf"
    xmlns:eps-wsrf="http://schemas.ogf.org/rss/2007/03/eps/wsrf"
    xmlns:rw="http://docs.oasis-open.org/wsrf/rw-1"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:wsrf-rp="http://docs.oasis-open.org/wsrf/rp-1"
    xmlns:wsrf-rlw="http://docs.oasis-open.org/wsrf/rlw-1"
    xmlns:wsrf-rpw="http://docs.oasis-open.org/wsrf/rpw-1"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">

    <wsdl:documentation>
      This describes the details of the rendering of the Execution Planning
      Service and Iterable Set Service using WS-ResourceFramework over SOAP.
```

```
2124
2125        Copyright © 2006-2007, Open Grid Forum
2126    </wsdl:documentation>
2127    <!-- Need to check the copyright and get the full copyright statement in here. -->
2128
2129    <wsdl:types>
2130      <xsd:schema>
2131        <xsd:import namespace="http://docs.oasis-open.org/wsrf/rp-1"
2132          schemaLocation="http://docs.oasis-open.org/wsrf/rp-1" />
2133        <xsd:import
2134          namespace="http://schemas.ogf.org/rss/2006/12/eps/wsrf"
2135          schemaLocation="eps-wsrf.xsd" />
2136      </xsd:schema>
2137    </wsdl:types>
2138    <wsdl:import namespace="http://docs.oasis-open.org/wsrf/rw-1"
2139      location="http://docs.oasis-open.org/wsrf/rw-1" />
2140    <wsdl:import namespace="http://docs.oasis-open.org/wsrf/rlw-1"
2141      location="http://docs.oasis-open.org/wsrf/rlw-1" />
2142    <wsdl:import namespace="http://docs.oasis-open.org/wsrf/rpw-1"
2143      location="http://docs.oasis-open.org/wsrf/rpw-1" />
2144
2145    <wsdl:message name="GetCandidates">
2146      <wsdl:documentation>
2147        This message requests that the Execution Planning Service generate
2148        an set of plans for executing a job. The set of plans MUST be
2149        ordered according to the scoring function described within the
2150        message or, if that is absent, the service's default.
2151      </wsdl:documentation>
2152      <wsdl:part element="eps-wsrf:GetCandidatesRequest"
2153        name="GetCandidatesRequest" />
2154    </wsdl:message>
2155    <wsdl:message name="GetCandidatesResult">
2156      <wsdl:documentation>
2157        This message contains either the ordered set of plans (if that is
2158        a 'small' set) or a reference to a service that manages a much
2159        larger set of plans such that you can retrieve the results a bit
2160        at a time.
2161      </wsdl:documentation>
2162      <wsdl:part element="eps-wsrf:GetCandidatesResponse"
2163        name="GetCandidatesResponse" />
2164    </wsdl:message>
2165    <wsdl:message name="GetMoreCandidates">
2166      <wsdl:documentation>
2167        This is message sent to retrieve part of a set of plans when
2168        operating in partial-transfer mode.
2169      </wsdl:documentation>
2170      <wsdl:part name="GetMoreCandidatesRequest"
2171        element="eps-wsrf:GetMoreCandidatesRequest" />
2172    </wsdl:message>
2173    <wsdl:message name="GetMoreCandidatesResult">
2174      <wsdl:documentation>
2175        This message holds plans sent when doing a partial transfer.
2176      </wsdl:documentation>
2177      <wsdl:part name="GetMoreCandidatesResponse"
2178        element="eps-wsrf:GetMoreCandidatesResponse" />
2179    </wsdl:message>
```

```
<wsdl:message name="UnsupportedOrderingLanguageFault">
  <wsdl:documentation>
    Fault thrown if the EPS cannot handle the submitted candidate
    ordering language.
  </wsdl:documentation>
  <wsdl:part name="UnsupportedOrderingLanguageFault"
    element="eps-wsrf:UnsupportedOrderingLanguageFault" />
</wsdl:message>
<wsdl:message name="BadCandidateCardinalityRulesFault">
  <wsdl:documentation>
    Fault thrown if an attempt to violate the rules on plan
    cardinality is made.
  </wsdl:documentation>
  <wsdl:part name="BadCandidateCardinalityRulesFault"
    element="eps-wsrf:BadCandidateCardinalityRulesFault" />
</wsdl:message>
<wsdl:message name="UnableToGenerateCandidatesFault">
  <wsdl:documentation>
    Fault thrown if it is impossible to generate plans for some other
    reason not covered by the other faults.
  </wsdl:documentation>
  <wsdl:part name="UnableToGenerateCandidatesFault"
    element="eps-wsrf:UnableToGenerateCandidatesFault" />
</wsdl:message>
<wsdl:message name="UnsupportedCSGProfileFault">
  <wsdl:documentation>
    Fault thrown if the request to the EPS isn't one that is
    understood properly.
  </wsdl:documentation>
  <wsdl:part name="UnsupportedCSGProfileFault"
    element="eps-wsrf:UnsupportedCSGProfileFault" />
</wsdl:message>

<wsdl:portType name="ExecutionPlanningService"
  wsrf-rp:ResourceProperties="eps-wsrf:EPSProperties">
  <wsdl:documentation>
    The execution planner port-type describes the real core of the EPS
    specification, as rendered using WSRF.
  </wsdl:documentation>
  <wsdl:operation name="GetCandidates">
    <wsdl:input message="eps-wsrf:GetCandidates"
      wsa:Action="http://schemas.ogf.org/rss/2007/03/eps/wsrf/GetCandidates" />
    <wsdl:output message="eps-wsrf:GetCandidatesResult"
      wsa:Action="http://schemas.ogf.org/rss/2007/03/eps/wsrf/GetCandidatesResponse" />
    <wsdl:fault name="UnsupportedOrderingLanguageFault"
      message="eps-wsrf:UnsupportedOrderingLanguageFault" />
    <wsdl:fault name="BadCandidateCardinalityRulesFault"
      message="eps-wsrf:BadCandidateCardinalityRulesFault" />
    <wsdl:fault name="UnableToGenerateCandidatesFault"
      message="eps-wsrf:UnableToGenerateCandidatesFault" />
    <wsdl:fault name="ResourceUnknownFault"
      message="rw:ResourceUnknownFault" />
    <wsdl:fault name="UnsupportedCSGProfileFault"
      message="eps-wsrf:UnsupportedCSGProfileFault" />
  </wsdl:operation>
  <wsdl:operation name="GetResourcePropertyDocument">
```

```
        <wsdl:input name="GetResourcePropertyDocumentRequest"
            message="wsrf-rpw:GetResourcePropertyDocumentRequest" />
        <wsdl:output name="GetResourcePropertyDocumentResponse"
            message="wsrf-rpw:GetResourcePropertyDocumentResponse" />
        <wsdl:fault name="ResourceUnknownFault"
            message="rw:ResourceUnknownFault" />
      </wsdl:operation>
    </wsdl:portType>
    <wsdl:portType name="IterableSetService"
      wsrf-rp:ResourceProperties="eps-wsrf:ISSProperties">
      <wsdl:documentation>
        The execution planner result port-type is used to implement
        partial transfers of results. Note that in the WSRF rendering, the
        ISS must implement the WS-ResourceLifetime operations too.
      </wsdl:documentation>
      <wsdl:operation name="GetMoreCandidates">
        <wsdl:input message="eps-wsrf:GetMoreCandidates"
            wsa:Action="http://schemas.ogf.org/rss/2007/03/eps/wsrf/GetMoreCandidates" />
        <wsdl:output message="eps-wsrf:GetMoreCandidatesResult"
       wsa:Action="http://schemas.ogf.org/rss/2007/03/eps/wsrf/GetMoreCandidatesResponse" />
        <wsdl:fault name="BadCandidateCardinalityRulesFault"
            message="eps-wsrf:BadCandidateCardinalityRulesFault" />
        <wsdl:fault name="UnableToGenerateCandidatesFault"
            message="eps-wsrf:UnableToGenerateCandidatesFault" />
        <wsdl:fault name="ResourceUnknownFault"
            message="rw:ResourceUnknownFault" />
      </wsdl:operation>
      <wsdl:operation name="Destroy">
        <wsdl:input name="DestroyRequest"
            message="wsrf-rlw:DestroyRequest" />
        <wsdl:output name="DestroyResponse"
            message="wsrf-rlw:DestroyResponse" />
        <wsdl:fault name="ResourceNotDestroyedFault"
            message="wsrf-rlw:ResourceNotDestroyedFault" />
        <wsdl:fault name="ResourceUnknownFault"
            message="rw:ResourceUnknownFault" />
      </wsdl:operation>
      <wsdl:operation name="SetTerminationTime">
        <wsdl:input name="SetTerminationTimeRequest"
            message="wsrf-rlw:SetTerminationTimeRequest" />
        <wsdl:output name="SetTerminationTimeResponse"
            message="wsrf-rlw:SetTerminationTimeResponse" />
        <wsdl:fault name="UnableToSetTerminationTimeFault"
            message="wsrf-rlw:UnableToSetTerminationTimeFault" />
        <wsdl:fault name="ResourceUnknownFault"
            message="rw:ResourceUnknownFault" />
        <wsdl:fault name="TerminationTimeChangeRejectedFault"
            message="wsrf-rlw:TerminationTimeChangeRejectedFault" />
      </wsdl:operation>
      <wsdl:operation name="GetResourcePropertyDocument">
        <wsdl:input name="GetResourcePropertyDocumentRequest"
            message="wsrf-rpw:GetResourcePropertyDocumentRequest" />
        <wsdl:output name="GetResourcePropertyDocumentResponse"
            message="wsrf-rpw:GetResourcePropertyDocumentResponse" />
        <wsdl:fault name="ResourceUnknownFault"
            message="rw:ResourceUnknownFault" />
```

```
      </wsdl:operation>
   </wsdl:portType>

   <wsdl:binding name="eps-wsrf-SOAP"
      type="eps-wsrf:ExecutionPlanningService">
      <soap:binding style="document"
         transport="http://schemas.xmlsoap.org/soap/http" />
      <wsdl:operation name="GetCandidates">
         <soap:operation
            soapAction="http://schemas.ogf.org/rss/2007/03/eps/wsrf/GetExecutionPlans" />
         <wsdl:input>
            <soap:body use="literal" />
         </wsdl:input>
         <wsdl:output>
            <soap:body use="literal" />
         </wsdl:output>
         <wsdl:fault name="UnsupportedOrderingLanguageFault">
            <soap:fault
               namespace="http://schemas.ogf.org/rss/2007/03/eps/wsrf"
               use="literal" name="UnsupportedOrderingLanguageFault" />
         </wsdl:fault>
         <wsdl:fault name="BadPlanCardinalityRulesFault">
            <soap:fault
               namespace="http://schemas.ogf.org/rss/2007/03/eps/wsrf"
               use="literal" name="BadPlanCardinalityRulesFault" />
         </wsdl:fault>
         <wsdl:fault name="UnableToGeneratePlansFault">
            <soap:fault
               namespace="http://schemas.ogf.org/rss/2007/03/eps/wsrf"
               use="literal" name="UnableToGeneratePlansFault" />
         </wsdl:fault>
         <wsdl:fault name="ResourceUnknownFault">
            <soap:fault
               namespace="http://schemas.ogf.org/rss/2007/03/eps/wsrf"
               use="literal" name="ResourceUnknownFault" />
         </wsdl:fault>
      </wsdl:operation>
      <wsdl:operation name="GetResourcePropertyDocument">
         <soap:operation
       soapAction="http://schemas.ogf.org/rss/2007/03/eps/wsrf/GetResourcePropertyDocument" />
         <wsdl:input name="GetResourcePropertyDocumentRequest">
            <soap:body use="literal" />
         </wsdl:input>
         <wsdl:output name="GetResourcePropertyDocumentResponse">
            <soap:body use="literal" />
         </wsdl:output>
         <wsdl:fault name="ResourceUnknownFault">
            <soap:fault
               namespace="http://schemas.ogf.org/rss/2007/03/eps/wsrf"
               use="literal" name="ResourceUnknownFault" />
         </wsdl:fault>
      </wsdl:operation>
   </wsdl:binding>
   <wsdl:binding name="iss-wsrf-SOAP" type="eps-wsrf:IterableSetService">
      <soap:binding style="document"
         transport="http://schemas.xmlsoap.org/soap/http" />
```

```
2348        <wsdl:operation name="GetMoreCandidates">
2349            <soap:operation
2350                soapAction="http://schemas.ogf.org/rss/2007/03/eps/wsrf/GetMorePlans" />
2351            <wsdl:input>
2352                <soap:body use="literal" />
2353            </wsdl:input>
2354            <wsdl:output>
2355                <soap:body use="literal" />
2356            </wsdl:output>
2357            <wsdl:fault name="BadPlanCardinalityRulesFault">
2358                <soap:fault
2359                    namespace="http://schemas.ogf.org/rss/2007/03/eps/wsrf"
2360                    use="literal" name="BadPlanCardinalityRulesFault" />
2361            </wsdl:fault>
2362            <wsdl:fault name="UnableToGeneratePlansFault">
2363                <soap:fault
2364                    namespace="http://schemas.ogf.org/rss/2007/03/eps/wsrf"
2365                    use="literal" name="UnableToGeneratePlansFault" />
2366            </wsdl:fault>
2367            <wsdl:fault name="ResourceUnknownFault">
2368                <soap:fault
2369                    namespace="http://schemas.ogf.org/rss/2007/03/eps/wsrf"
2370                    use="literal" name="ResourceUnknownFault" />
2371            </wsdl:fault>
2372        </wsdl:operation>
2373        <wsdl:operation name="Destroy">
2374            <soap:operation
2375                soapAction="http://schemas.ogf.org/rss/2007/03/eps/wsrf/Destroy" />
2376            <wsdl:input name="DestroyRequest">
2377                <soap:body use="literal" />
2378            </wsdl:input>
2379            <wsdl:output name="DestroyResponse">
2380                <soap:body use="literal" />
2381            </wsdl:output>
2382            <wsdl:fault name="ResourceNotDestroyedFault">
2383                <soap:fault
2384                    namespace="http://schemas.ogf.org/rss/2007/03/eps/wsrf"
2385                    use="literal" name="ResourceNotDestroyedFault" />
2386            </wsdl:fault>
2387            <wsdl:fault name="ResourceUnknownFault">
2388                <soap:fault
2389                    namespace="http://schemas.ogf.org/rss/2007/03/eps/wsrf"
2390                    use="literal" name="ResourceUnknownFault" />
2391            </wsdl:fault>
2392        </wsdl:operation>
2393        <wsdl:operation name="SetTerminationTime">
2394            <soap:operation
2395                soapAction="http://schemas.ogf.org/rss/2007/03/eps/wsrf/SetTerminationTime" />
2396            <wsdl:input name="SetTerminationTimeRequest">
2397                <soap:body use="literal" />
2398            </wsdl:input>
2399            <wsdl:output name="SetTerminationTimeResponse">
2400                <soap:body use="literal" />
2401            </wsdl:output>
2402            <wsdl:fault name="UnableToSetTerminationTimeFault">
2403                <soap:fault
```

```
                namespace="http://schemas.ogf.org/rss/2007/03/eps/wsrf"
                use="literal" name="UnableToSetTerminationTimeFault" />
          </wsdl:fault>
          <wsdl:fault name="ResourceUnknownFault">
             <soap:fault
                namespace="http://schemas.ogf.org/rss/2007/03/eps/wsrf"
                use="literal" name="ResourceUnknownFault" />
          </wsdl:fault>
          <wsdl:fault name="TerminationTimeChangeRejectedFault">
             <soap:fault
                namespace="http://schemas.ogf.org/rss/2007/03/eps/wsrf"
                use="literal" name="TerminationTimeChangeRejectedFault" />
          </wsdl:fault>
       </wsdl:operation>
       <wsdl:operation name="GetResourcePropertyDocument">
          <soap:operation
     soapAction="http://schemas.ogf.org/rss/2007/03/eps/wsrf/GetResourcePropertyDocument" />
          <wsdl:input name="GetResourcePropertyDocumentRequest">
             <soap:body use="literal" />
          </wsdl:input>
          <wsdl:output name="GetResourcePropertyDocumentResponse">
             <soap:body use="literal" />
          </wsdl:output>
          <wsdl:fault name="ResourceUnknownFault">
             <soap:fault
                namespace="http://schemas.ogf.org/rss/2007/03/eps/wsrf"
                use="literal" name="ResourceUnknownFault" />
          </wsdl:fault>
       </wsdl:operation>
    </wsdl:binding>

    <wsdl:service name="eps-wsrf-SOAP">
       <wsdl:documentation>
          The same service should normally implement both the EPS and the
          ISS port-types. The locations specified here are arbitrary though;
          callers should know who they are talking to independent of this
          specification, and the reference to the ISS port type is
          transferred to the client as an endpoint reference.
       </wsdl:documentation>
       <wsdl:port binding="eps-wsrf:eps-wsrf-SOAP" name="eps-wsrf-SOAP">
          <soap:address location="http://localhost/" />
       </wsdl:port>
       <wsdl:port name="iss-wsrf-SOAP" binding="eps-wsrf:iss-wsrf-SOAP">
          <soap:address location="http://localhost/" />
       </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```

A.5 XML Schema for Basic-EPS-XF

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<xsd:schema targetNamespace="http://schemas.ogf.org/rss/2007/10/eps/transfer"
    xmlns:csg="http://schemas.ogf.org/rss/2007/03/csg"
    xmlns:wsen="http://schemas.xmlsoap.org/ws/2004/09/enumeration"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    elementFormDefault="qualified" xml:lang="en">
```

```
   <xsd:annotation>
      <xsd:documentation>
         This schema describes the elements that are only used when
         implementing an EPS on top of WS-Transfer and WS-Enumeration.

         Copyright © 2007, Open Grid Forum
      </xsd:documentation>
      <!-- Need to check the copyright and get the full copyright statement in here. -->
   </xsd:annotation>

   <xsd:import namespace="http://schemas.ogf.org/rss/2007/03/csg"
      schemaLocation="csg.xsd" />
   <xsd:import namespace="http://schemas.xmlsoap.org/ws/2004/09/enumeration"
      schemaLocation="wsen-hacks.xsd" />

   <xsd:element name="EnumerateCandidatesRequest">
      <xsd:annotation>
         <xsd:documentation>
            The format of a request to enumerate candidates.
         </xsd:documentation>
      </xsd:annotation>
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element ref="csg:RequestCandidates" />
            <xsd:element ref="wsen:EndTo" minOccurs="0" />
            <xsd:element ref="wsen:Expires" minOccurs="0" />
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>

   <xsd:complexType name="EPSRepresentation">
      <xsd:annotation>
         <xsd:documentation>
            The minimal required representation for an EPS when using
            WS-Transfer is to describe the usage profiles and ordering
            languages that are supported.
         </xsd:documentation>
      </xsd:annotation>
      <xsd:sequence>
         <xsd:element ref="csg:UsageProfile" minOccurs="1"
            maxOccurs="unbounded" />
         <xsd:element ref="csg:SupportedOrderingLanguages" minOccurs="1"
            maxOccurs="unbounded" />
      </xsd:sequence>
   </xsd:complexType>
</xsd:schema>
```

A.6 WSDL for Basic-EPS-XF

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<wsdl:definitions name="eps-xf" xml:lang="en"
   xmlns:eps-xf="http://schemas.ogf.org/rss/2007/10/eps/transfer"
   xmlns:csg="http://schemas.ogf.org/rss/2007/03/csg"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
```

```
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsen="http://schemas.xmlsoap.org/ws/2004/09/enumeration"
xmlns:wsxf="http://schemas.xmlsoap.org/ws/2004/09/transfer"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://schemas.ogf.org/rss/2007/10/eps/transfer">
<wsdl:documentation>
   Defines the required parts of an Execution Planning Service as
   implemented using WS-Transfer and WS-Enumeration.

   Copyright © 2007, Open Grid Forum
</wsdl:documentation>

<wsdl:types>
   <xsd:schema>
      <xsd:import
         namespace="http://schemas.ogf.org/rss/2007/10/eps/transfer"
         schemaLocation="eps-transfer.xsd" />
      <xsd:import namespace="http://schemas.ogf.org/rss/2007/03/csg"
         schemaLocation="csg.xsd" />
      <xsd:import
      schemaLocation="http://schemas.xmlsoap.org/ws/2004/09/enumeration/enumeration.xsd"
         namespace="http://schemas.xmlsoap.org/ws/2004/09/enumeration" />
      <xsd:import
         namespace="http://schemas.xmlsoap.org/ws/2004/08/addressing"
         schemaLocation="http://schemas.xmlsoap.org/ws/2004/08/addressing" />
   </xsd:schema>
</wsdl:types>
<wsdl:import
   location="http://schemas.xmlsoap.org/ws/2004/09/transfer/transfer.wsdl"
   namespace="http://schemas.xmlsoap.org/ws/2004/09/transfer" />

<wsdl:message name="EnumerateCandidatesRequest">
   <wsdl:part element="eps-xf:EnumerateCandidatesRequest" name="Body" />
</wsdl:message>
<wsdl:message name="EnumerateCandidatesResponse">
   <wsdl:part element="wsen:EnumerateResponse" name="Body" />
</wsdl:message>
<wsdl:message name="EPSRepresentation">
   <wsdl:part name="Body" type="eps-xf:EPSRepresentation" />
</wsdl:message>
<wsdl:portType name="eps-transfer">
   <wsdl:operation name="EnumerateCandidates">
      <wsdl:input message="eps-xf:EnumerateCandidatesRequest"
         wsa:Action="http://schemas.ogf.org/rss/2007/10/eps/transfer/EnumerateCandidates" />
      <wsdl:output message="eps-xf:EnumerateCandidatesResponse"
     wsa:Action="http://schemas.xmlsoap.org/ws/2004/09/enumeration/EnumerateResponse" />
   </wsdl:operation>
   <wsdl:operation name="Get">
      <wsdl:input message="wsxf:EmptyMessage"
         wsa:Action="http://schemas.xmlsoap.org/ws/2004/09/transfer/Get" />
      <wsdl:output message="eps-xf:EPSRepresentation"
         wsa:Action="http://schemas.xmlsoap.org/ws/2004/09/transfer/GetResponse" />
   </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="eps-transferSOAP" type="eps-xf:eps-transfer">
   <soap:binding style="document"
```

```
        transport="http://schemas.xmlsoap.org/soap/http" />
      <wsdl:operation name="EnumerateCandidates">
        <soap:operation
          soapAction="http://schemas.ogf.org/rss/2007/10/eps/transfer/EnumerateCandidates" />
        <wsdl:input>
          <soap:body use="literal" />
        </wsdl:input>
        <wsdl:output>
          <soap:body use="literal" />
        </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="Get">
        <soap:operation
          soapAction="http://schemas.xmlsoap.org/ws/2004/09/transfer/Get" />
        <wsdl:input>
          <soap:body use="literal" />
        </wsdl:input>
        <wsdl:output>
          <soap:body use="literal" />
        </wsdl:output>
      </wsdl:operation>
    </wsdl:binding>
    <wsdl:service name="eps-transfer">
      <wsdl:port binding="eps-xf:eps-transferSOAP"
        name="eps-transferSOAP">
        <soap:address location="http://localhost/" />
      </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```

A.7 Supplemental XML Schema for WS-Enumeration

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
   targetNamespace="http://schemas.xmlsoap.org/ws/2004/09/enumeration"
   xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
   xmlns:wsen="http://schemas.xmlsoap.org/ws/2004/09/enumeration"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   elementFormDefault="qualified" xml:lang="en">
   <xsd:annotation>
      <xsd:documentation>
         This just adds the EndTo and Expires elements as top-level
         elements, as they were missing from the original WS-Enumeration
         specification.

         Copyright © 2007, Open Grid Forum
      </xsd:documentation>
   </xsd:annotation>
   <xsd:include
      schemaLocation="http://schemas.xmlsoap.org/ws/2004/09/enumeration/enumeration.xsd">
   </xsd:include>
   <xsd:import namespace="http://schemas.xmlsoap.org/ws/2004/08/addressing"
      schemaLocation="http://schemas.xmlsoap.org/ws/2004/08/addressing/addressing.xsd" />
   <xsd:element name="EndTo" type="wsa:EndpointReferenceType" />
   <xsd:element name="Expires" type="wsen:ExpirationType" />
</xsd:schema>
```

A.8 XML Schema for SCOL

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://schemas.ogf.org/rss/2007/05/scol"
   xmlns:select="http://schemas.ogf.org/rss/2007/05/scol"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   elementFormDefault="qualified" attributeFormDefault="unqualified"
   xml:lang="en">

   <xsd:annotation>
      <xsd:documentation>
         This describes the types and elements used to describe the RSS
         Simple Candidate Ordering Language.

         Copyright © 2006-2007, Open Grid Forum
      </xsd:documentation>
   </xsd:annotation>

   <!-- Basic types -->
   <xsd:simpleType name="XPathType">
      <xsd:annotation>
         <xsd:documentation>
            The string values of this type MUST be XPath 2.0 expressions,
            but the W3C does not seem to have defined an XSD type for
            this. It is up to the implementing application to enforce this
            restriction.
         </xsd:documentation>
      </xsd:annotation>
      <xsd:restriction base="xsd:string" />
   </xsd:simpleType>
   <xsd:complexType name="empty">
      <xsd:annotation>
         <xsd:documentation>
            The type of elements with no content.
         </xsd:documentation>
      </xsd:annotation>
      <xsd:sequence minOccurs="0" maxOccurs="0" />
   </xsd:complexType>
   <xsd:simpleType name="SelectOperation">
      <xsd:annotation>
         <xsd:documentation>
            Enumeration of ways to convert a NodeSet to a single double.
         </xsd:documentation>
      </xsd:annotation>
      <xsd:restriction base="xsd:string">
         <xsd:enumeration value="first">
            <xsd:annotation>
               <xsd:documentation>
                  Value is the value of the first node in the NodeSet
                  resulting from the XPath selection, interpreted as a
                  double. If the node is an Element, then concatenate
                  all the children of the Element that are Text nodes,
                  and parse the resulting string. The evaluation of the
                  containing select:Select term MUST fail if the NodeSet
                  is empty or the parsed string cannot be interpreted as
                  a valid double value in the range (-Inf..+Inf).
```

```
2677              </xsd:documentation>
2678           </xsd:annotation>
2679        </xsd:enumeration>
2680        <xsd:enumeration value="count">
2681           <xsd:annotation>
2682              <xsd:documentation>
2683                 Value is the cardinality of the NodeSet resulting from
2684                 the XPath selection. Does not examine the node set
2685                 contents. Does not have any special failure modes.
2686              </xsd:documentation>
2687           </xsd:annotation>
2688        </xsd:enumeration>
2689        <xsd:enumeration value="exists">
2690           <xsd:annotation>
2691              <xsd:documentation>
2692                 Value is 1 if the NodeSet resulting from the XPath
2693                 selection is non-empty, and 0 otherwise. Does not have
2694                 any special failure modes.
2695              </xsd:documentation>
2696           </xsd:annotation>
2697        </xsd:enumeration>
2698        <xsd:enumeration value="total">
2699           <xsd:annotation>
2700              <xsd:documentation>
2701                 Value is the sum of the (floating-point) values of the
2702                 nodes in the NodeSet resulting from the XPath
2703                 selection, all interpreted as doubles. Exception if
2704                 not all values in the NodeSet are interpretable as
2705                 floats. 0 if the NodeSet is empty. If any Node is an
2706                 Element, then all the Text Node children of that
2707                 element are concatenated to form a single string that
2708                 is then used in place of the actually selected Node.
2709                 The evaluation of the containing select:Select term
2710                 MUST fail if any of the strings extracted from the
2711                 NodeSet (as described) cannot be interpreted as a
2712                 valid double value in range (-Inf..+Inf).
2713              </xsd:documentation>
2714           </xsd:annotation>
2715        </xsd:enumeration>
2716     </xsd:restriction>
2717  </xsd:simpleType>
2718
2719  <!-- Non-term elements -->
2720  <xsd:element name="Value" type="xsd:double">
2721     <xsd:annotation>
2722        <xsd:documentation>
2723           A simple element containing a double-precision floating-point
2724           value.
2725        </xsd:documentation>
2726     </xsd:annotation>
2727  </xsd:element>
2728  <xsd:element name="Path" type="select:XPathType">
2729     <xsd:annotation>
2730        <xsd:documentation>
2731           A simple element containing an XPath search specification. It
2732           is expected that this element might well have additional
```

```
            namespace specifiers attached to it.
         </xsd:documentation>
      </xsd:annotation>
   </xsd:element>
   <xsd:element name="Bind">
      <xsd:annotation>
         <xsd:documentation>
            This element allows the specification of bindings of
            namespaces to prefixes for use with the Path element.
         </xsd:documentation>
      </xsd:annotation>
      <xsd:complexType>
         <xsd:complexContent>
            <xsd:extension base="select:empty">
               <xsd:attribute name="namespace" type="xsd:anyURI"
                  use="required" />
               <xsd:attribute name="prefix" type="xsd:NCName"
                  use="required" />
            </xsd:extension>
         </xsd:complexContent>
      </xsd:complexType>
   </xsd:element>
   <xsd:element name="OrderFunction">
      <xsd:annotation>
         <xsd:documentation>
            This element acts as a container for a term, representing the
            whole concept of an ordering function instead of just how to
            score an individual candidate. Conceptually, every candidate
            is scored (as a floating-point value) by the embedded term and
            then the candidates are ordered by their scores, smallest
            first.
         </xsd:documentation>
      </xsd:annotation>
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element ref="select:Term" minOccurs="1" maxOccurs="1" />
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>

   <!-- Terms, the whole point of this schema -->
   <xsd:complexType name="Term">
      <xsd:annotation>
         <xsd:documentation>
            Base type of all terms. Conceptually, a term describes how to
            calculate a double-precision floating-point value, and
            evaluates to either a double value in the range (-Inf..+Inf)
            or a failure. If the evaluation of an overall term expression
            for a context document fails, that context document MUST be
            rejected from the set of legal context documents.
         </xsd:documentation>
      </xsd:annotation>
      <xsd:complexContent>
         <xsd:extension base="select:empty" />
      </xsd:complexContent>
   </xsd:complexType>
```

```
<xsd:element name="Term" abstract="true" type="select:Term">
  <xsd:annotation>
    <xsd:documentation>
      Abstract base element of all terms, used whenever any concrete
      term is usable in a particular location. Conceptually, a term
      describes how to calculate a double-precision floating-point
      value, and evaluates to either a double value in the range
      (-Inf..+Inf) or a failure. If the evaluation of an overall
      term expression for a context document fails, that context
      document MUST be rejected from the set of legal context
      documents.
    </xsd:documentation>
  </xsd:annotation>
</xsd:element>
<xsd:element name="Sum" substitutionGroup="select:Term">
  <xsd:annotation>
    <xsd:documentation>
      A term that calculates its value by summing the values of the
      terms inside it. The evaluation of this term MUST fail if the
      evaluation of any of the contained terms fails.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="select:Term">
        <xsd:sequence>
          <xsd:element ref="select:Term" minOccurs="2"
            maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>
<xsd:element name="Product" substitutionGroup="select:Term">
  <xsd:annotation>
    <xsd:documentation>
      A term that calculates its value by multiplying together the
      values of the terms inside it. The evaluation of this term
      MUST fail if the evaluation of any of the contained terms
      fails.
    </xsd:documentation>
  </xsd:annotation>
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:extension base="select:Term">
        <xsd:sequence>
          <xsd:element ref="select:Term" minOccurs="2"
            maxOccurs="unbounded" />
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>
<xsd:element name="Power" substitutionGroup="select:Term">
  <xsd:annotation>
    <xsd:documentation>
```

```
            A term that calculates its value by raising the value of the
            term inside it to some (fixed) power. The evaluation MUST fail
            if an attempt to calcuate a non-integral power of a negative
            value is made, or if the evaluation of the contained term
            fails.
         </xsd:documentation>
      </xsd:annotation>
      <xsd:complexType>
         <xsd:complexContent>
            <xsd:extension base="select:Term">
               <xsd:sequence>
                  <xsd:element ref="select:Term" minOccurs="1"
                     maxOccurs="1" />
               </xsd:sequence>
               <xsd:attribute name="exponent" type="xsd:double"
                  use="required" />
            </xsd:extension>
         </xsd:complexContent>
      </xsd:complexType>
   </xsd:element>
   <xsd:element name="Negate" substitutionGroup="select:Term">
      <xsd:annotation>
         <xsd:documentation>
            A term that caluclates its value by negating the value of the
            term inside it. The evaluation MUST fail if the evaluation of
            the contained term fails.
         </xsd:documentation>
      </xsd:annotation>
      <xsd:complexType>
         <xsd:complexContent>
            <xsd:extension base="select:Term">
               <xsd:sequence>
                  <xsd:element ref="select:Term" minOccurs="1"
                     maxOccurs="1" />
               </xsd:sequence>
            </xsd:extension>
         </xsd:complexContent>
      </xsd:complexType>
   </xsd:element>
   <xsd:element name="Log" substitutionGroup="select:Term">
      <xsd:annotation>
         <xsd:documentation>
            A term that calculates its value by taking the logarithm of
            the value of the term inside it. The base of the logarithm
            defaults to e. The evaluation of this term MUST fail if the
            evaluation of the contained term fails or results in a
            negative number. The base attribute MUST be positive.
         </xsd:documentation>
      </xsd:annotation>
      <xsd:complexType>
         <xsd:complexContent>
            <xsd:extension base="select:Term">
               <xsd:sequence>
                  <xsd:element ref="select:Term" minOccurs="1"
                     maxOccurs="1" />
               </xsd:sequence>
```

```
            <xsd:attribute name="base" type="xsd:double"
              default="2.71828182846" />
          </xsd:extension>
        </xsd:complexContent>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="Exp" substitutionGroup="select:Term">
      <xsd:annotation>
        <xsd:documentation>
          A term that calculates its value by raising some fixed base to
          the degree of the value of the term inside it. The base of the
          exponent defaults to e. The base MUST be positive. The
          evaluation of this term MUST fail if the evaluation of the
          contained term fails.
        </xsd:documentation>
      </xsd:annotation>
      <xsd:complexType>
        <xsd:complexContent>
          <xsd:extension base="select:Term">
            <xsd:sequence>
              <xsd:element ref="select:Term" minOccurs="1"
                maxOccurs="1" />
            </xsd:sequence>
            <xsd:attribute name="base" type="xsd:double"
              default="2.71828182846" />
          </xsd:extension>
        </xsd:complexContent>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="Abs" substitutionGroup="select:Term">
      <xsd:annotation>
        <xsd:documentation>
          A term that calculates its value by taking the absolute value
          of the term inside it (i.e. negating the value if-and-only-if
          the value is negative). The evaluation of this term MUST fail
          if the evaluation of the contained term fails.
        </xsd:documentation>
      </xsd:annotation>
      <xsd:complexType>
        <xsd:complexContent>
          <xsd:extension base="select:Term">
            <xsd:sequence>
              <xsd:element ref="select:Term" minOccurs="1"
                maxOccurs="1" />
            </xsd:sequence>
          </xsd:extension>
        </xsd:complexContent>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="Constant" substitutionGroup="select:Term">
      <xsd:annotation>
        <xsd:documentation>
          A term whose value is a constant double-precision
          floating-point value. The evaluation of this term MUST NOT
          fail.
        </xsd:documentation>
```

```
    </xsd:annotation>
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="select:Term">
          <xsd:sequence>
            <xsd:element ref="select:Value" />
          </xsd:sequence>
        </xsd:extension>
      </xsd:complexContent>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Select" substitutionGroup="select:Term">
    <xsd:annotation>
      <xsd:documentation>
        A term whose value is computed from the set of nodes returned
        by an XPath search conducted over some document (i.e. the
        context document which we are computing an overall value for).
        The computation from the NodeSet is to be performed as
        follows: if a baseInstant attribute is specified on the Select
        element, then the NodeSet is to be replaced with another
        NodeSet where each node of the set is replaced by a node (of
        type xsd:double) that represents the number of seconds from
        the basInstant to the time parsed out of the original Node; if
        no baseInstant is specified, the node set MUST NOT be
        interpreted as containing time instants. Then the set of Nodes
        is converted to a double according to the action specified in
        the operation attribute; see the documentation of the
        select:SelectOperation type for details. The evaluation of
        this term MUST fail if the contained select:Path element does
        not hold a legal XPath expression; see the documentation of
        the select:SelectOperation type for other failure modes.
      </xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
      <xsd:complexContent>
        <xsd:extension base="select:Term">
          <xsd:sequence>
            <!-- Question: what about the namespace map? -->
            <xsd:element ref="select:Bind" maxOccurs="unbounded"
              minOccurs="0" />
            <xsd:element ref="select:Path" />
          </xsd:sequence>
          <xsd:attribute name="baseInstant" type="xsd:dateTime"
            use="optional">
            <xsd:annotation>
              <xsd:documentation>
                When nodes in the NodeSet returned from the
                XPath search represent time instants, they
                should be converted into numeric values by
                taking the number of seconds from the instant
                defined in this attribute and the instant in
                the node. It SHOULD be an error to set this
                attribute if the nodes are not interpretable
                as times. If any node is an Element, it should
                be parsed as if it was the concatenation of
                all the children of the element that are Text
```

```
                        nodes.
                    </xsd:documentation>
                </xsd:annotation>
            </xsd:attribute>
            <xsd:attribute name="operation"
                type="select:SelectOperation" default="first"
                use="optional">
                <xsd:annotation>
                    <xsd:documentation>
                        How to convert a NodeSet into a single double.
                        Defaults to taking the first element of the
                        NodeSet and just using that. The evaluation
                        failure modes of the containing select:Select
                        term are partially determined by this
                        attribute; see the documentation of the
                        select:SelectOperation type for details.
                    </xsd:documentation>
                </xsd:annotation>
            </xsd:attribute>
        </xsd:extension>
    </xsd:complexContent>
</xsd:complexType>
</xsd:element>
<xsd:element name="Bound" substitutionGroup="select:Term">
    <xsd:annotation>
        <xsd:documentation>
            This element evaluates its contained term and returns it only
            if it is within the range of values described by the pair of
            bounding attributes. If the bounding constraints are not
            satisfied, the whole term evaluation MUST fail. The evaluation
            of this term MUST fail if the evaluation of the contained term
            fails.
        </xsd:documentation>
    </xsd:annotation>
    <xsd:complexType>
        <xsd:complexContent>
            <xsd:extension base="select:Term">
                <xsd:sequence>
                    <xsd:element ref="select:Term" minOccurs="1"
                        maxOccurs="1" />
                </xsd:sequence>
                <xsd:attribute name="lowerBound" type="xsd:double"
                    use="optional">
                    <xsd:annotation>
                        <xsd:documentation>
                            This indicates the lower (inclusive) bound of
                            the term's value. If omitted, there is no
                            lower bound (or, alternatively, the lower
                            bound is negative infinity).
                        </xsd:documentation>
                    </xsd:annotation>
                </xsd:attribute>
                <xsd:attribute name="upperBound" type="xsd:double"
                    use="optional">
                    <xsd:annotation>
                        <xsd:documentation>
```

```
                        This indicates the upper (inclusive) bound of
                        the term's value. If omitted, there is no
                        upper bound (or, alternatively, the upper
                        bound is positive infinity).
                    </xsd:documentation>
                </xsd:annotation>
            </xsd:attribute>
        </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
 </xsd:element>
 <xsd:element name="OneOf" substitutionGroup="select:Term">
  <xsd:annotation>
   <xsd:documentation>
    This element has the value of the first contained term in it
    that evaluates successfully. All subsequent contained terms
    after the first successfully-computed one MUST NOT be
    evaluated. The evaluation of this term MUST fail if and only
    if the evaluation of all contained terms fails.
   </xsd:documentation>
  </xsd:annotation>
  <xsd:complexType>
   <xsd:complexContent>
    <xsd:extension base="select:Term">
     <xsd:sequence>
      <xsd:element ref="select:Term" minOccurs="1"
       maxOccurs="unbounded" />
     </xsd:sequence>
    </xsd:extension>
   </xsd:complexContent>
  </xsd:complexType>
 </xsd:element>
</xsd:schema>
```