

1 **OGSA[®] Resource Selection Services: Specification**

2 Status of This Document

3 This document provides information to the Grid community on the definitions of the OGSA Re-
4 source Selection service interfaces. Distribution is unlimited.

5 Copyright Notice

6 Copyright © Open Grid Forum 2005-2007. All Rights Reserved.

7 Trademark

8 Open Grid Services Architecture and OGSA are registered trademarks of the Open Grid Forum.

9 Abstract

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

17 Contents

18	Abstract.....	1
19	1. Introduction	3
20	2. Notational Conventions	3
21	3. What are the Resource Selection Services?	3
22	4. Specification of a Candidate Set Generator	4
23	4.1 CSG Abstract Functional Interface	4
24	4.2 CSG Basic Input/Output Template Model.....	5
25	4.3 Example Request for Candidates.....	6
26	4.4 Common CSG Properties	7
27	5. Specification of the Basic Execution Planning Service Profile	7
28	5.1 Contents of the <i>CandidateQualityProperties</i>	8
29	5.2 Standard Interpretation of the Candidate Execution Plan	11
30	5.3 Example Candidate Execution Plan	11
31	6. Concrete Rendering of the Basic Execution Planning Service	12
32	6.1 Conceptual Operation of the Basic-EPS.....	12
33	6.2 Specialization of the Basic-EPS to WS-RF.....	12
34	6.3 Specialization of the EPS to WS-Transfer/WS-Enumeration	18
35	7. Simple Candidate Ordering Language.....	19

36	7.1	Organizing Principles of SCOL for Candidate Generation	20
37	7.2	SCOL Syntax and Semantics	20
38	7.3	Example <i>OrderFunction</i>	25
39	7.4	Extending SCOL	26
40	8.	Security Considerations	26
41	9.	Contributors.....	26
42	10.	Intellectual Property Statement	27
43	11.	Disclaimer.....	27
44	12.	Full Copyright Notice	27
45	13.	References.....	27
46		Appendix A: Normative Schemas.....	29
47	A.1	XML Schema for CSG Core.....	29
48	A.2	XML Schema for EPS Core	34
49	A.3	XML Schema for Basic-EPS-RF	39
50	A.4	WSDL for Basic-EPS-RF	44
51	A.5	XML Schema for Basic-EPS-XF	50
52	A.6	WSDL for Basic-EPS-XF.....	51
53	A.7	Supplemental XML Schema for WS-Enumeration	53
54	A.8	XML Schema for SCOL.....	54
55			

55 1. Introduction

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

64 2. Notational Conventions

65 The key words “MUST,” “MUST NOT,” “REQUIRED,” “SHALL,” “SHALL NOT,” “SHOULD,”
 66 “SHOULD NOT,” “RECOMMENDED,” “MAY,” and “OPTIONAL” are to be interpreted as de-
 67 scribed in RFC 2119[Brad]

68 The specification provides a pseudo-schema for each component as part of the description of the
 69 component. They use BNF-style conventions for attributes and elements: ‘?’ denotes zero or one
 70 occurrences; ‘*’ denotes zero or more occurrences; ‘+’ denotes one or more occurrences. Attrib-
 71 utes are conventionally assigned a value that corresponds to their type, as defined in the norma-
 72 tive schema.

```
73 <!-- sample pseudo-schema -->
74 <defined_element
75     required_attribute_of_type_string="xsd:string"
76     optional_attribute_of_type_int="xsd:int" ? >
77   <required_element />
78   <optional_element /> ?
79   <zero_or_more_of_this_element /> *
80   <one_or_more_of_this_element /> +
81 </defined_element>
```

82 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

83 3. What are the Resource Selection Services?

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

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

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

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

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

111 **4. Specification of a Candidate Set Generator**

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

120 **4.1 CSG Abstract Functional Interface**

121 Abstractly, the only CSG operation is:

```
122     getCandidates: InputDescription * optional(CandidateOrdering)  
123     → orderedSet(OutputCandidate)
```

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

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

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

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

139 4.2 CSG Basic Input/Output Template Model

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

142 All named elements in this section MUST use the namespace:

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

144 All CSG renderings MUST support a way of reading a property of every CSG instance, *Usage-*
 145 *Profile*, whose contents MUST be a set of URIs characterizing the contents of the *Input-*
 146 *Description* and *CandidateDescription* elements (described below) and how they are intended to
 147 be used. All CSG renderings MUST support a way of reading a property of every CSG instance,
 148 *OrderingLanguages*, whose contents MUST be a set of URIs characterizing the permitted formats
 149 of the *CandidateOrdering* elements (described below). Each CSG profile and each ordering lan-
 150 guage MUST define at least one URI; URIs beneath `http://schemas.ogf.org/rss/` are reserved for
 151 OGF use when not otherwise described.

152 4.2.1 Input Description Model

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

158 The contents of the *InputDescription* element MUST be determined by the profile specified in the
 159 *UsageProfile* element, if given; if omitted, the CSG SHOULD use a default profile but MAY in-
 160 stead guess based on the content of the *InputDescription*. The *CandidateOrdering* element
 161 MUST have a *dialect* element whose content MUST be an arbitrary URI that describes the con-
 162 tent of the element and the manner in which it is to be used to generate the ordered set of candi-
 163 dates.

164 Each *PermittedCandidateExtension* element must describe (using a QName) the extension ele-
 165 ment to which it is referring, and SHOULD give (via an optional *type* attribute) where in a candi-
 166 date the element whose name is the content of the *PermittedCandidateExtension* element may
 167 be placed. The *type* attribute MUST be of type *ExtensionType*, which is an enumeration consist-
 168 ing of "functional" and "quality", defaulting to "functional" if unspecified. Note that the presence of
 169 a *PermittedCandidateExtension* SHALL NOT force the processing CSG to generate a *Candidate*
 170 that contains such an extension element.

```
171 <csg:RequestCandidates>
172   <csg:UsageProfile> xsd:anyURI </csg:UsageProfile> ?
173   <csg:InputDescription> ... </csg:InputDescription>
174   <csg:CandidateOrdering dialect="xsd:anyURI"> ... </csg:CandidateOrdering> ?
175   <csg:PermittedCandidateExtension type="csg:ExtensionType"?>
176     xsd:QName
177   </csg:PermittedCandidateExtension> *
178   ...
179 </csg:RequestCandidates>
```

180 4.2.2 Output Candidate Model

181 Each candidate returned by the CSG MUST be an element named *Candidate*, whose contents
 182 MUST consist of the following three elements, *CandidateDescription*, *CandidateQuality-*
 183 *Properties*, *CandidateValidity*, followed by any arbitrary sequence of elements from XML name-
 184 spaces other than the CSG namespace. Those added elements beyond what is required in the

185 profile MUST be described by *PermittedCandidateExtension* elements (with type equal to “func-
186 tional”) in the *RequestCandidates* which caused the containing *Candidate* element to be gener-
187 ated.

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

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

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

```
207 <csg:Candidate>
208   <csg:CandidateDescription> ... </csg:CandidateDescription>
209   <csg:CandidateQualityProperties> ... </csg:CandidateQualityProperties>
210   <csg:CandidateValidity notBefore="xsd:dateTime" notAfter="xsd:dateTime" />
211   <xsd:any namespace="##other" processContents="lax" /> *
212 </csg:Candidate>
```

213 4.3 Example Request for Candidates

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

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

```
221 <b>csg:RequestCandidates</b>
222   <b>csg:UsageProfile</b> http://schemas.ogf.org/rss/2007/03/eps/basic </b>csg:UsageProfile>
223   <b>csg:InputDescription</b>
224     <jSDL:JobDefinition>
225       <jSDL:JobDescription>
226         <jSDL:Application>
227           <jSDL:ApplicationName> BLAST </jSDL:ApplicationName>
228         </jSDL:Application>
229       </jSDL:JobDescription>
230     </jSDL:JobDefinition>
231   </b>csg:InputDescription>
232   <b>csg:CandidateOrdering dialect="urn:private:demo">
233     Order by Amortized Price
234   </b>csg:CandidateOrdering>
235   <b>csg:PermittedCandidateExtension type="functional">
```

```

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

```

240 4.4 Common CSG Properties

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

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

247 4.4.1 The *UsageProfile* Property

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

252 4.4.2 The *SupportedOrderingLanguage* Property

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

257 5. Specification of the Basic Execution Planning Service Profile

258 A Basic Execution Planning Service (Basic-EPS) is a profile of the Candidate Set Generator ab-
 259 stract service where the contents of the *InputDescription* element MUST be a JSDL *JobDefinition*
 260 document, and the contents of the *CandidateDescription* element must be a sequence of a JSDL
 261 *JobDefinition* and an element, called *BESReference*, containing a WS-Addressing[WSA] End-
 262 point Reference to a service implementing the Basic Execution Service port-type. Within the con-
 263 text of the Basic-EPS profile, the contents of a *Candidate* element may be referred to as a candi-
 264 date execution plan (CEP). The usage model supported by this usage profile is that the consumer
 265 of the candidate execution plan takes action on the CEP by submitting the *JobDefinition* within
 266 the CEP to the Basic Execution Service described by the CEP. The *CandidateQualityProperties*
 267 element MUST support the elements described further below in this section. Note that the re-
 268 quirements of the CSG *PermittedCandidateExtension* MUST still be respected.

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

271 The URI for this usage profile is:

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

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

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

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

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

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

281 MAY be selected, or additional file staging requirements MAY be added. This list of ways of varia-
282 tion is not intended to be exhaustive.

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

```
284 <csg:RequestCandidates>
285   <csg:UsageProfile> xsd:anyURI </csg:UsageProfile> ?
286   <csg:InputDescription>
287     <jsdl:JobDefinition> ... </jsdl:JobDefinition>
288   </csg:InputDescription>
289   <csg:CandidateOrdering> ... </csg:CandidateOrdering> ?
290   <csg:PermittedCandidateExtension csg:type="csg:ExtensionType"?>
291     xsd:QName
292   </csg:PermittedCandidateExtension> *
293 </csg:RequestCandidates>
```

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

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

```
298 <csg:Candidate>
299   <csg:CandidateDescription>
300     <jsdl:JobDefinition> ... </jsdl:JobDefinition>
301     <eps-basic:BESReference>
302       ws-a:EndpointReferenceType
303     </eps-basic:BESReference>
304   </csg:CandidateDescription>
305   <csg:CandidateQualityProperties>
306     <eps-basic:Prices ... /> ?
307     <eps-basic:StartTimes ... /> ?
308     <eps-basic:StartDelays ... /> ?
309     <eps-basic:EndTimes ... /> ?
310     <eps-basic:ExecutionDuration ... /> ?
311     <xsd:any namespace="##other" processContents="lax" /> *
312   </csg:CandidateQualityProperties>
313   <csg:CandidateValidity csg:notBefore="xsd:dateTime" csg:notAfter="xsd:dateTime" />
314   <xsd:any namespace="##other" processContents="lax" /> *
315 </csg:Candidate>
```

316 5.1 Contents of the *CandidateQualityProperties*

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

```
325 <csg:CandidateQualityProperties>
326   <eps-basic:Prices ... /> ?
327   <eps-basic:StartTimes ... /> ?
328   <eps-basic:StartDelays ... /> ?
329   <eps-basic:EndTimes ... /> ?
330   <eps-basic:ExecutionDuration ... /> ?
```



```

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

```

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

338 5.1.1 The *Prices* Element

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

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

351 The content of the *PriceRange* element which MAY consist of mixed strings and elements, if pre-
 352 sent, MUST be used to describe the origin of the particular charge associated with that
 353 *PriceRange*. This specification does not ascribe any particular meaning to this content, and con-
 354 sumers of the candidate MAY ignore it. The *PriceRange* element MUST have two attributes, *from*
 355 and *to*, that describe the range of prices in that particular range. The *from* attribute MUST NOT be
 356 greater than the *to* attribute. The *from* attribute MAY be equal to the *to* attribute. Either attribute
 357 MAY be negative; this MUST describe the case where the client of the BES is paid for their cus-
 358 tom.

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

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

362 Pseudo-schema for the *Prices* element:

```

363 <eps-basic:Prices>
364   <eps-basic:Price eps-basic:currency="xsd:anyURI">
365     <eps-basic:PriceRange eps-basic:from="xsd:double" eps-basic:to="xsd:double">
366       <xsd:any##other> * | xsd:string
367     </eps-basic:PriceRange> +
368   </eps-basic:Price> +
369 </eps-basic:Prices>

```

370 5.1.2 The *StartTimes* Element

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

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

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

378 Pseudo-schema for the *StartTimes* element:

```

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

```

382 5.1.3 The *StartDelays* Element

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

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

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

392 Pseudo-schema for the *StartDelays* element:

```

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

```

396 5.1.4 The *EndTimes* Element

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

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

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

404 Pseudo-schema for the *EndTimes* element:

```

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

```

408 5.1.5 The *ExecutionDuration* Element

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

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

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

418 Pseudo-schema for the *ExecutionDuration* element:

```

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

```

422 5.2 Standard Interpretation of the Candidate Execution Plan

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

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

434 5.3 Example Candidate Execution Plan

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

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

```

442 <csq:Candidate
443   xmlns:csg="http://schemas.ogf.org/rss/2007/03/csg"
444   xmlns:eps-basic="http://schemas.ogf.org/rss/2007/03/eps/basic"
445   xmlns:jSDL="http://schemas.ggf.org/jSDL/2005/11/jSDL"
446   xmlns:jSDL-hpcp="http://schemas.ggf.org/jSDL/2006/07/jSDL-hpcp"
447   xmlns:bes="http://schemas.ggf.org/bes/2006/08/bes-factory"
448   xmlns:wsa="http://www.w3.org/2005/08/addressing"
449   xmlns:wsaw="http://www.w3.org/2005/03/addressing/wSDL"
450   xmlns:serv="http://some.service.com/private">
451   <csq:CandidateDescription>
452     <jSDL:JobDefinition>
453       <jSDL:JobDescription>
454         <jSDL:JobIdentification>
455           <jSDL:JobName>C.E.P. Example</jSDL:JobName>
456         </jSDL:JobIdentification>
457         <jSDL:Application>
458           <jSDL:ApplicationName> BLAST </jSDL:ApplicationName>
459           <jSDL-hpcp:HPCProfileApplication>
460             <jSDL-hpcp:Output>blast_out.txt</jSDL-hpcp:Output>
461             <jSDL-hpcp:Error>blast_err.txt</jSDL-hpcp:Error>
462           </jSDL-hpcp:HPCProfileApplication>
463         </jSDL:Application>
464       </jSDL:JobDescription>
465     </jSDL:JobDefinition>
466     <eps-basic:BESReference>
467       <wsa:Address> http://some.service.com/Execution </wsa:Address>
468       <wsa:Metadata>
469         <wsaw:InterfaceName>
470           bes:BESFactoryPortType
471         </wsaw:InterfaceName>
472       </wsa:Metadata>
473     </eps-basic:BESReference>
474   </csq:CandidateDescription>
475 </csq:Candidate>
  
```

```

474         <serv:Backend> bigiron.service.com </serv:Backend>
475         </wsa:ReferenceParameters>
476         </eps-basic:BESReference>
477     </csg:CandidateDescription>
478     <csg:CandidateQualityParameters>
479         <eps-basic:Prices>
480             <eps-basic:Price eps-basic:currency="currency:USD">
481                 <eps-basic:PriceRange eps-basic:from="0" eps-basic:to="0" />
482             </eps-basic:Price>
483         </eps-basic:Prices>
484         <eps-basic:StartDelay>
485             <eps-basic:DelayRange eps-basic:from="0" eps-basic:to="120" />
486         </eps-basic:StartDelay>
487         <eps-basic:ExecutionDuration>
488             <eps-basic:RuntimeRange eps-basic:from="240" eps-basic:to="300" />
489         </eps-basic:ExecutionDuration>
490     </csg:CandidateQualityParameters>
491     <csg:CandidateValidity csg:from="2006-12-24T00:00:00Z"
492         csg:to="2006-12-25T23:59:59Z"/>
493 </csg:Candidate>

```

494 6. Concrete Rendering of the Basic Execution Planning Service

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

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

505 6.1 Conceptual Operation of the Basic-EPS

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

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

514 6.2 Specialization of the Basic-EPS to WS-RF

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

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

523 <http://schemas.ogf.org/rss/2007/03/eps/wsrf>

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

526 6.2.1 Operations of the Basic-EPS-RF

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

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

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

540 • The *CandidateOrdering* element inside the *RequestCandidates* element MUST have a
541 “*dialect*” attribute (with a content model permitting any URI) and MUST allow any single
542 element as content. The *dialect* describes which ordering language is to be used to order
543 the set of candidates, and the content of the *CandidateOrdering* element MUST follow
544 the requirements of the language described by the *dialect*; the SCOL (see Section 0) is
545 the only language that MUST be supported.

546 • The *MaxCandidatesPerTransfer* element MUST have a positive-integer content model,
547 and MUST describe the maximum number of Candidate Execution Plans that the user
548 wishes to receive per SOAP response. If not present, the Basic-EPS-RF service
549 SHOULD use the *MaxCandidatesPerTransfer* property to define the limit.

550 • The *MaxCandidatesTotal* element MUST have a positive-integer content model, and
551 MUST describe the maximum number of Candidate Execution Plans that the user wishes
552 to receive overall (permitting implementations to clean up resources more efficiently
553 when that many CEPs have been retrieved). If not present, the Basic-EPS-RF SHOULD
554 NOT place any limit on the number of CEPs to be returned, and transfers MUST be done
555 in stages using an ISS-RF unless the total number of responses is less than the value
556 described within the *MaxCandidatesPerTransfer* element.

557 • The *MaxResponseOctets* element MUST have a positive integer content model, and
558 MUST describe the maximum number of octets (bytes) that the user wishes to receive in
559 any SOAP response that returns more than one *Candidate* elements. If not present, the
560 Basic-EPS-RF service SHOULD use the *MaxResponseOctets* property of itself to define
561 the limit, if present. If that is also absent, no statement is made by this specification about
562 the number of octets in any response. It is RECOMMENDED that neither the Basic-EPS-
563 RF nor the ISS-RF transfer very large numbers of Candidate Execution Plans in a single
564 message in order that clients of the service can process the messages using implemen-
565 tations based on W3C DOM.

566 • The *IterableSetLifetime* element MUST have an XML Schema time duration content
567 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.

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

```
570 <soap:Envelope>
571   <soap:Header>
572     <wsa:Action>
573       http://schemas.ogf.org/rss/2007/03/eps/wsrf/GetCandidates
574     </wsa:Action>
575     ...
576   </soap:Header>
577   <soap:Body>
578     <eps-wsrf:GetCandidatesRequest>
579       <csg:RequestCandidates> ... </csg:RequestCandidates>
580       <eps-wsrf:MaxCandidatesPerTransfer>
581         xsd:positiveInteger
582       </eps-wsrf:MaxCandidatesPerTransfer> ?
583       <eps-wsrf:MaxCandidatesTotal>
584         xsd:positiveInteger
585       </eps-wsrf:MaxCandidatesTotal> ?
586       <eps-wsrf:MaxResponseOctets>
587         xsd:positiveInteger
588       </eps-wsrf:MaxResponseOctets> ?
589       <eps-wsrf:IterableSetLifetime>
590         xsd:duration
591       </eps-wsrf:IterableSetLifetime> ?
592     </eps-wsrf:GetCandidatesRequest>
593   </soap:Body>
594 </soap:Envelope>
```

595 The *GetCandidatesResponse* document MUST contain a sequence of zero, one or more
596 *Candidate* elements, each of which MUST match the profile for the input document selected, fol-
597 lowed optionally by an *IterableSetReference* element. Clients of the Basic-EPS-RF MUST sup-
598 port the receipt of an *IterableSetReference*.

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

```
606 <soap:Envelope>
607   <soap:Header>
608     <wsa:Action>
609       http://schemas.ogf.org/rss/2007/03/eps/wsrf/GetCandidatesResponse
610     </wsa:Action>
611     ...
612   </soap:Header>
613   <soap:Body>
614     <eps-wsrf:GetCandidatesResponse>
615       <csg:Candidate> ... </csg:Candidate> *
616       <eps-wsrf:IterableSetReference>
617         wsa:EndpointReferenceType
618       </eps-wsrf:IterableSetReference> ?
619     </eps-wsrf:GetCandidatesResponse>
```

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

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

- 626 • An *UnsupportedCSGProfileFault* MUST be indicated when the client supplies a *RequestCandidates*
 627 element that does not match any supported profile. The content model of this
 628 fault (other than that it MUST be a WS-BaseFault with a particular name) is not defined.
- 629 • An *UnsupportedOrderingLanguageFault* MUST be indicated when client asks to order the
 630 Candidate Execution Plans using a language that is not supported or if the content of the
 631 Ordering element is malformed. This fault MUST NOT be indicated if the *Ordering* contains
 632 a valid SCOL document. The content model of this fault (other than that it MUST be
 633 a WS-BaseFault with a particular name) is not defined.
- 634 • A *BadCandidateCardinalityRulesFault* SHOULD be indicated if the client requests a
 635 number of rules per transfer that the Basic-EPS-RF does not support. The content model
 636 of this fault (other than that it MUST be WS-BaseFault with a particular name) is not defined.
 637
- 638 • An *UnableToGenerateCandidatesFault* SHOULD be indicated in other failure cases. The
 639 content model of this fault (other than that it MUST be a WS-BaseFault with a particular
 640 name) is not defined, but the fault SHOULD describe what the failure was. This fault
 641 MUST NOT be generated if there are simply no possible *Candidates* that match the
 642 *RequestCandidates*.
- 643 • The Basic-EPS-RF SHOULD throw a *ResourceUnknownFault* (from the standard WS-RF
 644 fault set) if it receives a request for a Basic-EPS-RF resource that it does not know about.

645 6.2.2 Operations of the Iterable Set Service (ISS-RF)

646 The ISS-RF MUST support the operation *GetMoreCandidates*, MUST support the operations of
 647 WS-ResourceLifetime[WSRL], and MUST support the read operations described in WS-ResourceProperties[WSRP];
 648 supporting the update operations described in WS-ResourceProperties is NOT RECOMMENDED.
 649

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

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

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

```

669     ...
670     </soap:Header>
671     <soap:Body>
672         <eps-wsrf:GetMoreCandidatesRequest>
673             <eps-wsrf:Count> xsd:positiveInteger </eps-wsrf:Count> ?
674         </eps-wsrf:GetMoreCandidatesRequest>
675     </soap:Body>
676 </soap:Envelope>

```

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

- 679 • 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).

```

685 <soap:Envelope>
686     <soap:Header>
687         <wsa:Action>
688             http://schemas.ogf.org/rss/2007/03/eps/wsrf/GetMoreCandidatesResponse
689         </wsa:Action>
690         ...
691     </soap:Header>
692     <soap:Body>
693         <eps-wsrf:GetMoreCandidatesResponse>
694             <csg:Candidate> ... </csg:Candidate> *
695         </eps-wsrf:GetMoreCandidatesResponse>
696     </soap:Body>
697 </soap:Envelope>

```

698 6.2.3 Properties

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

706 *csg:UsageProfile* (see §4.4.1)

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

712 *csg:SupportedOrderingLanguages* (see §4.4.2)

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

718 *eps-wsrf:MaxCandidatesPerTransfer*

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

724 *eps-wsrf:MaxResponseOctets*

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

732 *eps-wsrf:DefaultIterableSetLifetime*

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

739 *eps-wsrf:CandidatesTransferredSoFar*

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

745 *eps-wsrf:AnyMoreCandidatesToTransfer*

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

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

```
752 <csg:UsageProfile> xsd:anyURI </csg:UsageProfile> +
753 <csg:SupportedOrderingLanguages> xsd:anyURI </csg:SupportedOrderingLanguages> +
754 <eps-wsrf:MaxCandidatesPerTransfer>
755   xsd:positiveInteger
756 </eps-wsrf:MaxCandidatesPerTransfer>
757 <eps-wsrf:MaxResponseOctets> xsd:positiveInteger <eps-wsrf:MaxResponseOctets> ?
758 <eps-wsrf:DefaultIterableSetLifetime> xsd:duration </eps-wsrf:DefaultIterableSetLifetime> ?
```

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

```
760 <eps-wsrf:MaxCandidatesPerTransfer>
761   xsd:positiveInteger
762 </eps-wsrf:MaxCandidatesPerTransfer>
763 <eps-wsrf:MaxResponseOctets> xsd:positiveInteger <eps-wsrf:MaxResponseOctets> ?
```

```

764 <eps-wsrf:CandidatesTransferredSoFar>
765     xsd:nonNegativeInteger
766 </eps-wsrf:CandidatesTransferredSoFar> ?
767 <eps-wsrf:AnyMoreCandidatesToTransfer>
768     xsd:boolean
769 </eps-wsrf:AnyMoreCandidatesToTransfer> ?

```

770 6.3 Specialization of the EPS to WS-Transfer/WS-Enumeration

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

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

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

779 6.3.1 Operations of the Basic-EPS-XF

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

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

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

```

795 <soap:Envelope>
796   <soap:Header>
797     <wsa:Action>
798       http://schemas.ogf.org/rss/2007/10/eps/transfer/EnumerateCandidates
799     </wsa:Action>
800     <wsa:MessageID> xsd:anyURI </wsa:MessageID>
801     <wsa:To> xsd:anyURI </wsa:To>
802     ...
803   </soap:Header>
804   <soap:Body>
805     <eps-xfer:EnumerateCandidatesRequest>
806       <csg:RequestCandidates> ... </csg:RequestCandidates>
807       <wsen:EndTo> wsa:EndpointReferenceType </wsen:EndTo> ?
808       <wsen:Expires> [ xsd:dateTime | xsd:duration ] </wsen:Expires> ?
809     </eps-xfer:EnumerateCandidatesRequest>
810   </soap:Body>
811 </soap:Envelope>

```

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

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

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

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

826 All faults MUST be indicated (using SOAP 1.2[SOAP] rules²) via the *s12:Code/s12:Value* field
 827 being *s12:Sender*, the *s12:Code/s12:Subcode/s12:Value* field being the qualified name for the
 828 fault, and the *s12:Reason/s12:Text* field containing a human-readable description of the detailed
 829 reason for the fault. This parallels the encoding used for faults in WS-Enumeration.

830 6.3.2 Resource Representation of the Basic-EPS-XF

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

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

841 No other elements are required to be supported.

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

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

844 7. Simple Candidate Ordering Language

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

848 `http://schemas.opengis.net/rss/2006/12/scol`

849 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.

850 7.1 Organizing Principles of SCOL for Candidate Generation

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

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

866 7.1.1 Implementation Notes

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

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

875 7.2 SCOL Syntax and Semantics

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

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

884 Syntax:

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

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

888 7.2.1 Term: *Constant*

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

892 Syntax:

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

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

895 **<Constant><Value>42</Value></Constant>**

896 7.2.2 Term: *Sum*

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

900 Syntax:

901 **<Sum> <Term ... /> + </Sum>**

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

904 **<Sum>**
905 <Constant><Value>2</Value></Constant>
906 <Constant><Value>3</Value></Constant>
907 <Constant><Value>5</Value></Constant>
908 **</Sum>**

909 Example: The value of this term is the difference between the two contained *Constant* terms, *i.e.*,
910 70.

911 **<Sum>**
912 <Constant><Value>99</Value></Constant>
913 <Negate>
914 <Constant><Value>29</Value></Constant>
915 </Negate>
916 **</Sum>**

917 7.2.3 Term: *Product*

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

921 Syntax:

922 **<Product> <Term ... /> * </Product>**

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

925 **<Product>**
926 <Constant><Value>2</Value></Constant>
927 <Constant><Value>3</Value></Constant>
928 <Constant><Value>5</Value></Constant>
929 **</Product>**

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

931 **<Product>**
932 <Constant><Value>99</Value></Constant>
933 <Power exponent="-1">
934 <Constant><Value>9</Value></Constant>
935 </Power>
936 **</Product>**

937 7.2.4 Term: *Power*

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

943 Syntax:

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

945 Example: The value of this *Power* is the square of the value of the contained *Constant* term, *i.e.*,
 946 11.

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

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

```
951 <Product>
952   <Constant><Value>99</Value></Constant>
953   <Power exponent="-1">
954     <Constant><Value>9</Value></Constant>
955   </Power>
956 </Product>
```

957 7.2.5 Term: *Negate*

958 The *Negate* term MUST contain one nested term. It MUST evaluate to the negation of the value
 959 of the nested term. If the nested term evaluates to a failure, the *Negate* MUST evaluate to a fail-
 960 ure.

961 Syntax:

```
962 <Negate> <Term ... /> </Negate>
```

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

```
965 <Negate>
966   <Constant><Value>42</Value></Constant>
967 </Negate>
```

968 Example: The value of this term is the difference between the two contained *Constant* terms, *i.e.*,
 969 70.

```
970 <Sum>
971   <Constant><Value>99</Value></Constant>
972   <Negate>
973     <Constant><Value>29</Value></Constant>
974   </Negate>
975 </Sum>
```

976 7.2.6 Term: *Log*

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

981 term MUST evaluate to a failure if the nested term evaluates to a failure or the base is not a posi-
982 tive number.

983 Syntax:

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

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

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

990 7.2.7 Term: *Exp*

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

997 Syntax:

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

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

```
1001 <Exp base="10">  
1002   <Constant><Value>3</Value></Constant>  
1003 </Exp>
```

1004 7.2.8 Term: *Abs*

1005 The *Abs* term MUST contain one nested term. It MUST evaluate to the absolute value of the
1006 value of the nested term (*i.e.*, with the sign of the value negated if and only if it is negative). The
1007 *Abs* term MUST evaluate to a failure if the nested term evaluates to a failure.

1008 Syntax:

```
1009 <Abs> <Term ... /> </Abs>
```

1010 Example: The value of this *Abs* is the positive magnitude of the value of the contained *Constant*
1011 term, *i.e.*, 19.

```
1012 <Abs>  
1013   <Constant><Value>-19</Value></Constant>  
1014 </Abs>
```

1015 7.2.9 Term: *Select*

1016 The *Select* term MUST contain a single *Path* element, and must permit optional “*baseInstant*” and
1017 “*operation*” attributes. The *Path* element MUST contain an XPath 2.0[XP] expression. If present,
1018 the “*baseInstant*” attribute MUST contain a reference to a particular time instant. The “*operation*”
1019 attribute MUST be one of the strings “*first*”, “*count*”, “*exists*” and “*total*” and MUST, if absent, be
1020 treated as if it was present with the value “*first*”. The behavior of the terms according to the opera-
1021 tions is as follows:

1022 *first* If the *operation* is “*first*”, the *Select* term MUST evaluate to the value of the text content of
1023 the first node selected by applying the XPath expression to the context document (*i.e.*,
1024 the Candidate Execution Plan). If the *baseInstant* attribute is present, the value of the

- 1025 *Select* MUST be the number of seconds after the *baseInstant* that the time instant value
 1026 selected refers to. If the *baseInstant* attribute is absent, the value of the *Select* MUST be
 1027 the floating-point interpretation of the text node. The *Select* term MUST evaluate to a fail-
 1028 ure if no nodes are selected, if the text content of the first selected node cannot be inter-
 1029 preted as a time instant when the *baseInstant* attribute is present, or if the text content of
 1030 the selected node cannot be interpreted as a floating-point number when the *baseInstant*
 1031 attribute is absent.
- 1032 *count* If the *operation* is “*count*”, the *Select* term MUST evaluate to the number of nodes se-
 1033 lected by applying the XPath expression to the context document. The content of those
 1034 nodes SHOULD be ignored.
- 1035 *exists* If the *operation* is “*exists*”, the *Select* term MUST evaluate to 1 if the number of nodes
 1036 selected by applying the XPath expression to the context document is at least 1, and 0
 1037 otherwise. The content of those nodes SHOULD be ignored.
- 1038 *total* If the *operation* is “*total*”, the *Select* term MUST evaluate to the sum of the value of each
 1039 text node selected by applying the XPath expression to the context document. The value
 1040 of each selected node MUST be computed according to the same rules as for the “*first*”
 1041 operation (*i.e.*, as seconds from *baseInstant* if that is present, or as floating-point values if
 1042 *baseInstant* is absent). The *Select* term MUST evaluate to 0 if no nodes are matched.
 1043 The *Select* term MUST evaluate to a failure if the text content of any selected node can-
 1044 not be interpreted as a time instant (when *baseInstant* is present) or a floating-point value
 1045 (when *baseInstant* is absent).

1046 *Select* term syntax:

```
1047 <Select operation="first|count|exists|total" ? baseInstant="xsd:dateTime" ? >
1048   <Bind namespace="xsd:anyURI" prefix="xsd:NCName" /> *
1049   <Path> xsd:string </Path>
1050 </Sum>
```

1051 Example: The value of this *Select* is the total numeric value of the nodes selected from the con-
 1052 text document; the nodes selected are Price nodes directly within a QoS node.

```
1053 <Select operation="total">
1054   <Path> /*:CandidateQualityProperties/*:Prices/*:Price </Path>
1055 </Select>
```

1056 Example: The value of this *Select* is the numeric value of the first node selected from the context
 1057 document; the node selected is an exact count of CPUs requested in a JSDL job description.

```
1058 <Select operation="first">
1059   <Bind namespace="http://schemas.ggf.org/jsdl/2005/11/jsdl" prefix="jsdl" />
1060   <Path>
1061     //jsdl:JobDescription/jsdl:Resources/jsdl:TotalCPUCount/jsdl:Exact[not(@epsilon)]
1062   </Path>
1063 </Select>
```

1064 7.2.10 Term: *Bound*

1065 The *Bound* term MUST contain one nested term, and MUST permit either or both of the attributes
 1066 “*lowerBound*” and “*upperBound*”. Both the *lowerBound* and the *upperBound* MUST, if present, be
 1067 interpreted as floating-point values. The *Bound* term MUST evaluate to the value that the nested
 1068 term evaluates to unless the *lowerBound* (if present) is a larger value or the *upperBound* (if pre-
 1069 sent) is a smaller value. The *Bound* term MUST evaluate to a failure if the nested term evaluates
 1070 to a failure, if the *lowerBound* (when present) is larger than the value that the nested term evalu-
 1071 ates to, or if the *upperBound* (when present) is smaller than the value that the nested term evalu-
 1072 ates to.

1073 Syntax:

1074 `<Bound lowerBound="xsd:double" ? upperBound="xsd:double" ? > <Term ... /> </Bound>`

1075 Example: The value of this *Bound* is the value of the *Sum* unless that is either less than 1 or more
1076 than 42.

```
1077 <Bound lowerBound="1" upperBound="42">
1078   <Sum>...</Sum>
1079 </Bound>
```

1080 7.2.11 Term: *OneOf*

1081 The *OneOf* term MUST contain at least one nested term. The nested terms MUST be evaluated
1082 in document order, and the value of the *OneOf* term MUST be the value of the first nested term
1083 that does not evaluate to failure; all subsequent terms MAY be ignored. The *OneOf* term MUST
1084 evaluate to failure if and only if all the nested terms evaluate to failure.

1085 Syntax:

```
1086 <OneOf> <Term ... /> + </OneOf>
```

1087 Example: The value of this *OneOf* is the value of the first contained *Bound* that is satisfied.

```
1088 <OneOf>
1089   <Bound ... />
1090   <Bound ... />
1091 </OneOf>
```

1092 Example: The value of this *OneOf* is the value of the *Bound* if that is satisfied, or the value of the
1093 *Constant* otherwise. This never fails (as long as the *Constant* is syntactically correct).

```
1094 <OneOf>
1095   <Bound ... />
1096   <Constant ... />
1097 </OneOf>
```

1098 7.3 Example *OrderFunction*

1099 This example (which should be evaluated within the context of a Candidate Execution Plan) de-
1100 fines the valuation of a candidate to be the total price per CPU (which must be in US dollars per
1101 CPU) which must be kept within acceptable bounds (no more than \$42/CPU). The example in-
1102 cludes extra spaces for greater clarity.

```
1103 <OrderFunction>
1104   <Bound upperBound="42">
1105     <Product>
1106       <Select operation="total">
1107         <!-- Gets a price, assuming we've got an exact one. -->
1108         <Bind namespace="http://schemas.ogf.org/rss/2007/03/csg"
1109           prefix="csg"/>
1110         <Bind namespace="http://schemas.ogf.org/rss/2007/03/eps/basic"
1111           prefix="eps"/>
1112         <Path>
1113           // csg:CandidateQualityProperties
1114           // eps:Price[@currency="currency:USD"]
1115           / eps:PriceRange[@from=@to] @from
1116         </Path>
1117       </Select>
1118     <Power exponent="-1">
1119       <Select operation="first">
1120         <!-- Gets the # of CPUs to be allocated to the job. -->
```

```

1121         <Bind namespace="http://schemas.ogf.org/rss/2007/03/csg"
1122             prefix="csg"/>
1123         <Bind namespace="http://schemas.ogf.org/jsdl/2005/11/jsdl"
1124             prefix="jsdl"/>
1125         <Path>
1126             // csg:CandidateDescription
1127             // jsdl:Resources
1128             / jsdl:TotalCPUCount
1129             / jsdl:Exact
1130         </Path>
1131     </Select>
1132 </Power>
1133 </Product>
1134 </Bound>
1135 </OrderFunction>

```

1136 7.4 Extending SCOL

1137 Uses of SCOL where an extension is used MUST use a different URI to describe the language.
 1138 All extensions MUST be done by introducing new elements. All extension term elements MUST
 1139 be members of the *Term* substitution group, and the type of every extension term element MUST
 1140 be an extension of the *Term* type, and hence must be complex (*i.e.*, element) content. All exten-
 1141 sions MUST also introduce a new top-level element which must be used instead of the *Order-*
 1142 *Function* element.

1143 8. Security Considerations

1144 Since execution plans are intended for use by a specific user, implementations of the EPS
 1145 SHOULD permit the specification of a security context that includes the identity of the final user
 1146 as part of any invocation of the *GetCandidates* operation. They SHOULD (where possible) verify
 1147 that, for each Candidate Execution Plan, the Basic Execution Service referenced by it will accept
 1148 the JSDL *JobDefinition* described by the plan under the identity of the final user. This is particu-
 1149 larly important when the EPS is delegating the generation of some Candidate Execution Plans to
 1150 other EPS instances; in that case, it SHOULD also delegate the users' identities to the inner EPS
 1151 instances.

1152 It is also commonly the case that information systems provide different responses to different us-
 1153 ers. Because of this, it is important for the identity provided within the security context to be used
 1154 when accessing information as part of creating a Candidate Execution Plan.

1155 As there may be user-specific information within a Candidate Execution Plan, implementations
 1156 SHOULD take steps to ensure the confidentiality of the ordered set of plans. This may be by only
 1157 permitting access to the ISS from the same security context that performed the initial invocation
 1158 of the *GetCandidates* operation on the EPS, and ensuring that communications between the cli-
 1159 ent and the EPS and ISS go over a secured channel.

1160 This specification does *not* describe what the form of users' identities look like, how to delegate
 1161 them to other services, or how to communicate a security context to the EPS or the ISS.

1162 9. Contributors

1163 Donal K. Fellows
 1164 University of Manchester
 1165 Oxford Road
 1166 Manchester, M13 9PL
 1167 U. K.
 1168 email: donal.k.fellows@manchester.ac.uk

1169 The author would like to thank the following people (in alphabetical order) for their invaluable as-
 1170 sistance in the design and preparation of this specification: Mathias Dalheimer, Andrew Grim-

1171 shaw, Soonwook Hwang, Stephen Pickles, Andreas Savva, Jay Unger, Philipp Wieder, Ramin
1172 Yahyapour.

1173 **10. Intellectual Property Statement**

1174 The OGF takes no position regarding the validity or scope of any intellectual property or other
1175 rights that might be claimed to pertain to the implementation or use of the technology described in
1176 this document or the extent to which any license under such rights might or might not be avail-
1177 able; neither does it represent that it has made any effort to identify any such rights. Copies of
1178 claims of rights made available for publication and any assurances of licenses to be made avail-
1179 able, or the result of an attempt made to obtain a general license or permission for the use of
1180 such proprietary rights by implementers or users of this specification can be obtained from the
1181 OGF Secretariat.

1182 The OGF invites any interested party to bring to its attention any copyrights, patents or patent
1183 applications, or other proprietary rights which may cover technology that may be required to prac-
1184 tice this recommendation. Please address the information to the OGF Executive Director.

1185 **11. Disclaimer**

1186 This document and the information contained herein is provided on an “As Is” basis and the OGF
1187 disclaims all warranties, express or implied, including but not limited to any warranty that the use
1188 of the information herein will not infringe any rights or any implied warranties of merchantability or
1189 fitness for a particular purpose.

1190 **12. Full Copyright Notice**

1191 Copyright (C) Open Grid Forum (2005-2007). All Rights Reserved.

1192 This document and translations of it may be copied and furnished to others, and derivative works
1193 that comment on or otherwise explain it or assist in its implementation may be prepared, copied,
1194 published and distributed, in whole or in part, without restriction of any kind, provided that the
1195 above copyright notice and this paragraph are included on all such copies and derivative works.
1196 However, this document itself may not be modified in any way, such as by removing the copyright
1197 notice or references to the OGF or other organizations, except as needed for the purpose of de-
1198 veloping Grid Recommendations in which case the procedures for copyrights defined in the OGF
1199 Document process must be followed, or as required to translate it into languages other than Eng-
1200 lish.

1201 The limited permissions granted above are perpetual and will not be revoked by the OGF or its
1202 successors or assignees.

1203 **13. References**

1204 [BES] Foster, I., Grimshaw, A., Lane, P., Lee, W., Morgan, M., Newhouse, S., Pickles, S.,
1205 Pulsipher, D., Smith, C., Theimer, M., *GFD.108: OGSA® Basic Execution Service Ver-*
1206 *sion 1.0*, Open Grid Forum, August 2007.

1207 [Brad] Bradner, S., *RFC 2119: Key Words for Use in RFCs to Indicate Requirement Levels*,
1208 IETF, March 1997.

1209 [EMSS] Savva, A., Kishimoto, H., Newhouse, S., Pulsipher, D., *GFD.106: OGSA® EMS Archi-*
1210 *tecture Scenarios, Version 1.0*, Open Grid Forum. May 2007.

1211 [JSDL] Anjomshoaa, A., Brisard, F., Drescher, M., Fellows, D., Ly, A., McGough, S., Pulsipher,
1212 D., Savva, A., *GFD.56: Job Submission Description Language, v1.0*, Global Grid Fo-
1213 rum. November 2005.

1214 [OWSR] Foster, I., Maguire, T., Snelling, D., *GFD.72: OGSA® WSRF Basic Profile 1.0*, Open
1215 Grid Forum, September 2006.

- 1216 [SOAP] Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J., Nielsen, H., Karmarkar, A., Lafon,
1217 Y., *SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)*, W3C, April
1218 2007.
- 1219 [WSA] Gudgin, M., Hadley, M., Rogers, T., *Web Services Addressing 1.0 – Core*, W3C, May
1220 2006.
- 1221 [WSAG] Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Kakata, T., Pruyne, J.,
1222 Rofrano, J., Tuecke, S., Xu, M., *GFD.107: Web Services Agreement Specification (WS-
1223 Agreement)*, Open Grid Forum, May 2007.
- 1224 [WSBF] Liu, L., Meder, S., *Web Services Base Faults 1.2 (WS-BaseFaults)*, OASIS, April 2006.
- 1225 [WSR] Graham, S., Karmarkar, A., Mischkinsky, J., Robinson, I., Sedukhin, I., *Web Services
1226 Resource 1.2 (WS-Resource)*, OASIS, April 2006.
- 1227 [WSRL] Srinivasan, L., Banks, T., *Web Services Resource Lifetime 1.2 (WS-ResourceLifetime)*,
1228 OASIS, April 2006.
- 1229 [WSRP] Graham, S., Treadwell, J., *Web Services Resource Properties 1.2 (WS-ResourceProp-
1230 erties)*, OASIS, April 2006.
- 1231 [XQ] Boag, S., Chamberlin, D., Fernández, M., Florescu, D., Robie, J., Siméon, J., *XQuery
1232 1.0: An XML Query Language*, W3C, January 2007.
- 1233 [XP] Berglund, A., Boag, S., Chamberlin, D., Fernández, M., Kay, M., Robie, J., Siméon, J.,
1234 *XML Path Language (XPath) 2.0*, W3C, January 2007.
- 1235

1235 **Appendix A: Normative Schemas**1236 **A.1 XML Schema for CSG Core**

```

1237 <?xml version="1.0" encoding="UTF-8"?>
1238 <xsd:schema targetNamespace="http://schemas.ogf.org/rss/2007/03/csg"
1239   xmlns:csg="http://schemas.ogf.org/rss/2007/03/csg"
1240   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
1241   attributeFormDefault="unqualified" elementFormDefault="qualified"
1242   xml:lang="en">
1243
1244   <xsd:annotation>
1245     <xsd:documentation>
1246       This describes the basic elements used in all candidate set
1247       generators.
1248
1249       Copyright © 2007, Open Grid Forum
1250     </xsd:documentation>
1251     <!-- Need to check the copyright and get the full copyright statement in here. -->
1252   </xsd:annotation>
1253
1254   <xsd:element name="RequestCandidates">
1255     <xsd:annotation>
1256       <xsd:documentation>
1257         This describes an entire request for candidate generation. The
1258         input document MUST have a description of the input task (the
1259         thing that the requestor is proposing to do, and which the CSG
1260         should generate candidates for how to carry out) and SHOULD
1261         have a description of what profile is being used, how to order
1262         the set of candidates generated, and a list of all extension
1263         elements that the candidate consumer expects to be able to
1264         deal with. Anything else is completely optional from the point
1265         of view of this specification, SHOULD be described by the
1266         UsageProfile, but MUST result in a fault if the particular CSG
1267         does not understand it.
1268       </xsd:documentation>
1269     </xsd:annotation>
1270     <xsd:complexType>
1271       <xsd:sequence>
1272         <xsd:element ref="csg:UsageProfile" maxOccurs="1"
1273           minOccurs="0" />
1274         <xsd:element ref="csg:InputDescription" />
1275         <xsd:element ref="csg:CandidateOrdering" minOccurs="0" />
1276         <xsd:element ref="csg:PermittedCandidateExtension"
1277           maxOccurs="unbounded" minOccurs="0" />
1278         <xsd:any namespace="##other" minOccurs="0"
1279           processContents="lax" maxOccurs="unbounded" />
1280       </xsd:sequence>
1281     </xsd:complexType>
1282   </xsd:element>
1283
1284   <xsd:element name="UsageProfile" type="xsd:anyURI">
1285     <xsd:annotation>
1286       <xsd:documentation>
1287         This element is used to declare exactly what profile (among
1288         those supported by the CSG) is being used for this particular

```

```

1289     request for candidates.
1290
1291     When used as a property of a CSG, the usage profile declares
1292     what profiles of that CSG (i.e. what input descriptions they
1293     accept and candidates they produce) are actually implemented.
1294     Each particular profile has its own unique "well-known" URI.
1295     The set of profiles supported should be modelled by the
1296     service as some kind of property; the manner in which this is
1297     done is defined by the service rendering. Every CSG must
1298     support at least one usage profile.
1299     </xsd:documentation>
1300   </xsd:annotation>
1301 </xsd:element>
1302
1303 <xsd:element name="InputDescription">
1304   <xsd:annotation>
1305     <xsd:documentation>
1306       This element describes what the action the requestor wants to
1307       carry out, and what they want candidate strategies for
1308       performing.
1309     </xsd:documentation>
1310   </xsd:annotation>
1311   <xsd:complexType mixed="true">
1312     <xsd:sequence>
1313       <xsd:any namespace="##other" processContents="lax"
1314         minOccurs="1" maxOccurs="1" />
1315     </xsd:sequence>
1316     <xsd:anyAttribute namespace="##other" processContents="lax">
1317       <xsd:annotation>
1318         <xsd:documentation>
1319           Any attribute from namespaces other than the current
1320           one MAY be added by the requesting agent, but the CSG
1321           MUST fault when faced with an attribute it does not
1322           understand.
1323         </xsd:documentation>
1324       </xsd:annotation>
1325     </xsd:anyAttribute>
1326   </xsd:complexType>
1327 </xsd:element>
1328
1329 <xsd:element name="CandidateOrdering">
1330   <xsd:annotation>
1331     <xsd:documentation>
1332       This describes how to order (and possibly how to go about
1333       generating) the candidates in response to the request
1334       document. The content of this element is any sequence of XML
1335       elements (this schema does not define any scheme) but the
1336       dialect attribute MUST be given so that agents processing this
1337       element can determine whether they know how to correctly
1338       process the contents of this element at an early stage.
1339     </xsd:documentation>
1340   </xsd:annotation>
1341   <xsd:complexType mixed="true">
1342     <xsd:sequence>
1343       <xsd:any namespace="##other" processContents="lax"
1344         minOccurs="0" maxOccurs="unbounded" />

```

```

1345     </xsd:sequence>
1346     <xsd:attribute name="dialect" type="xsd:anyURI" use="required">
1347         <xsd:annotation>
1348             <xsd:documentation>
1349                 This element describes (via a URI) what the contents
1350                 of the CandidateOrdering element are.
1351             </xsd:documentation>
1352         </xsd:annotation>
1353     </xsd:attribute>
1354     <xsd:anyAttribute namespace="##other" processContents="lax">
1355         <xsd:annotation>
1356             <xsd:documentation>
1357                 Attributes from namespaces other than the current one
1358                 MAY also be added. This schema does not constrain
1359                 them.
1360             </xsd:documentation>
1361         </xsd:annotation>
1362     </xsd:anyAttribute>
1363 </xsd:complexType>
1364 </xsd:element>
1365
1366 <xsd:element name="Candidate">
1367     <xsd:annotation>
1368         <xsd:documentation>
1369             This element is one of the ordered set returned in response to
1370             a request for candidates. It contains three mandatory parts:
1371             the description of what the candidate is, a best-guess
1372             estimate of how good a candidate it is, and the (contiguous)
1373             period of time for which the candidate is valid. The arbitrary
1374             elements after the CandidateValidity element MUST all be
1375             understood by any consumer of the Candidate; it MUST ignore
1376             any Candidate that contains an element as a direct child of
1377             the Candidate element that it does not understand.
1378         </xsd:documentation>
1379     </xsd:annotation>
1380 <xsd:complexType>
1381     <xsd:sequence>
1382         <xsd:element ref="csg:CandidateDescription" />
1383         <xsd:element ref="csg:CandidateQualityProperties" />
1384         <xsd:element ref="csg:CandidateValidity" />
1385         <xsd:any minOccurs="0" maxOccurs="unbounded"
1386             namespace="##other" processContents="lax" />
1387     </xsd:sequence>
1388 </xsd:complexType>
1389 </xsd:element>
1390
1391 <xsd:element name="CandidateDescription">
1392     <xsd:annotation>
1393         <xsd:documentation>
1394             This states what the candidate is. It should describe how to
1395             carry out the action described in the InputDescription. It MAY
1396             contain a copy of part or all of the InputDescription; that
1397             depends on the profile. Consumers of the Candidate MUST
1398             understand the contents of this element.
1399         </xsd:documentation>
1400     </xsd:annotation>

```

```

1401     <xsd:complexType>
1402     <xsd:sequence>
1403         <xsd:any minOccurs="1" maxOccurs="unbounded"
1404             namespace="##other" processContents="lax" />
1405     </xsd:sequence>
1406     <xsd:anyAttribute namespace="##other" processContents="lax" />
1407 </xsd:complexType>
1408 </xsd:element>
1409
1410 <xsd:element name="CandidateQualityProperties">
1411     <xsd:annotation>
1412         <xsd:documentation>
1413             This describes &quot;how good&quot; a candidate we've got.
1414             Consumers of a Candidate MAY ignore anything inside this
1415             element; none of it needs to be understood.
1416         </xsd:documentation>
1417     </xsd:annotation>
1418     <xsd:complexType>
1419         <xsd:sequence>
1420             <xsd:any maxOccurs="unbounded" minOccurs="0"
1421                 namespace="##other" processContents="lax" />
1422         </xsd:sequence>
1423     </xsd:complexType>
1424 </xsd:element>
1425
1426 <xsd:element name="CandidateValidity">
1427     <xsd:annotation>
1428         <xsd:documentation>
1429             This element describes the contiguous period of time for which
1430             the containing Candidate is valid. The CSG need not warrant
1431             anything at all about what happens if a Candidate is actioned
1432             (in whatever way makes sense according to the profile) outside
1433             the validity period; the CandidateQualityProperties might be
1434             totally meaningful then, the Candidate might be impossible to
1435             action, or anything else in the spectrum of things between
1436             those two extremes. If the consumer of the Candidate tries to
1437             action the Candidate within the validity period, it should be
1438             at least possible (within the bounds of best-effort semantics)
1439             to action it, and in that case there should be a reasonable
1440             expectation that the quality described in the
1441             CandidateQualityProperties be achieved. If non-contiguous
1442             period of validity is desired (or if the description or
1443             quality should vary over time) then multiple Candidates should
1444             be generated. This element MUST be empty.
1445         </xsd:documentation>
1446     </xsd:annotation>
1447     <xsd:complexType>
1448         <xsd:sequence maxOccurs="0" minOccurs="0" />
1449         <xsd:attribute name="notBefore" type="xsd:dateTime"
1450             use="required">
1451             <xsd:annotation>
1452                 <xsd:documentation>
1453                     This describes the first instant of the validity
1454                     period.
1455                 </xsd:documentation>
1456             </xsd:annotation>

```



```

1457     </xsd:attribute>
1458     <xsd:attribute name="notAfter" type="xsd:dateTime"
1459         use="required">
1460         <xsd:annotation>
1461             <xsd:documentation>
1462                 This describes the last instant of the validity
1463                 period.
1464             </xsd:documentation>
1465         </xsd:annotation>
1466     </xsd:attribute>
1467 </xsd:complexType>
1468 </xsd:element>
1469
1470 <xsd:element name="PermittedCandidateExtension">
1471     <xsd:annotation>
1472         <xsd:documentation>
1473             This element is used to describe which extension elements may
1474             be added to a candidate by the CSG: the CSG SHOULD NOT return
1475             any candidate that contains an extension element that is not
1476             described by a PermittedCandidateExtension element in the
1477             request.
1478         </xsd:documentation>
1479     </xsd:annotation>
1480     <xsd:complexType>
1481         <xsd:attribute name="type" type="csg:ExtensionType" use="optional"
1482             default="functional" />
1483     </xsd:complexType>
1484 </xsd:element>
1485
1486 <xsd:simpleType name="ExtensionType">
1487     <xsd:annotation>
1488         <xsd:documentation>
1489             Used to describe the type of an extension element. Functional
1490             extension elements are direct children of the Candidate
1491             element and MUST be understood by the consuming system.
1492             Quality extension elements are children of the
1493             CandidateQualityProperties element, and MAY be ignored by the
1494             consuming system.
1495         </xsd:documentation>
1496     </xsd:annotation>
1497     <xsd:restriction base="xsd:string">
1498         <xsd:enumeration value="functional">
1499             <xsd:annotation>
1500                 <xsd:documentation>
1501                     Describes an element that must be understood if
1502                     present in a Candidate, and which is present as a
1503                     direct child of the Candidate element.
1504                 </xsd:documentation>
1505             </xsd:annotation>
1506         </xsd:enumeration>
1507         <xsd:enumeration value="quality">
1508             <xsd:annotation>
1509                 <xsd:documentation>
1510                     Describes an element that may be ignored if present in
1511                     a Candidate, and which may only be present in a
1512                     Candidate as a child of the CandidateQualityProperties

```

```

1513         element.
1514         </xsd:documentation>
1515         </xsd:annotation>
1516         </xsd:enumeration>
1517         </xsd:restriction>
1518         </xsd:simpleType>
1519
1520         <!-- This, along with UsageProfile, are properties common to all Candidate Set Generators -->
1521
1522         <xsd:element name="SupportedOrderingLanguages" type="xsd:anyURI">
1523             <xsd:annotation>
1524                 <xsd:documentation>
1525                     The supported ordering languages gives a scheme for allowing
1526                     implementations of a CSG to declare what they accept in the
1527                     CandidateOrdering element. Each particular language has its
1528                     own unique "well-known" URI. The set of languages supported
1529                     should be modelled by the service as some kind of property;
1530                     the manner in which this is done is defined by the service
1531                     rendering. Every CSG must support at least one ordering
1532                     language.
1533                 </xsd:documentation>
1534             </xsd:annotation>
1535         </xsd:element>
1536
1537 </xsd:schema>

```

1538 A.2 XML Schema for EPS Core

```

1539 <?xml version="1.0" encoding="UTF-8"?>
1540 <xsd:schema targetNamespace="http://schemas.ogf.org/rss/2007/03/eps/basic"
1541     xmlns:eps="http://schemas.ogf.org/rss/2007/03/eps/basic"
1542     xmlns:jsdl="http://schemas.ggf.org/jsdl/2005/11/jsdl"
1543     xmlns:wsa="http://www.w3.org/2005/03/addressing"
1544     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
1545     attributeFormDefault="unqualified" elementFormDefault="qualified"
1546     xml:lang="en">
1547
1548     <xsd:annotation>
1549         <xsd:documentation>
1550             This describes the elements in candidate set generators that are
1551             also execution planning service interfaces.
1552
1553             Copyright © 2007, Open Grid Forum
1554         </xsd:documentation>
1555         <!-- Need to check the copyright and get the full copyright statement in here. -->
1556     </xsd:annotation>
1557
1558     <xsd:import namespace="http://www.w3.org/2005/03/addressing"
1559         schemaLocation="http://www.w3.org/2005/03/addressing" />
1560     <xsd:import namespace="http://schemas.ggf.org/jsdl/2005/11/jsdl"
1561         schemaLocation="http://schemas.ggf.org/jsdl/2005/11/jsdl" />
1562
1563     <xsd:element name="BESReference" type="wsa:EndpointReferenceType">
1564         <xsd:annotation>
1565             <xsd:documentation>
1566                 This element describes a reference to a basic execution

```

```

1567     service that the JSDL document (in the same candidate) should
1568     be submitted to in order to enact the candidate.
1569     </xsd:documentation>
1570   </xsd:annotation>
1571 </xsd:element>
1572 <xsd:complexType name="EPSCandidateDescriptionType">
1573   <xsd:annotation>
1574     <xsd:documentation>
1575       csg:CandidateDescription elements produced by the EPS MUST
1576       match this type.
1577     </xsd:documentation>
1578   </xsd:annotation>
1579   <xsd:sequence>
1580     <xsd:element ref="jSDL:JobDefinition" />
1581     <xsd:element ref="eps:BESReference" />
1582   </xsd:sequence>
1583 </xsd:complexType>
1584 <xsd:element name="Prices">
1585   <xsd:annotation>
1586     <xsd:documentation>
1587       The Prices is used to describe the amount charged for a job
1588       execution carried out by a BES. It describes a sum of
1589       charges within one or more currencies; note that where a
1590       charge is made in multiple currencies, it is in general
1591       non-trivial to convert into a charge in a single currency as
1592       it depends on many factors outside the reasonable control of
1593       any planning service. Sources of such problems may include the
1594       degree of currency uncertainty (as moderated by the clients'
1595       currency hedges!) and the fact that with some virtual
1596       currencies the actual rate may be not determined at all until
1597       substantially after the charge was made (it might depend on
1598       the overall usage of a particular resource over some period of
1599       time). As such, this specification states that such
1600       conversions should not normally be done.
1601     </xsd:documentation>
1602   </xsd:annotation>
1603   <xsd:complexType>
1604     <xsd:sequence>
1605       <xsd:element ref="eps:Price" maxOccurs="unbounded"
1606         minOccurs="1" />
1607     </xsd:sequence>
1608   </xsd:complexType>
1609 </xsd:element>
1610 <xsd:element name="Price">
1611   <xsd:annotation>
1612     <xsd:documentation>
1613       The Price element describes a price for resources and services
1614       calculated using a single currency, which may be a virtual or
1615       real currency. Real currencies MUST be denoted using the
1616       pseudo-URI &quot;currency:TLC&quot; where TLC is the ISO 4217
1617       three-letter code for that particular currency. The Price
1618       element must contain one or more PriceRange elements, each of
1619       which describes a range of prices that may be charged; the use
1620       of multiple ranges allows for finer estimates and
1621       trace-ability of billing than a single range. The overall
1622       price within a given currency is given by the sum of the

```

```

1623         values selected from within each of the contained ranges.
1624     </xsd:documentation>
1625 </xsd:annotation>
1626 <xsd:complexType>
1627     <xsd:sequence>
1628         <xsd:element ref="eps:PriceRange" minOccurs="1"
1629             maxOccurs="unbounded" />
1630     </xsd:sequence>
1631     <xsd:attribute name="currency" type="xsd:anyURI" use="required" />
1632 </xsd:complexType>
1633 </xsd:element>
1634 <xsd:element name="PriceRange">
1635     <xsd:annotation>
1636         <xsd:documentation>
1637             The PriceRange element describes an individual contiguous
1638             range of possible prices. The actual price charged (due to the
1639             cause described by the element or string content) SHOULD be a
1640             value in the inclusive range defined by the from and to
1641             attributes; the range is a singleton if the from value is the
1642             same as the to value. The from value MUST NOT be greater than
1643             the to value. A reverse direction charge (where the client of
1644             the EPS is paid for their custom) MUST be expressed using a
1645             PriceRange whose from and to attributes are negative.
1646         </xsd:documentation>
1647     </xsd:annotation>
1648     <xsd:complexType mixed="true">
1649         <xsd:sequence maxOccurs="1" minOccurs="1">
1650             <xsd:any namespace="##other" processContents="lax"
1651                 minOccurs="0" maxOccurs="unbounded" />
1652         </xsd:sequence>
1653         <xsd:attribute name="from" type="xsd:double" use="required" />
1654         <xsd:attribute name="to" type="xsd:double" use="required" />
1655     </xsd:complexType>
1656 </xsd:element>
1657 <xsd:complexType name="DelaySpanType">
1658     <xsd:annotation>
1659         <xsd:documentation>
1660             This type describes a contiguous range of durations, in
1661             seconds. Elements of this type MUST NOT have any element or
1662             string content.
1663         </xsd:documentation>
1664     </xsd:annotation>
1665     <xsd:sequence maxOccurs="0" minOccurs="0" />
1666     <xsd:attribute name="from" type="xsd:double" />
1667     <xsd:attribute name="to" type="xsd:double" />
1668 </xsd:complexType>
1669 <xsd:complexType name="TimeSpanType">
1670     <xsd:annotation>
1671         <xsd:documentation>
1672             This type describes a contiguous period of time. Elements of
1673             this type MUST NOT have any element or string content.
1674         </xsd:documentation>
1675     </xsd:annotation>
1676     <xsd:sequence minOccurs="0" maxOccurs="0" />
1677     <xsd:attribute name="from" type="xsd:dateTime" />
1678     <xsd:attribute name="to" type="xsd:dateTime" />

```

```

1679 </xsd:complexType>
1680 <xsd:element name="StartTimes">
1681   <xsd:annotation>
1682     <xsd:documentation>
1683       This element describes the periods of time when the job may
1684       start executing. The start times are given as a
1685       possibly-discontinuous set of times.
1686     </xsd:documentation>
1687   </xsd:annotation>
1688 </xsd:complexType>
1689 <xsd:sequence>
1690   <xsd:element ref="eps:StartAt" minOccurs="1"
1691     maxOccurs="unbounded" />
1692 </xsd:sequence>
1693 </xsd:complexType>
1694 </xsd:element>
1695 <xsd:element name="StartAt" type="eps:TimeSpanType">
1696   <xsd:annotation>
1697     <xsd:documentation>
1698       This element describes a contiguous period of time during
1699       which the execution of the job may commence. Both the from and
1700       the to attribute MUST be present. The from attribute MUST NOT
1701       specify an instant before the to attribute.
1702     </xsd:documentation>
1703   </xsd:annotation>
1704 </xsd:element>
1705 <xsd:element name="StartDelays">
1706   <xsd:annotation>
1707     <xsd:documentation>
1708       This element describes the delay between when the job is
1709       submitted to the BES and when the job may start executing. The
1710       delay is given as a possibly-discontinuous set of durations.
1711     </xsd:documentation>
1712   </xsd:annotation>
1713 </xsd:complexType>
1714 <xsd:sequence>
1715   <xsd:element ref="eps:DelayRange" maxOccurs="unbounded"
1716     minOccurs="1" />
1717 </xsd:sequence>
1718 </xsd:complexType>
1719 </xsd:element>
1720 <xsd:element name="DelayRange" type="eps:DelaySpanType">
1721   <xsd:annotation>
1722     <xsd:documentation>
1723       This element describes a contiguous range of delays, in
1724       seconds. Two attributes, from and to, MUST be present. The
1725       value of the from attribute MUST NOT be greater than the value
1726       of the to attribute.
1727     </xsd:documentation>
1728   </xsd:annotation>
1729 </xsd:element>
1730 <xsd:element name="EndTimes">
1731   <xsd:complexType>
1732     <xsd:sequence>
1733       <xsd:element ref="eps:EndAt" maxOccurs="unbounded"
1734         minOccurs="1" />

```

```

1735     </xsd:sequence>
1736   </xsd:complexType>
1737 </xsd:element>
1738 <xsd:element name="EndAt" type="eps:TimeSpanType">
1739   <xsd:annotation>
1740     <xsd:documentation>
1741       This element describes a contiguous period of time during
1742       which the execution of the job may finish. Both the from and
1743       the to attribute MUST be present. The from attribute MUST NOT
1744       specify an instant before the to attribute.
1745     </xsd:documentation>
1746   </xsd:annotation>
1747 </xsd:element>
1748 <xsd:element name="ExecutionDuration">
1749   <xsd:annotation>
1750     <xsd:documentation>
1751       This element describes the amount of (wall-clock) time for
1752       which the job will execute. The run-time is given as a
1753       possibly-discontinuous set of durations.
1754     </xsd:documentation>
1755   </xsd:annotation>
1756   <xsd:complexType>
1757     <xsd:sequence>
1758       <xsd:element ref="eps:RuntimeRange" maxOccurs="unbounded"
1759         minOccurs="1" />
1760     </xsd:sequence>
1761   </xsd:complexType>
1762 </xsd:element>
1763 <xsd:element name="RuntimeRange" type="eps:DelaySpanType">
1764   <xsd:annotation>
1765     <xsd:documentation>
1766       This element describes a contiguous range of runtimes, in
1767       seconds. Two attributes, from and to, MUST be present. The
1768       value of the from attribute MUST NOT be greater than the value
1769       of the to attribute.
1770     </xsd:documentation>
1771   </xsd:annotation>
1772 </xsd:element>
1773 <xsd:complexType name="EPSCandidateQualityPropertiesType">
1774   <xsd:annotation>
1775     <xsd:documentation>
1776       csg:CandidateQualityProperties elements produced by the EPS
1777       MUST match or extend this type.
1778     </xsd:documentation>
1779   </xsd:annotation>
1780   <xsd:sequence>
1781     <xsd:element ref="eps:Prices" maxOccurs="1" minOccurs="0" />
1782     <xsd:element ref="eps:StartTimes" maxOccurs="1" minOccurs="0" />
1783     <xsd:element ref="eps:StartDelays" maxOccurs="1" minOccurs="0" />
1784     <xsd:element ref="eps:EndTimes" maxOccurs="1" minOccurs="0" />
1785     <xsd:element ref="eps:ExecutionDuration" maxOccurs="1"
1786       minOccurs="0" />
1787   </xsd:sequence>
1788 </xsd:complexType>
1789 </xsd:schema>
1790

```

1791 A.3 XML Schema for Basic-EPS-RF

```

1792 <?xml version="1.0" encoding="UTF-8"?>
1793 <xsd:schema targetNamespace="http://schemas.ogf.org/rss/2007/03/eps/wsrf"
1794   xmlns:csg="http://schemas.ogf.org/rss/2007/03/csg"
1795   xmlns:eps-basic="http://schemas.ogf.org/rss/2007/03/eps/basic"
1796   xmlns:eps-wsrf="http://schemas.ogf.org/rss/2007/03/eps/wsrf"
1797   xmlns:ogsa-bp="http://schemas.ggf.org/ogsa/2006/05/wsrf-bp"
1798   xmlns:ws-a="http://www.w3.org/2005/03/addressing"
1799   xmlns:wsrf-bf="http://docs.oasis-open.org/wsrf/bf-1"
1800   xmlns:wsrf-rl="http://docs.oasis-open.org/wsrf/rl-1"
1801   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
1802   attributeFormDefault="unqualified" elementFormDefault="qualified"
1803   xml:lang="en">
1804
1805   <xsd:annotation>
1806     <xsd:documentation>
1807       This schema describes the elements that are only used when
1808       implementing an EPS on top of WS-RF.
1809
1810       Copyright © 2007, Open Grid Forum
1811     </xsd:documentation>
1812     <!-- Need to check the copyright and get the full copyright statement in here. -->
1813   </xsd:annotation>
1814
1815   <xsd:import namespace="http://schemas.ogf.org/rss/2007/03/csg"
1816     schemaLocation="csg.xsd" />
1817   <xsd:import namespace="http://schemas.ogf.org/rss/2007/03/eps/basic"
1818     schemaLocation="eps.xsd" />
1819   <xsd:import namespace="http://docs.oasis-open.org/wsrf/bf-1"
1820     schemaLocation="http://docs.oasis-open.org/wsrf/bf-1" />
1821   <xsd:import namespace="http://www.w3.org/2005/03/addressing"
1822     schemaLocation="http://www.w3.org/2005/03/addressing" />
1823   <xsd:import namespace="http://docs.oasis-open.org/wsrf/rl-1"
1824     schemaLocation="http://docs.oasis-open.org/wsrf/rl-1" />
1825   <xsd:import namespace="http://schemas.ggf.org/ogsa/2006/05/wsrf-bp"
1826     schemaLocation="http://schemas.ggf.org/ogsa/2006/05/wsrf-bp.xsd" />
1827
1828   <xsd:element name="GetCandidatesRequest">
1829     <xsd:annotation>
1830       <xsd:documentation>
1831         This element describes the content of the request part of the
1832         GetCandidates operation of an EPS when implemented using
1833         WS-RF.
1834       </xsd:documentation>
1835     </xsd:annotation>
1836     <xsd:complexType>
1837       <xsd:sequence>
1838         <xsd:element ref="csg:RequestCandidates" />
1839         <xsd:element ref="eps-wsrf:MaxCandidatesPerTransfer"
1840           maxOccurs="1" minOccurs="0" />
1841         <xsd:element ref="eps-wsrf:MaxCandidatesTotal" maxOccurs="1"
1842           minOccurs="0" />
1843         <xsd:element ref="eps-wsrf:MaxResponseOctets" maxOccurs="1"
1844           minOccurs="0">
1845       </xsd:sequence>

```

```
1846     <xsd:element ref="eps-wsrf:IterableSetLifetime" maxOccurs="1"
1847         minOccurs="0" />
1848     <xsd:any namespace="##other" processContents="lax"
1849         minOccurs="0" maxOccurs="unbounded" />
1850     </xsd:sequence>
1851 </xsd:complexType>
1852 </xsd:element>
1853
1854 <xsd:element name="GetCandidatesResponse">
1855     <xsd:annotation>
1856         <xsd:documentation>
1857             The response of the GetCandidates operation is an optional
1858             sequence of candidates (satisfying the general cardinality
1859             rules of the service and the requestor) followed by an
1860             optional reference to a resource which will provide the
1861             remaining candidates. If there is no reference to an Iterable
1862             Set Service, the sequence of candidates in this response
1863             message must be all the candidates that the EPS is prepared to
1864             give the client. Note that if a reference to an ISS is
1865             present, no conclusions about the total number of remaining
1866             candidates may be drawn.
1867         </xsd:documentation>
1868     </xsd:annotation>
1869     <xsd:complexType>
1870         <xsd:sequence>
1871             <xsd:element ref="csg:Candidate" maxOccurs="unbounded"
1872                 minOccurs="0" />
1873             <xsd:element ref="eps-wsrf:IterableSetReference" maxOccurs="1"
1874                 minOccurs="0" />
1875         </xsd:sequence>
1876     </xsd:complexType>
1877 </xsd:element>
1878
1879 <xsd:element name="UnsupportedCSGProfileFault"
1880     type="wsrf-bf:BaseFaultType">
1881     <xsd:annotation>
1882         <xsd:documentation>
1883             This fault is thrown when the request to the GetCandidates
1884             operation does not match any profile that is supported by the
1885             EPS.
1886         </xsd:documentation>
1887     </xsd:annotation>
1888 </xsd:element>
1889
1890 <xsd:element name="UnsupportedOrderingLanguageFault"
1891     type="wsrf-bf:BaseFaultType">
1892     <xsd:annotation>
1893         <xsd:documentation>
1894             This fault is thrown when the ordering component of the
1895             request to the GetCandidates operation is not in a language
1896             supported by the EPS.
1897         </xsd:documentation>
1898     </xsd:annotation>
1899 </xsd:element>
1900
1901 <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

```

```

1958         using the interfaces given by WS-Lifetime.
1959         </xsd:documentation>
1960     </xsd:annotation>
1961 </xsd:element>
1962
1963 <xsd:element name="IterableSetReference"
1964     type="ws-a:EndpointReferenceType">
1965     <xsd:annotation>
1966         <xsd:documentation>
1967             This element contains a reference to an Iterable Set Service
1968             resource containing additional candidates generated in
1969             response to a GetCandidates request.
1970         </xsd:documentation>
1971     </xsd:annotation>
1972 </xsd:element>
1973
1974 <xsd:element name="GetMoreCandidatesRequest">
1975     <xsd:annotation>
1976         <xsd:documentation>
1977             This is the content of the request to the GetMoreCandidates
1978             operation of the Iterable Set Service. It contains an optional
1979             Count element, but is otherwise empty.
1980         </xsd:documentation>
1981     </xsd:annotation>
1982     <xsd:complexType>
1983         <xsd:sequence>
1984             <xsd:element ref="eps-wsrf:Count" maxOccurs="1" minOccurs="0" />
1985         </xsd:sequence>
1986     </xsd:complexType>
1987 </xsd:element>
1988
1989 <xsd:element name="GetMoreCandidatesResponse">
1990     <xsd:annotation>
1991         <xsd:documentation>
1992             This is the content of the response to the GetMoreCandidates
1993             operation of the IterableSetService. It contains zero or more
1994             candidates; the response SHOULD only contain zero candidates
1995             when the list of candidates (internal to the ISS) is
1996             exhausted.
1997         </xsd:documentation>
1998     </xsd:annotation>
1999     <xsd:complexType>
2000         <xsd:sequence>
2001             <xsd:element ref="csg:Candidate" maxOccurs="unbounded"
2002             minOccurs="0" />
2003         </xsd:sequence>
2004     </xsd:complexType>
2005 </xsd:element>
2006
2007 <xsd:element name="Count" type="xsd:positiveInteger">
2008     <xsd:annotation>
2009         <xsd:documentation>
2010             This element describes how many candidates the client wants to
2011             retrieve from the ISS this time.
2012         </xsd:documentation>
2013     </xsd:annotation>

```

```

2014 </xsd:element>
2015
2016 <xsd:element name="EPSProperties">
2017   <xsd:annotation>
2018     <xsd:documentation>
2019       This describes the resource properties document for the
2020       Execution Planning Service.
2021     </xsd:documentation>
2022   </xsd:annotation>
2023   <xsd:complexType>
2024     <xsd:sequence>
2025       <xsd:element ref="ogsa-bp:ResourcePropertyNames" />
2026       <xsd:element ref="ogsa-bp:FinalWSResourceInterface" />
2027       <xsd:element ref="ogsa-bp:WSResourceInterfaces" />
2028       <xsd:element ref="ogsa-bp:ResourceEndpointReference" />
2029       <xsd:element ref="csg:UsageProfile" maxOccurs="unbounded"
2030         minOccurs="1" />
2031       <xsd:element ref="csg:SupportedOrderingLanguages"
2032         maxOccurs="unbounded" minOccurs="1" />
2033       <xsd:element ref="eps-wsrf:MaxCandidatesPerTransfer"
2034         minOccurs="1" maxOccurs="1" />
2035       <xsd:element ref="eps-wsrf:MaxResponseOctets"></xsd:element>
2036       <xsd:element ref="eps-wsrf:DefaultIterableSetLifetime" />
2037       <xsd:any namespace="##other" minOccurs="0"
2038         maxOccurs="unbounded" />
2039     </xsd:sequence>
2040   </xsd:complexType>
2041 </xsd:element>
2042
2043 <xsd:element name="ISSProperties">
2044   <xsd:annotation>
2045     <xsd:documentation>
2046       This describes the resource properties document for the
2047       Iterable Set Service.
2048     </xsd:documentation>
2049   </xsd:annotation>
2050   <xsd:complexType>
2051     <xsd:sequence>
2052       <xsd:element ref="ogsa-bp:ResourcePropertyNames" />
2053       <xsd:element ref="ogsa-bp:FinalWSResourceInterface" />
2054       <xsd:element ref="ogsa-bp:WSResourceInterfaces" />
2055       <xsd:element ref="ogsa-bp:ResourceEndpointReference" />
2056       <xsd:element ref="wsrf-rl:CurrentTime" maxOccurs="1"
2057         minOccurs="1" />
2058       <xsd:element ref="wsrf-rl:TerminationTime" maxOccurs="1"
2059         minOccurs="1" />
2060       <xsd:element ref="eps-wsrf:MaxCandidatesPerTransfer" />
2061       <xsd:element ref="eps-wsrf:MaxResponseOctets" maxOccurs="1"
2062         minOccurs="0">
2063     </xsd:element>
2064     <xsd:element ref="eps-wsrf:CandidatesTransferredSoFar"
2065       maxOccurs="1" minOccurs="0" />
2066     <xsd:element ref="eps-wsrf:AnyMoreCandidatesToTransfer"
2067       maxOccurs="1" minOccurs="0" />
2068   </xsd:sequence>
2069 </xsd:complexType>

```

```

2070 </xsd:element>
2071
2072 <xsd:element name="DefaultIterableSetLifetime" type="xsd:duration">
2073   <xsd:annotation>
2074     <xsd:documentation>
2075       This describes the default lifetime of Iterable Set Service
2076       resources created by the Execution Planning Service.
2077     </xsd:documentation>
2078   </xsd:annotation>
2079 </xsd:element>
2080
2081 <xsd:element name="CandidatesTransferredSoFar"
2082   type="xsd:nonNegativeInteger">
2083   <xsd:annotation>
2084     <xsd:documentation>
2085       This describes the number of candidates that have already been
2086       transferred from the (conceptual) underlying ordered set,
2087       including any returned in the result of the initial call to
2088       GetCandidates.
2089     </xsd:documentation>
2090   </xsd:annotation>
2091 </xsd:element>
2092
2093 <xsd:element name="AnyMoreCandidatesToTransfer" type="xsd:boolean">
2094   <xsd:annotation>
2095     <xsd:documentation>
2096       This describes whether there are any more candidates in the
2097       underlying (conceptual) ordered set. Support for it is not
2098       required, but if it is present then no candidates should ever
2099       be returned after the value of this property becomes true.
2100       Implementations may choose to release internal resources at
2101       the same time as setting this property to true.
2102     </xsd:documentation>
2103   </xsd:annotation>
2104 </xsd:element>
2105
2106 </xsd:schema>

```

2107 A.4 WSDL for Basic-EPS-RF

```

2108 <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2109 <wsdl:definitions name="eps-wsrf" xml:lang="en"
2110   targetNamespace="http://schemas.ogf.org/rss/2007/03/eps/wsrf"
2111   xmlns:eps-wsrf="http://schemas.ogf.org/rss/2007/03/eps/wsrf"
2112   xmlns:rw="http://docs.oasis-open.org/wsrf/rw-1"
2113   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
2114   xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
2115   xmlns:wSDL="http://schemas.xmlsoap.org/wsdl/"
2116   xmlns:wSrf-rp="http://docs.oasis-open.org/wsrf/rp-1"
2117   xmlns:wSrf-rlw="http://docs.oasis-open.org/wsrf/rlw-1"
2118   xmlns:wSrf-rpw="http://docs.oasis-open.org/wsrf/rpw-1"
2119   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
2120
2121   <wsdl:documentation>
2122     This describes the details of the rendering of the Execution Planning
2123     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>

```

```

2180 <wsdl:message name="UnsupportedOrderingLanguageFault">
2181   <wsdl:documentation>
2182     Fault thrown if the EPS cannot handle the submitted candidate
2183     ordering language.
2184   </wsdl:documentation>
2185   <wsdl:part name="UnsupportedOrderingLanguageFault"
2186     element="eps-wsrf:UnsupportedOrderingLanguageFault" />
2187 </wsdl:message>
2188 <wsdl:message name="BadCandidateCardinalityRulesFault">
2189   <wsdl:documentation>
2190     Fault thrown if an attempt to violate the rules on plan
2191     cardinality is made.
2192   </wsdl:documentation>
2193   <wsdl:part name="BadCandidateCardinalityRulesFault"
2194     element="eps-wsrf:BadCandidateCardinalityRulesFault" />
2195 </wsdl:message>
2196 <wsdl:message name="UnableToGenerateCandidatesFault">
2197   <wsdl:documentation>
2198     Fault thrown if it is impossible to generate plans for some other
2199     reason not covered by the other faults.
2200   </wsdl:documentation>
2201   <wsdl:part name="UnableToGenerateCandidatesFault"
2202     element="eps-wsrf:UnableToGenerateCandidatesFault" />
2203 </wsdl:message>
2204 <wsdl:message name="UnsupportedCSGProfileFault">
2205   <wsdl:documentation>
2206     Fault thrown if the request to the EPS isn't one that is
2207     understood properly.
2208   </wsdl:documentation>
2209   <wsdl:part name="UnsupportedCSGProfileFault"
2210     element="eps-wsrf:UnsupportedCSGProfileFault" />
2211 </wsdl:message>
2212
2213 <wsdl:portType name="ExecutionPlanningService"
2214   wsrf-rp:ResourceProperties="eps-wsrf:EPSProperties">
2215   <wsdl:documentation>
2216     The execution planner port-type describes the real core of the EPS
2217     specification, as rendered using WSRF.
2218   </wsdl:documentation>
2219   <wsdl:operation name="GetCandidates">
2220     <wsdl:input message="eps-wsrf:GetCandidates"
2221       wsdl:Action="http://schemas.ogf.org/rss/2007/03/eps/wsrf/GetCandidates" />
2222     <wsdl:output message="eps-wsrf:GetCandidatesResult"
2223       wsdl:Action="http://schemas.ogf.org/rss/2007/03/eps/wsrf/GetCandidatesResponse" />
2224     <wsdl:fault name="UnsupportedOrderingLanguageFault"
2225       message="eps-wsrf:UnsupportedOrderingLanguageFault" />
2226     <wsdl:fault name="BadCandidateCardinalityRulesFault"
2227       message="eps-wsrf:BadCandidateCardinalityRulesFault" />
2228     <wsdl:fault name="UnableToGenerateCandidatesFault"
2229       message="eps-wsrf:UnableToGenerateCandidatesFault" />
2230     <wsdl:fault name="ResourceUnknownFault"
2231       message="rw:ResourceUnknownFault" />
2232     <wsdl:fault name="UnsupportedCSGProfileFault"
2233       message="eps-wsrf:UnsupportedCSGProfileFault" />
2234   </wsdl:operation>
2235   <wsdl:operation name="GetResourcePropertyDocument">

```

```

2236     <wsdl:input name="GetResourcePropertyDocumentRequest"
2237         message="wsrf-rpw:GetResourcePropertyDocumentRequest" />
2238     <wsdl:output name="GetResourcePropertyDocumentResponse"
2239         message="wsrf-rpw:GetResourcePropertyDocumentResponse" />
2240     <wsdl:fault name="ResourceUnknownFault"
2241         message="rw:ResourceUnknownFault" />
2242 </wsdl:operation>
2243 </wsdl:portType>
2244 <wsdl:portType name="IterableSetService"
2245     wsrf-rp:ResourceProperties="eps-wsrf:ISSProperties">
2246     <wsdl:documentation>
2247         The execution planner result port-type is used to implement
2248         partial transfers of results. Note that in the WSRF rendering, the
2249         ISS must implement the WS-ResourceLifetime operations too.
2250     </wsdl:documentation>
2251     <wsdl:operation name="GetMoreCandidates">
2252         <wsdl:input message="eps-wsrf:GetMoreCandidates"
2253             wsa:Action="http://schemas.ogf.org/rss/2007/03/eps/wsrf/GetMoreCandidates" />
2254         <wsdl:output message="eps-wsrf:GetMoreCandidatesResult"
2255             wsa:Action="http://schemas.ogf.org/rss/2007/03/eps/wsrf/GetMoreCandidatesResponse" />
2256         <wsdl:fault name="BadCandidateCardinalityRulesFault"
2257             message="eps-wsrf:BadCandidateCardinalityRulesFault" />
2258         <wsdl:fault name="UnableToGenerateCandidatesFault"
2259             message="eps-wsrf:UnableToGenerateCandidatesFault" />
2260         <wsdl:fault name="ResourceUnknownFault"
2261             message="rw:ResourceUnknownFault" />
2262     </wsdl:operation>
2263     <wsdl:operation name="Destroy">
2264         <wsdl:input name="DestroyRequest"
2265             message="wsrf-rlw:DestroyRequest" />
2266         <wsdl:output name="DestroyResponse"
2267             message="wsrf-rlw:DestroyResponse" />
2268         <wsdl:fault name="ResourceNotDestroyedFault"
2269             message="wsrf-rlw:ResourceNotDestroyedFault" />
2270         <wsdl:fault name="ResourceUnknownFault"
2271             message="rw:ResourceUnknownFault" />
2272     </wsdl:operation>
2273     <wsdl:operation name="SetTerminationTime">
2274         <wsdl:input name="SetTerminationTimeRequest"
2275             message="wsrf-rlw:SetTerminationTimeRequest" />
2276         <wsdl:output name="SetTerminationTimeResponse"
2277             message="wsrf-rlw:SetTerminationTimeResponse" />
2278         <wsdl:fault name="UnableToSetTerminationTimeFault"
2279             message="wsrf-rlw:UnableToSetTerminationTimeFault" />
2280         <wsdl:fault name="ResourceUnknownFault"
2281             message="rw:ResourceUnknownFault" />
2282         <wsdl:fault name="TerminationTimeChangeRejectedFault"
2283             message="wsrf-rlw:TerminationTimeChangeRejectedFault" />
2284     </wsdl:operation>
2285     <wsdl:operation name="GetResourcePropertyDocument">
2286         <wsdl:input name="GetResourcePropertyDocumentRequest"
2287             message="wsrf-rpw:GetResourcePropertyDocumentRequest" />
2288         <wsdl:output name="GetResourcePropertyDocumentResponse"
2289             message="wsrf-rpw:GetResourcePropertyDocumentResponse" />
2290         <wsdl:fault name="ResourceUnknownFault"
2291             message="rw:ResourceUnknownFault" />

```

```

2292     </wsdl:operation>
2293 </wsdl:portType>
2294
2295 <wsdl:binding name="eps-wsrf-SOAP"
2296   type="eps-wsrf:ExecutionPlanningService">
2297   <soap:binding style="document"
2298     transport="http://schemas.xmlsoap.org/soap/http" />
2299   <wsdl:operation name="GetCandidates">
2300     <soap:operation
2301       soapAction="http://schemas.ogf.org/rss/2007/03/eps/wsrif/GetExecutionPlans" />
2302     <wsdl:input>
2303       <soap:body use="literal" />
2304     </wsdl:input>
2305     <wsdl:output>
2306       <soap:body use="literal" />
2307     </wsdl:output>
2308     <wsdl:fault name="UnsupportedOrderingLanguageFault">
2309       <soap:fault
2310         namespace="http://schemas.ogf.org/rss/2007/03/eps/wsrif"
2311         use="literal" name="UnsupportedOrderingLanguageFault" />
2312     </wsdl:fault>
2313     <wsdl:fault name="BadPlanCardinalityRulesFault">
2314       <soap:fault
2315         namespace="http://schemas.ogf.org/rss/2007/03/eps/wsrif"
2316         use="literal" name="BadPlanCardinalityRulesFault" />
2317     </wsdl:fault>
2318     <wsdl:fault name="UnableToGeneratePlansFault">
2319       <soap:fault
2320         namespace="http://schemas.ogf.org/rss/2007/03/eps/wsrif"
2321         use="literal" name="UnableToGeneratePlansFault" />
2322     </wsdl:fault>
2323     <wsdl:fault name="ResourceUnknownFault">
2324       <soap:fault
2325         namespace="http://schemas.ogf.org/rss/2007/03/eps/wsrif"
2326         use="literal" name="ResourceUnknownFault" />
2327     </wsdl:fault>
2328   </wsdl:operation>
2329   <wsdl:operation name="GetResourcePropertyDocument">
2330     <soap:operation
2331       soapAction="http://schemas.ogf.org/rss/2007/03/eps/wsrif/GetResourcePropertyDocument" />
2332     <wsdl:input name="GetResourcePropertyDocumentRequest">
2333       <soap:body use="literal" />
2334     </wsdl:input>
2335     <wsdl:output name="GetResourcePropertyDocumentResponse">
2336       <soap:body use="literal" />
2337     </wsdl:output>
2338     <wsdl:fault name="ResourceUnknownFault">
2339       <soap:fault
2340         namespace="http://schemas.ogf.org/rss/2007/03/eps/wsrif"
2341         use="literal" name="ResourceUnknownFault" />
2342     </wsdl:fault>
2343   </wsdl:operation>
2344 </wsdl:binding>
2345 <wsdl:binding name="iss-wsrf-SOAP" type="eps-wsrf:IterableSetService">
2346   <soap:binding style="document"
2347     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
```

```

2404     namespace="http://schemas.ogf.org/rss/2007/03/eps/wsrp"
2405     use="literal" name="UnableToSetTerminationTimeFault" />
2406 </wsdl:fault>
2407 <wsdl:fault name="ResourceUnknownFault">
2408     <soap:fault
2409         namespace="http://schemas.ogf.org/rss/2007/03/eps/wsrp"
2410         use="literal" name="ResourceUnknownFault" />
2411 </wsdl:fault>
2412 <wsdl:fault name="TerminationTimeChangeRejectedFault">
2413     <soap:fault
2414         namespace="http://schemas.ogf.org/rss/2007/03/eps/wsrp"
2415         use="literal" name="TerminationTimeChangeRejectedFault" />
2416 </wsdl:fault>
2417 </wsdl:operation>
2418 <wsdl:operation name="GetResourcePropertyDocument">
2419     <soap:operation
2420 soapAction="http://schemas.ogf.org/rss/2007/03/eps/wsrp/GetResourcePropertyDocument" />
2421     <wsdl:input name="GetResourcePropertyDocumentRequest">
2422         <soap:body use="literal" />
2423     </wsdl:input>
2424     <wsdl:output name="GetResourcePropertyDocumentResponse">
2425         <soap:body use="literal" />
2426     </wsdl:output>
2427     <wsdl:fault name="ResourceUnknownFault">
2428         <soap:fault
2429             namespace="http://schemas.ogf.org/rss/2007/03/eps/wsrp"
2430             use="literal" name="ResourceUnknownFault" />
2431     </wsdl:fault>
2432 </wsdl:operation>
2433 </wsdl:binding>
2434
2435 <wsdl:service name="eps-wsrp-SOAP">
2436     <wsdl:documentation>
2437         The same service should normally implement both the EPS and the
2438         ISS port-types. The locations specified here are arbitrary though;
2439         callers should know who they are talking to independent of this
2440         specification, and the reference to the ISS port type is
2441         transferred to the client as an endpoint reference.
2442     </wsdl:documentation>
2443     <wsdl:port binding="eps-wsrp:eps-wsrp-SOAP" name="eps-wsrp-SOAP">
2444         <soap:address location="http://localhost/" />
2445     </wsdl:port>
2446     <wsdl:port name="iss-wsrp-SOAP" binding="eps-wsrp:iss-wsrp-SOAP">
2447         <soap:address location="http://localhost/" />
2448     </wsdl:port>
2449 </wsdl:service>
2450 </wsdl:definitions>

```

2451 A.5 XML Schema for Basic-EPS-XF

```

2452 <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2453 <xsd:schema targetNamespace="http://schemas.ogf.org/rss/2007/10/eps/transfer"
2454     xmlns:csg="http://schemas.ogf.org/rss/2007/03/csg"
2455     xmlns:wsen="http://schemas.xmlsoap.org/ws/2004/09/enumeration"
2456     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
2457     elementFormDefault="qualified" xml:lang="en">

```

```

2458
2459 <xsd:annotation>
2460   <xsd:documentation>
2461     This schema describes the elements that are only used when
2462     implementing an EPS on top of WS-Transfer and WS-Enumeration.
2463
2464     Copyright © 2007, Open Grid Forum
2465   </xsd:documentation>
2466   <!-- Need to check the copyright and get the full copyright statement in here. -->
2467 </xsd:annotation>
2468
2469 <xsd:import namespace="http://schemas.ogf.org/rss/2007/03/csg"
2470   schemaLocation="csg.xsd" />
2471 <xsd:import namespace="http://schemas.xmlsoap.org/ws/2004/09/enumeration"
2472   schemaLocation="wsen-hacks.xsd" />
2473
2474 <xsd:element name="EnumerateCandidatesRequest">
2475   <xsd:annotation>
2476     <xsd:documentation>
2477       The format of a request to enumerate candidates.
2478     </xsd:documentation>
2479   </xsd:annotation>
2480   <xsd:complexType>
2481     <xsd:sequence>
2482       <xsd:element ref="csg:RequestCandidates" />
2483       <xsd:element ref="wsen:EndTo" minOccurs="0" />
2484       <xsd:element ref="wsen:Expires" minOccurs="0" />
2485     </xsd:sequence>
2486   </xsd:complexType>
2487 </xsd:element>
2488
2489 <xsd:complexType name="EPSRepresentation">
2490   <xsd:annotation>
2491     <xsd:documentation>
2492       The minimal required representation for an EPS when using
2493       WS-Transfer is to describe the usage profiles and ordering
2494       languages that are supported.
2495     </xsd:documentation>
2496   </xsd:annotation>
2497   <xsd:sequence>
2498     <xsd:element ref="csg:UsageProfile" minOccurs="1"
2499       maxOccurs="unbounded" />
2500     <xsd:element ref="csg:SupportedOrderingLanguages" minOccurs="1"
2501       maxOccurs="unbounded" />
2502   </xsd:sequence>
2503 </xsd:complexType>
2504 </xsd:schema>

```

2505 A.6 WSDL for Basic-EPS-XF

```

2506 <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2507 <wsdl:definitions name="eps-xf" xml:lang="en"
2508   xmlns:eps-xf="http://schemas.ogf.org/rss/2007/10/eps/transfer"
2509   xmlns:csg="http://schemas.ogf.org/rss/2007/03/csg"
2510   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
2511   xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"

```

```

2512   xmlns:wSDL="http://schemas.xmlsoap.org/wSDL/"
2513   xmlns:wSen="http://schemas.xmlsoap.org/ws/2004/09/enumeration"
2514   xmlns:wSxf="http://schemas.xmlsoap.org/ws/2004/09/transfer"
2515   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
2516   targetNamespace="http://schemas.ogf.org/rss/2007/10/eps/transfer">
2517   <wSDL:documentation>
2518     Defines the required parts of an Execution Planning Service as
2519     implemented using WS-Transfer and WS-Enumeration.
2520
2521     Copyright © 2007, Open Grid Forum
2522   </wSDL:documentation>
2523
2524   <wSDL:types>
2525     <xsd:schema>
2526       <xsd:import
2527         namespace="http://schemas.ogf.org/rss/2007/10/eps/transfer"
2528         schemaLocation="eps-transfer.xsd" />
2529       <xsd:import namespace="http://schemas.ogf.org/rss/2007/03/csg"
2530         schemaLocation="csg.xsd" />
2531       <xsd:import
2532         schemaLocation="http://schemas.xmlsoap.org/ws/2004/09/enumeration/enumeration.xsd"
2533         namespace="http://schemas.xmlsoap.org/ws/2004/09/enumeration" />
2534       <xsd:import
2535         namespace="http://schemas.xmlsoap.org/ws/2004/08/addressing"
2536         schemaLocation="http://schemas.xmlsoap.org/ws/2004/08/addressing" />
2537     </xsd:schema>
2538   </wSDL:types>
2539   <wSDL:import
2540     location="http://schemas.xmlsoap.org/ws/2004/09/transfer/transfer.wSDL"
2541     namespace="http://schemas.xmlsoap.org/ws/2004/09/transfer" />
2542
2543   <wSDL:message name="EnumerateCandidatesRequest">
2544     <wSDL:part element="eps-xf:EnumerateCandidatesRequest" name="Body" />
2545   </wSDL:message>
2546   <wSDL:message name="EnumerateCandidatesResponse">
2547     <wSDL:part element="wSen:EnumerateResponse" name="Body" />
2548   </wSDL:message>
2549   <wSDL:message name="EPSRepresentation">
2550     <wSDL:part name="Body" type="eps-xf:EPSRepresentation" />
2551   </wSDL:message>
2552   <wSDL:portType name="eps-transfer">
2553     <wSDL:operation name="EnumerateCandidates">
2554       <wSDL:input message="eps-xf:EnumerateCandidatesRequest"
2555         wsa:Action="http://schemas.ogf.org/rss/2007/10/eps/transfer/EnumerateCandidates" />
2556       <wSDL:output message="eps-xf:EnumerateCandidatesResponse"
2557         wsa:Action="http://schemas.xmlsoap.org/ws/2004/09/enumeration/EnumerateResponse" />
2558     </wSDL:operation>
2559     <wSDL:operation name="Get">
2560       <wSDL:input message="wSxf:EmptyMessage"
2561         wsa:Action="http://schemas.xmlsoap.org/ws/2004/09/transfer/Get" />
2562       <wSDL:output message="eps-xf:EPSRepresentation"
2563         wsa:Action="http://schemas.xmlsoap.org/ws/2004/09/transfer/GetResponse" />
2564     </wSDL:operation>
2565   </wSDL:portType>
2566   <wSDL:binding name="eps-transferSOAP" type="eps-xf:eps-transfer">
2567     <soap:binding style="document"

```

```

2568     transport="http://schemas.xmlsoap.org/soap/http" />
2569 <wsdl:operation name="EnumerateCandidates">
2570   <soap:operation
2571     soapAction="http://schemas.ogf.org/rss/2007/10/eps/transfer/EnumerateCandidates" />
2572   <wsdl:input>
2573     <soap:body use="literal" />
2574   </wsdl:input>
2575   <wsdl:output>
2576     <soap:body use="literal" />
2577   </wsdl:output>
2578 </wsdl:operation>
2579 <wsdl:operation name="Get">
2580   <soap:operation
2581     soapAction="http://schemas.xmlsoap.org/ws/2004/09/transfer/Get" />
2582   <wsdl:input>
2583     <soap:body use="literal" />
2584   </wsdl:input>
2585   <wsdl:output>
2586     <soap:body use="literal" />
2587   </wsdl:output>
2588 </wsdl:operation>
2589 </wsdl:binding>
2590 <wsdl:service name="eps-transfer">
2591   <wsdl:port binding="eps-xf:eps-transferSOAP"
2592     name="eps-transferSOAP">
2593     <soap:address location="http://localhost/" />
2594   </wsdl:port>
2595 </wsdl:service>
2596 </wsdl:definitions>

```

2597 A.7 Supplemental XML Schema for WS-Enumeration

```

2598 <?xml version="1.0" encoding="UTF-8"?>
2599 <xsd:schema
2600   targetNamespace="http://schemas.xmlsoap.org/ws/2004/09/enumeration"
2601   xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
2602   xmlns:wsen="http://schemas.xmlsoap.org/ws/2004/09/enumeration"
2603   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
2604   elementFormDefault="qualified" xml:lang="en">
2605   <xsd:annotation>
2606     <xsd:documentation>
2607       This just adds the EndTo and Expires elements as top-level
2608       elements, as they were missing from the original WS-Enumeration
2609       specification.
2610
2611       Copyright © 2007, Open Grid Forum
2612     </xsd:documentation>
2613   </xsd:annotation>
2614   <xsd:include
2615     schemaLocation="http://schemas.xmlsoap.org/ws/2004/09/enumeration/enumeration.xsd">
2616   </xsd:include>
2617   <xsd:import namespace="http://schemas.xmlsoap.org/ws/2004/08/addressing"
2618     schemaLocation="http://schemas.xmlsoap.org/ws/2004/08/addressing/addressing.xsd" />
2619   <xsd:element name="EndTo" type="wsa:EndpointReferenceType" />
2620   <xsd:element name="Expires" type="wsen:ExpirationType" />
2621 </xsd:schema>

```

2622 A.8 XML Schema for SCOL

```

2623 <?xml version="1.0" encoding="UTF-8"?>
2624 <xsd:schema targetNamespace="http://schemas.ogf.org/rss/2007/05/scol"
2625   xmlns:select="http://schemas.ogf.org/rss/2007/05/scol"
2626   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
2627   elementFormDefault="qualified" attributeFormDefault="unqualified"
2628   xml:lang="en">
2629
2630   <xsd:annotation>
2631     <xsd:documentation>
2632       This describes the types and elements used to describe the RSS
2633       Simple Candidate Ordering Language.
2634
2635       Copyright © 2006-2007, Open Grid Forum
2636     </xsd:documentation>
2637   </xsd:annotation>
2638
2639   <!-- Basic types -->
2640   <xsd:simpleType name="XPathType">
2641     <xsd:annotation>
2642       <xsd:documentation>
2643         The string values of this type MUST be XPath 2.0 expressions,
2644         but the W3C does not seem to have defined an XSD type for
2645         this. It is up to the implementing application to enforce this
2646         restriction.
2647       </xsd:documentation>
2648     </xsd:annotation>
2649     <xsd:restriction base="xsd:string" />
2650   </xsd:simpleType>
2651   <xsd:complexType name="empty">
2652     <xsd:annotation>
2653       <xsd:documentation>
2654         The type of elements with no content.
2655       </xsd:documentation>
2656     </xsd:annotation>
2657     <xsd:sequence minOccurs="0" maxOccurs="0" />
2658   </xsd:complexType>
2659   <xsd:simpleType name="SelectOperation">
2660     <xsd:annotation>
2661       <xsd:documentation>
2662         Enumeration of ways to convert a NodeSet to a single double.
2663       </xsd:documentation>
2664     </xsd:annotation>
2665     <xsd:restriction base="xsd:string">
2666       <xsd:enumeration value="first">
2667         <xsd:annotation>
2668           <xsd:documentation>
2669             Value is the value of the first node in the NodeSet
2670             resulting from the XPath selection, interpreted as a
2671             double. If the node is an Element, then concatenate
2672             all the children of the Element that are Text nodes,
2673             and parse the resulting string. The evaluation of the
2674             containing select:Select term MUST fail if the NodeSet
2675             is empty or the parsed string cannot be interpreted as
2676             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

```

```

2733     namespace specifiers attached to it.
2734     </xsd:documentation>
2735   </xsd:annotation>
2736 </xsd:element>
2737 <xsd:element name="Bind">
2738   <xsd:annotation>
2739     <xsd:documentation>
2740       This element allows the specification of bindings of
2741       namespaces to prefixes for use with the Path element.
2742     </xsd:documentation>
2743   </xsd:annotation>
2744   <xsd:complexType>
2745     <xsd:complexContent>
2746       <xsd:extension base="select:empty">
2747         <xsd:attribute name="namespace" type="xsd:anyURI"
2748           use="required" />
2749         <xsd:attribute name="prefix" type="xsd:NCName"
2750           use="required" />
2751       </xsd:extension>
2752     </xsd:complexContent>
2753   </xsd:complexType>
2754 </xsd:element>
2755 <xsd:element name="OrderFunction">
2756   <xsd:annotation>
2757     <xsd:documentation>
2758       This element acts as a container for a term, representing the
2759       whole concept of an ordering function instead of just how to
2760       score an individual candidate. Conceptually, every candidate
2761       is scored (as a floating-point value) by the embedded term and
2762       then the candidates are ordered by their scores, smallest
2763       first.
2764     </xsd:documentation>
2765   </xsd:annotation>
2766   <xsd:complexType>
2767     <xsd:sequence>
2768       <xsd:element ref="select:Term" minOccurs="1" maxOccurs="1" />
2769     </xsd:sequence>
2770   </xsd:complexType>
2771 </xsd:element>
2772
2773 <!-- Terms, the whole point of this schema -->
2774 <xsd:complexType name="Term">
2775   <xsd:annotation>
2776     <xsd:documentation>
2777       Base type of all terms. Conceptually, a term describes how to
2778       calculate a double-precision floating-point value, and
2779       evaluates to either a double value in the range (-Inf..+Inf)
2780       or a failure. If the evaluation of an overall term expression
2781       for a context document fails, that context document MUST be
2782       rejected from the set of legal context documents.
2783     </xsd:documentation>
2784   </xsd:annotation>
2785   <xsd:complexContent>
2786     <xsd:extension base="select:empty" />
2787   </xsd:complexContent>
2788 </xsd:complexType>

```



```

2789 <xsd:element name="Term" abstract="true" type="select:Term">
2790   <xsd:annotation>
2791     <xsd:documentation>
2792       Abstract base element of all terms, used whenever any concrete
2793       term is usable in a particular location. Conceptually, a term
2794       describes how to calculate a double-precision floating-point
2795       value, and evaluates to either a double value in the range
2796       (-Inf..+Inf) or a failure. If the evaluation of an overall
2797       term expression for a context document fails, that context
2798       document MUST be rejected from the set of legal context
2799       documents.
2800     </xsd:documentation>
2801   </xsd:annotation>
2802 </xsd:element>
2803 <xsd:element name="Sum" substitutionGroup="select:Term">
2804   <xsd:annotation>
2805     <xsd:documentation>
2806       A term that calculates its value by summing the values of the
2807       terms inside it. The evaluation of this term MUST fail if the
2808       evaluation of any of the contained terms fails.
2809     </xsd:documentation>
2810   </xsd:annotation>
2811   <xsd:complexType>
2812     <xsd:complexContent>
2813       <xsd:extension base="select:Term">
2814         <xsd:sequence>
2815           <xsd:element ref="select:Term" minOccurs="2"
2816             maxOccurs="unbounded" />
2817         </xsd:sequence>
2818       </xsd:extension>
2819     </xsd:complexContent>
2820   </xsd:complexType>
2821 </xsd:element>
2822 <xsd:element name="Product" substitutionGroup="select:Term">
2823   <xsd:annotation>
2824     <xsd:documentation>
2825       A term that calculates its value by multiplying together the
2826       values of the terms inside it. The evaluation of this term
2827       MUST fail if the evaluation of any of the contained terms
2828       fails.
2829     </xsd:documentation>
2830   </xsd:annotation>
2831   <xsd:complexType>
2832     <xsd:complexContent>
2833       <xsd:extension base="select:Term">
2834         <xsd:sequence>
2835           <xsd:element ref="select:Term" minOccurs="2"
2836             maxOccurs="unbounded" />
2837         </xsd:sequence>
2838       </xsd:extension>
2839     </xsd:complexContent>
2840   </xsd:complexType>
2841 </xsd:element>
2842 <xsd:element name="Power" substitutionGroup="select:Term">
2843   <xsd:annotation>
2844     <xsd:documentation>

```

```

2845     A term that calculates its value by raising the value of the
2846     term inside it to some (fixed) power. The evaluation MUST fail
2847     if an attempt to calculate a non-integral power of a negative
2848     value is made, or if the evaluation of the contained term
2849     fails.
2850     </xsd:documentation>
2851 </xsd:annotation>
2852 <xsd:complexType>
2853   <xsd:complexContent>
2854     <xsd:extension base="select:Term">
2855       <xsd:sequence>
2856         <xsd:element ref="select:Term" minOccurs="1"
2857           maxOccurs="1" />
2858       </xsd:sequence>
2859       <xsd:attribute name="exponent" type="xsd:double"
2860         use="required" />
2861     </xsd:extension>
2862   </xsd:complexContent>
2863 </xsd:complexType>
2864 </xsd:element>
2865 <xsd:element name="Negate" substitutionGroup="select:Term">
2866   <xsd:annotation>
2867     <xsd:documentation>
2868       A term that calculates its value by negating the value of the
2869       term inside it. The evaluation MUST fail if the evaluation of
2870       the contained term fails.
2871     </xsd:documentation>
2872   </xsd:annotation>
2873   <xsd:complexType>
2874     <xsd:complexContent>
2875       <xsd:extension base="select:Term">
2876         <xsd:sequence>
2877           <xsd:element ref="select:Term" minOccurs="1"
2878             maxOccurs="1" />
2879         </xsd:sequence>
2880       </xsd:extension>
2881     </xsd:complexContent>
2882   </xsd:complexType>
2883 </xsd:element>
2884 <xsd:element name="Log" substitutionGroup="select:Term">
2885   <xsd:annotation>
2886     <xsd:documentation>
2887       A term that calculates its value by taking the logarithm of
2888       the value of the term inside it. The base of the logarithm
2889       defaults to e. The evaluation of this term MUST fail if the
2890       evaluation of the contained term fails or results in a
2891       negative number. The base attribute MUST be positive.
2892     </xsd:documentation>
2893   </xsd:annotation>
2894   <xsd:complexType>
2895     <xsd:complexContent>
2896       <xsd:extension base="select:Term">
2897         <xsd:sequence>
2898           <xsd:element ref="select:Term" minOccurs="1"
2899             maxOccurs="1" />
2900         </xsd:sequence>

```

```

2901     <xsd:attribute name="base" type="xsd:double"
2902         default="2.71828182846" />
2903     </xsd:extension>
2904 </xsd:complexContent>
2905 </xsd:complexType>
2906 </xsd:element>
2907 <xsd:element name="Exp" substitutionGroup="select:Term">
2908     <xsd:annotation>
2909         <xsd:documentation>
2910             A term that calculates its value by raising some fixed base to
2911             the degree of the value of the term inside it. The base of the
2912             exponent defaults to e. The base MUST be positive. The
2913             evaluation of this term MUST fail if the evaluation of the
2914             contained term fails.
2915         </xsd:documentation>
2916     </xsd:annotation>
2917 <xsd:complexType>
2918     <xsd:complexContent>
2919         <xsd:extension base="select:Term">
2920             <xsd:sequence>
2921                 <xsd:element ref="select:Term" minOccurs="1"
2922                     maxOccurs="1" />
2923             </xsd:sequence>
2924             <xsd:attribute name="base" type="xsd:double"
2925                 default="2.71828182846" />
2926         </xsd:extension>
2927     </xsd:complexContent>
2928 </xsd:complexType>
2929 </xsd:element>
2930 <xsd:element name="Abs" substitutionGroup="select:Term">
2931     <xsd:annotation>
2932         <xsd:documentation>
2933             A term that calculates its value by taking the absolute value
2934             of the term inside it (i.e. negating the value if-and-only-if
2935             the value is negative). The evaluation of this term MUST fail
2936             if the evaluation of the contained term fails.
2937         </xsd:documentation>
2938     </xsd:annotation>
2939 <xsd:complexType>
2940     <xsd:complexContent>
2941         <xsd:extension base="select:Term">
2942             <xsd:sequence>
2943                 <xsd:element ref="select:Term" minOccurs="1"
2944                     maxOccurs="1" />
2945             </xsd:sequence>
2946         </xsd:extension>
2947     </xsd:complexContent>
2948 </xsd:complexType>
2949 </xsd:element>
2950 <xsd:element name="Constant" substitutionGroup="select:Term">
2951     <xsd:annotation>
2952         <xsd:documentation>
2953             A term whose value is a constant double-precision
2954             floating-point value. The evaluation of this term MUST NOT
2955             fail.
2956         </xsd:documentation>

```

```

2957     </xsd:annotation>
2958     <xsd:complexType>
2959       <xsd:complexContent>
2960         <xsd:extension base="select:Term">
2961           <xsd:sequence>
2962             <xsd:element ref="select:Value" />
2963           </xsd:sequence>
2964         </xsd:extension>
2965       </xsd:complexContent>
2966     </xsd:complexType>
2967 </xsd:element>
2968 <xsd:element name="Select" substitutionGroup="select:Term">
2969   <xsd:annotation>
2970     <xsd:documentation>
2971       A term whose value is computed from the set of nodes returned
2972       by an XPath search conducted over some document (i.e. the
2973       context document which we are computing an overall value for).
2974       The computation from the NodeSet is to be performed as
2975       follows: if a baseInstant attribute is specified on the Select
2976       element, then the NodeSet is to be replaced with another
2977       NodeSet where each node of the set is replaced by a node (of
2978       type xsd:double) that represents the number of seconds from
2979       the baseInstant to the time parsed out of the original Node; if
2980       no baseInstant is specified, the node set MUST NOT be
2981       interpreted as containing time instants. Then the set of Nodes
2982       is converted to a double according to the action specified in
2983       the operation attribute; see the documentation of the
2984       select:SelectOperation type for details. The evaluation of
2985       this term MUST fail if the contained select:Path element does
2986       not hold a legal XPath expression; see the documentation of
2987       the select:SelectOperation type for other failure modes.
2988     </xsd:documentation>
2989   </xsd:annotation>
2990   <xsd:complexType>
2991     <xsd:complexContent>
2992       <xsd:extension base="select:Term">
2993         <xsd:sequence>
2994           <!-- Question: what about the namespace map? -->
2995           <xsd:element ref="select:Bind" maxOccurs="unbounded"
2996             minOccurs="0" />
2997           <xsd:element ref="select:Path" />
2998         </xsd:sequence>
2999         <xsd:attribute name="baseInstant" type="xsd:dateTime"
3000           use="optional">
3001           <xsd:annotation>
3002             <xsd:documentation>
3003               When nodes in the NodeSet returned from the
3004               XPath search represent time instants, they
3005               should be converted into numeric values by
3006               taking the number of seconds from the instant
3007               defined in this attribute and the instant in
3008               the node. It SHOULD be an error to set this
3009               attribute if the nodes are not interpretable
3010               as times. If any node is an Element, it should
3011               be parsed as if it was the concatenation of
3012               all the children of the element that are Text

```

```

3013         nodes.
3014         </xsd:documentation>
3015     </xsd:annotation>
3016 </xsd:attribute>
3017 <xsd:attribute name="operation"
3018     type="select:SelectOperation" default="first"
3019     use="optional">
3020 <xsd:annotation>
3021     <xsd:documentation>
3022         How to convert a NodeSet into a single double.
3023         Defaults to taking the first element of the
3024         NodeSet and just using that. The evaluation
3025         failure modes of the containing select:Select
3026         term are partially determined by this
3027         attribute; see the documentation of the
3028         select:SelectOperation type for details.
3029     </xsd:documentation>
3030 </xsd:annotation>
3031 </xsd:attribute>
3032 </xsd:extension>
3033 </xsd:complexContent>
3034 </xsd:complexType>
3035 </xsd:element>
3036 <xsd:element name="Bound" substitutionGroup="select:Term">
3037 <xsd:annotation>
3038     <xsd:documentation>
3039         This element evaluates its contained term and returns it only
3040         if it is within the range of values described by the pair of
3041         bounding attributes. If the bounding constraints are not
3042         satisfied, the whole term evaluation MUST fail. The evaluation
3043         of this term MUST fail if the evaluation of the contained term
3044         fails.
3045     </xsd:documentation>
3046 </xsd:annotation>
3047 <xsd:complexType>
3048     <xsd:complexContent>
3049         <xsd:extension base="select:Term">
3050             <xsd:sequence>
3051                 <xsd:element ref="select:Term" minOccurs="1"
3052                 maxOccurs="1" />
3053             </xsd:sequence>
3054             <xsd:attribute name="lowerBound" type="xsd:double"
3055             use="optional">
3056                 <xsd:annotation>
3057                     <xsd:documentation>
3058                         This indicates the lower (inclusive) bound of
3059                         the term's value. If omitted, there is no
3060                         lower bound (or, alternatively, the lower
3061                         bound is negative infinity).
3062                     </xsd:documentation>
3063                 </xsd:annotation>
3064             </xsd:attribute>
3065             <xsd:attribute name="upperBound" type="xsd:double"
3066             use="optional">
3067                 <xsd:annotation>
3068                     <xsd:documentation>

```

```
3069         This indicates the upper (inclusive) bound of
3070         the term's value. If omitted, there is no
3071         upper bound (or, alternatively, the upper
3072         bound is positive infinity).
3073         </xsd:documentation>
3074     </xsd:annotation>
3075 </xsd:attribute>
3076 </xsd:extension>
3077 </xsd:complexContent>
3078 </xsd:complexType>
3079 </xsd:element>
3080 <xsd:element name="OneOf" substitutionGroup="select:Term">
3081     <xsd:annotation>
3082         <xsd:documentation>
3083             This element has the value of the first contained term in it
3084             that evaluates successfully. All subsequent contained terms
3085             after the first successfully-computed one MUST NOT be
3086             evaluated. The evaluation of this term MUST fail if and only
3087             if the evaluation of all contained terms fails.
3088         </xsd:documentation>
3089     </xsd:annotation>
3090 <xsd:complexType>
3091     <xsd:complexContent>
3092         <xsd:extension base="select:Term">
3093             <xsd:sequence>
3094                 <xsd:element ref="select:Term" minOccurs="1"
3095                     maxOccurs="unbounded" />
3096             </xsd:sequence>
3097         </xsd:extension>
3098     </xsd:complexContent>
3099 </xsd:complexType>
3100 </xsd:element>
3101 </xsd:schema>
```