



OASIS ebXML RegRep Service and Protocols (ebRS) Version 4.0

Draft

October 1, 2008

Specification URIs:

This Version:

<http://docs.oasis-open.org/regrep/4.0-draft-1/specs/core/regrep-rs-4.0-draft-1.html>

<http://docs.oasis-open.org/regrep/4.0-draft-1/specs/core/regrep-rs-4.0-draft-1.odt>

<http://docs.oasis-open.org/regrep/4.0-draft-1/specs/core/regrep-rs-4.0-draft-1.pdf>

Previous Version:

<http://docs.oasis-open.org/regrep/v3.0/specs/regrep-rs-3.0-os.html>

<http://docs.oasis-open.org/regrep/v3.0/specs/regrep-rs-3.0-os.odt>

<http://docs.oasis-open.org/regrep/v3.0/specs/regrep-rs-3.0-os.pdf>

Latest Version:

<http://docs.oasis-open.org/regrep/4.0-draft-1/specs/core/regrep-rs-4.0-draft-1.html>

<http://docs.oasis-open.org/regrep/4.0-draft-1/specs/core/regrep-rs-4.0-draft-1.odt>

<http://docs.oasis-open.org/regrep/4.0-draft-1/specs/core/regrep-rs-4.0-draft-1.pdf>

Latest Approved Version:

<http://docs.oasis-open.org/regrep/v3.0/specs/regrep-rs-3.0-os.html>

<http://docs.oasis-open.org/regrep/v3.0/specs/regrep-rs-3.0-os.odt>

<http://docs.oasis-open.org/regrep/v3.0/specs/regrep-rs-3.0-os.pdf>

Technical Committee:

OASIS ebXML Registry TC

Chair(s):

Kathryn Breininger, Boeing

Editor(s):

Farrukh Najmi, Wellfleet Software

Nikola Stojanovic, RosettaNet

30 **Contributors:**

- 31 Kathryn Breininger, Boeing
- 32 Carl Mattocks, MetLife
- 33 Farrukh Najmi, Wellfleet Software
- 34 Oliver Newell, MIT Lincoln Labs
- 35 Nikola Stojanovic, RosettaNet
- 36 David Webber, Individual

37 **Related Work:**

38 This specification replaces or supercedes:

- 39 • [specifications replaced by this standard - OASIS as well as other standards organizations]
- 40 • [specifications replaced by this standard - OASIS as well as other standards organizations]

41 This specification is related to:

- 42 • [specifications related to this standard - OASIS as well as other standards organizations]
- 43 • [specifications related to this standard - OASIS as well as other standards organizations]

44 **Declared XML Namespace(s):**

45

46 This following table lists the namespace prefixes defined and / or referenced by this specification.

47

Namespace Prefix	Namespace URI	Defining Specification
enc	http://www.w3.org/2003/05/soap-encoding	A normative XML Schema [XML Schema Part 1], [XML Schema Part 2] document for the "http://www.w3.org/2003/05/soap-encoding" namespace can be found at http://www.w3.org/2003/05/soap-encoding .
env	http://www.w3.org/2003/05/soap-envelope	SOAP Version 1.2 Part 1. A normative XML Schema [XML Schema Part 1], [XML Schema Part 2] document for the "http://www.w3.org/2003/05/soap-envelope" namespace can be found at http://www.w3.org/2003/05/soap-envelope .
lcm	urn:oasis:names:tc:ebxml-regrep:xsd:lcm:4.0	ebXML RegRep Services and Protocols 4.0 (ebRS)
mime	http://schemas.xmlsoap.org/wsdl/mime/	WSDL namespace for WSDL MIME binding.
query	urn:oasis:names:tc:ebxml-regrep:xsd:query:4.0	ebXML RegRep Services and Protocols 4.0 (ebRS)
rim	urn:oasis:names:tc:ebxml-regrep:xsd:rim:4.0	ebXML RegRep Registry Information Model 4.0 (ebRIM)
rs	urn:oasis:names:tc:ebxml-regrep:xsd:rs:4.0	ebXML RegRep Services and Protocols 4.0 (ebRS)
wsdl	http://schemas.xmlsoap.org/wsdl/	WSDL 1.1 namespace defined by WSDL 1.1 specification .
xs	http://www.w3.org/2001/XMLSchema	XML Schema [XML Schema Part 1], [XML Schema Part 2] specification
xsi	"http://www.w3.org/2001/XMLSchema-instance	W3C XML Schema specification [XML Schema Part 1], [XML Schema Part 2].

Table 1: Namespaces Used

48

49

50

51 **Abstract:**

52 This document defines the services and protocols for an ebXML Registry

53 A separate document, ebXML Registry: Information Model [ebRIM], defines the types of metadata
54 and content that can be stored in an ebXML Registry.

55 **Status:**

56 This document is a draft specification for review, revision and approval by the OASIS ebXML
57 Registry TC.

58 Technical Committee members should send comments on this specification to the Technical
59 Committee's email list. Others should send comments to the Technical Committee by using the

60 "Send A Comment" button on the Technical Committee's web page at [http://www.oasis-open.org/](http://www.oasis-open.org/committees/regrep/)
61 [committees/regrep/](http://www.oasis-open.org/committees/regrep/).

62 For information on whether any patents have been disclosed that may be essential to
63 implementing this specification, and any offers of patent licensing terms, please refer to the
64 Intellectual Property Rights section of the Technical Committee web page ([http://www.oasis-](http://www.oasis-open.org/committees/regrep/ipr.php)
65 [open.org/committees/regrep/ipr.php](http://www.oasis-open.org/committees/regrep/ipr.php)).

66 The non-normative errata page for this specification is located at [http://docs.oasis-](http://docs.oasis-open.org/regrep/4.0-draft-1/specs/core/errata.pdf)
67 [open.org/regrep/4.0-draft-1/specs/core/errata.pdf](http://docs.oasis-open.org/regrep/4.0-draft-1/specs/core/errata.pdf)

68 Notices

69 Copyright © OASIS® 2008. All Rights Reserved.

70 All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual
71 Property Rights Policy (the "OASIS IPR Policy"). The full Policy may be found at the OASIS website.

72 This document and translations of it may be copied and furnished to others, and derivative works that
73 comment on or otherwise explain it or assist in its implementation may be prepared, copied, published,
74 and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice
75 and this section are included on all such copies and derivative works. However, this document itself may
76 not be modified in any way, including by removing the copyright notice or references to OASIS, except as
77 needed for the purpose of developing any document or deliverable produced by an OASIS Technical
78 Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be
79 followed) or as required to translate it into languages other than English.

80 The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors
81 or assigns.

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

87 OASIS requests that any OASIS Party or any other party that believes it has patent claims that would
88 necessarily be infringed by implementations of this OASIS Committee Specification or OASIS Standard, to
89 notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such
90 patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced
91 this specification.

92 OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of
93 any patent claims that would necessarily be infringed by implementations of this specification by a patent
94 holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR
95 Mode of the OASIS Technical Committee that produced this specification. OASIS may include such
96 claims on its website, but disclaims any obligation to do so.

97 OASIS takes no position regarding the validity or scope of any intellectual property or other rights that
98 might be claimed to pertain to the implementation or use of the technology described in this document or
99 the extent to which any license under such rights might or might not be available; neither does it represent
100 that it has made any effort to identify any such rights. Information on OASIS' procedures with respect to
101 rights in any document or deliverable produced by an OASIS Technical Committee can be found on the
102 OASIS website. Copies of claims of rights made available for publication and any assurances of licenses
103 to be made available, or the result of an attempt made to obtain a general license or permission for the
104 use of such proprietary rights by implementers or users of this OASIS Committee Specification or OASIS
105 Standard, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any
106 information or list of intellectual property rights will at any time be complete, or that any claims in such list
107 are, in fact, Essential Claims.

108 The names "OASIS", [insert specific trademarked names, abbreviations, etc. here] are trademarks of
109 [OASIS](#), the owner and developer of this specification, and should be used only to refer to the organization
110 and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications,

111 while reserving the right to enforce its marks against misleading uses. Please see [http://www.oasis-
open.org/who/trademark.php](http://www.oasis-
112 open.org/who/trademark.php) for above guidance.

113

114 Table of Contents

115	1 Introduction.....	11
116	1.1 Terminology.....	11
117	1.2 Normative References.....	11
118	1.3 Non-normative References.....	11
119	2 Overview.....	12
120	2.1 The Library Analogy.....	12
121	2.2 Core Specifications.....	12
122	3 QueryManager Interface.....	13
123	3.1 Parameterized Queries.....	13
124	3.2 Adhoc Queries.....	13
125	3.3 Query Protocol.....	13
126	3.3.1 QueryRequest.....	14
127	3.3.1.1 Syntax.....	14
128	3.3.1.2 Example.....	14
129	3.3.1.3 Description.....	15
130	3.3.1.4 Response.....	15
131	3.3.1.5 Exceptions.....	15
132	3.3.2 Element QueryInvocation.....	15
133	3.3.2.1 Syntax.....	16
134	3.3.2.2 Description:.....	16
135	3.3.3 Element ResponseOption.....	16
136	3.3.3.1 Syntax.....	16
137	3.3.3.2 Description:.....	17
138	3.3.4 QueryResponse.....	17
139	3.3.4.1 Syntax.....	17
140	3.3.4.2 Example.....	18
141	3.3.4.3 Description:.....	18
142	3.4 Parameterized Query Definition.....	18
143	3.5 Canonical Query: ArbitraryQuery.....	18
144	3.5.1 Parameter Summary.....	18
145	3.5.2 Query Semantics.....	19
146	3.6 Canonical Query: BasicQuery.....	19
147	3.6.1 Parameter Summary.....	19
148	3.6.2 Query Semantics.....	20
149	3.7 Canonical Query: FindByIdAndType.....	20
150	3.7.1 Parameter Summary.....	20
151	3.7.2 Query Semantics.....	20
152	3.8 Canonical Query: FindAssociations.....	20
153	3.8.1 Parameter Summary.....	20
154	3.8.2 Query Semantics.....	21
155	3.9 Canonical Query: FindAssociatedObjects.....	22
156	3.9.1 Parameter Summary.....	22

157	3.9.2 Query Semantics.....	23
158	3.10 Canonical Query: GetClassificationSchemesByld.....	23
159	3.10.1 Parameter Summary.....	23
160	3.10.2 Query Semantics.....	23
161	3.11 Canonical Query: GetRegistryPackagesByMemberId.....	24
162	3.11.1 Parameter Summary.....	24
163	3.11.2 Query Semantics.....	24
164	3.12 Canonical Query: GetMembersByRegistryPackageId.....	24
165	3.12.1 Parameter Summary.....	24
166	3.12.2 Query Semantics.....	24
167	3.13 Canonical Query: KeywordSearch.....	25
168	3.13.1 Canonical Indexes.....	25
169	3.13.2 Parameter Summary.....	26
170	3.13.3 Query Semantics.....	26
171	4 LifecycleManager Interface.....	28
172	4.1 SubmitObjects Protocol.....	28
173	4.1.1 SubmitObjectsRequest.....	28
174	4.1.1.1 Syntax.....	29
175	4.1.1.2 Description.....	29
176	4.1.1.3 Returns.....	29
177	4.1.1.4 Exceptions.....	30
178	4.1.2 RegistryResponse.....	30
179	4.1.3 Unique ID Generation.....	30
180	4.1.4 ID Attribute And Object References.....	30
181	4.1.5 Audit Trail Requirements.....	31
182	4.1.6 Sample SubmitObjectsRequest.....	31
183	4.2 The Update Objects Protocol.....	31
184	4.2.1 UpdateObjectsRequest.....	32
185	4.2.1.1 Syntax.....	32
186	4.2.1.2 Description.....	33
187	4.2.1.3 Returns.....	33
188	4.2.1.4 Exceptions.....	33
189	4.2.2 UpdateAction.....	33
190	4.2.2.1 Syntax.....	33
191	4.2.2.2 Description.....	34
192	4.2.3 Audit Trail Requirements.....	34
193	4.2.4 Sample UpdateObjectsRequest.....	35
194	4.3 RemoveObjects Protocol.....	35
195	4.3.1 RemoveObjectsRequest.....	35
196	4.3.1.1 Syntax.....	35
197	4.3.1.2 Description.....	36
198	4.3.1.3 Returns:.....	36
199	4.3.1.4 Exceptions:.....	36
200	4.3.2 Audit Trail Requirements.....	37
201	4.3.3 Sample UpdateObjectsRequest.....	37
202	5 Validator Interface.....	38
203	5.1 ValidateObjects Protocol.....	38
204	5.1.1 ValidateObjectsRequest.....	38

205	5.1.1.1 Syntax.....	38
206	5.1.1.2 Example.....	39
207	5.1.1.3 Description.....	39
208	5.1.1.4 Response.....	39
209	5.1.1.5 Exceptions.....	39
210	5.1.2 ValidateObjectsResponse.....	40
211	6 Cataloger Interface.....	41
212	6.1 CatalogObjects Protocol.....	41
213	6.1.1 CatalogObjectsRequest.....	41
214	6.1.1.1 Syntax.....	41
215	6.1.1.2 Example.....	42
216	6.1.1.3 Description.....	42
217	6.1.1.4 Response.....	42
218	6.1.1.5 Exceptions.....	42
219	6.1.2 CatalogObjectsResponse.....	42
220	6.1.2.1 Syntax.....	43
221	6.1.2.2 Example.....	43
222	6.1.2.3 Description.....	43
223	7 Server Plugin SPI.....	44
224	7.1 Query Plugins.....	44
225	7.1.1 Query Plugin Interface.....	44
226	7.2 Validator Plugins.....	44
227	7.2.1 Validator Plugin Interface.....	44
228	7.2.2 Canonical XML Validator Plugin.....	45
229	7.3 Cataloger Plugins.....	45
230	7.3.1 Cataloger Plugin Interface.....	45
231	7.3.2 Canonical XML Cataloger Plugin.....	46
232	8 Subscription and Notification.....	47
233	8.1 Server Events.....	47
234	8.1.1 Pruning of Events.....	47
235	8.2 Notifications.....	47
236	8.3 Creating a Subscription.....	47
237	8.3.1 Subscription Authorization.....	47
238	8.3.2 Subscription Quotas.....	48
239	8.3.3 Subscription Expiration.....	48
240	8.3.4 Event Selection.....	48
241	8.4 Event Delivery.....	48
242	8.4.1 Delivery Mode.....	48
243	8.4.2 Delivery to NotificationListener Web Service.....	49
244	8.4.3 Delivery to Email Address.....	49
245	8.4.3.1 Processing Email Notification Via XSLT.....	49
246	8.5 NotificationListener Interface.....	49
247	8.6 Notification Protocol.....	49
248	8.6.1 CatalogObjectsRequest.....	50
249	8.6.1.1 Syntax.....	50
250	8.6.1.2 Example.....	51
251	8.6.1.3 Description.....	51

252	8.6.1.4 Response.....	51
253	8.6.1.5 Exceptions.....	51
254	8.6.2 CatalogObjectsResponse.....	51
255	8.6.2.1 Syntax.....	51
256	8.6.2.2 Example.....	52
257	8.6.2.3 Description.....	52
258	8.7 Deleting a Subscription.....	52
259	9 Cooperating Registries.....	53
260	10 Security Features.....	54
261	11 Native Language Support (NLS).....	55
262	11.1 Terminology.....	55
263	11.2 NLS and Registry Protocol Messages.....	55
264	11.3 NLS Support in RegistryObjects	56
265	11.3.1 Character Set of LocalizedString.....	57
266	11.3.2 Language of LocalizedString.....	57
267	11.4 NLS and Repository Items	57
268	11.4.1 Character Set of Repository Items.....	57
269	11.4.2 Language of Repository Items.....	58
270	12 SOAP Binding.....	59
271		

Illustration Index

272	Illustration 1: Query Protocol Need to update figure??.....	14
	Illustration 2: SubmitObjects Protocol (replace??).....	28
	Illustration 3: UpdateObjects Protocol (replace??).....	32
	Illustration 4: RemoveObjects Protocol (replace??).....	35
	Illustration 5: ValidateObjects Protocol Need to update figure??.....	38
	Illustration 6: CatalogObjects Protocol Need to update figure??.....	41
	Illustration 7: CatalogObjects Protocol Need to update figure??.....	50

Index of Tables

273	Table 1: Namespaces Used.....	3
	Table 2: ebXML RegRep comparison with your local library.....	12

274 **1 Introduction**

275 [All text is normative unless otherwise indicated.]

276 **1.1 Terminology**

277 The keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD
278 NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this specification are to be interpreted as
279 described in IETF RFC 2119 .

280 **1.2 Normative References**

281 **[RFC 2119]** S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*. IETF
282 RFC 2119, March 1997. <http://www.ietf.org/rfc/rfc2119.txt>.

283 **[Reference]** [reference citation]

284 **1.3 Non-normative References**

285 **[Reference]** [reference citation]

286 **[Reference]** [reference citation]

287 2 Overview

288 ebXML RegRep is a standard defining the service interfaces, protocols and information model for an
289 integrated registry and repository. The repository is stores digital content while the repository stores
290 metadata that describes the content in the repository.

291 A more detailed overview of this specification is provided in the ebXML RegRep 4.0 Primer [Primer].

292 2.1 The Library Analogy

293 To explain what is an ebXML RegRep we use the following familiar analogy. The ebXML Registry-
294 Repository is to digital content, what your local library is to books and other published content. We make
295 this analogy clearer with the comparisons made in the following table:

296

Your Local Library	ebXML RegRep
Manages books and all types of published material	Manages all types of digital content
Has book shelves containing books and other published material	Has a "repository" containing all types of digital content
Has a card catalog that describes the published material that is available in the book shelves	Has a "registry" that describes the digital content that is available in the repository
Multiple libraries can voluntarily participate in a cooperative network and offer a unified service	Multiple ebXML Registry-Repository's can voluntarily participate in a cooperative network and offer a unified service

297 *Table 2: ebXML RegRep comparison with your local library*

298 2.2 Core Specifications

299 ebXML RegRep standard consists of two core specifications:

- 300 ● The *ebXML Registry Information Model (ebRIM)* specification defines the types of metadata that
301 can be stored in an ebXML Registry.
- 302 ● The *ebXML Registry Services and Protocols (ebRS)* defines the services provided by an ebXML
303 Registry and the protocols used by clients of the registry to interact with these services.

304

305 **3 QueryManager Interface**

306 The QueryManager interface allows a client to invoke queries on the server. A server MUST implement
307 the QueryManager interface as an endpoint.

308 **3.1 Parameterized Queries**

309 A server may support any number of pre-configured queries known as *Parameterized Queries*, that may
310 be invoked by clients. This specification defines a number of [canonical queries](#) that are standard queries
311 that MUST be supported by a server. Implementations and deployments may define additional
312 parameterized queries beyond the canonical queries defined by this specification.

313 A client invokes a parameterized query supported by the server by identifying its unique id as well as
314 values for any parameters supported by the query.

315 A parameterized query is represented by a specialized RegistryObject called ParameterizedQuery object
316 which is defined by ebRIM. The definition of a ParameterizedQuery may contain any number of
317 Parameters supported by the query.

318 **3.2 Adhoc Queries**

319 A client may invoke a client specific ad hoc query using the [canonical ArbitraryQuery query](#) defined by this
320 specification. Due to the risks associated with un-controlled ad hoc queries, a deployment MAY choose to
321 restrict the invocation of the ArbitraryQuery to specific roles. This specification does not define a standard
322 query expression syntax for ad hoc queries. A server MAY support any number of query expression
323 syntaxes for ad hoc queries.

324 **3.3 Query Protocol**

325 A client invokes a parameterized query using the *Query* protocol supported by the executeQuery
326 operation of the QueryManager interface.

327 A client initiates the Query protocol by sending an QueryRequest message to the QueryManager
328 endpoint.

329 The QueryManager sends an QueryResponse back to the client as response. The QueryResponse
330 contains a set of objects that match the query.

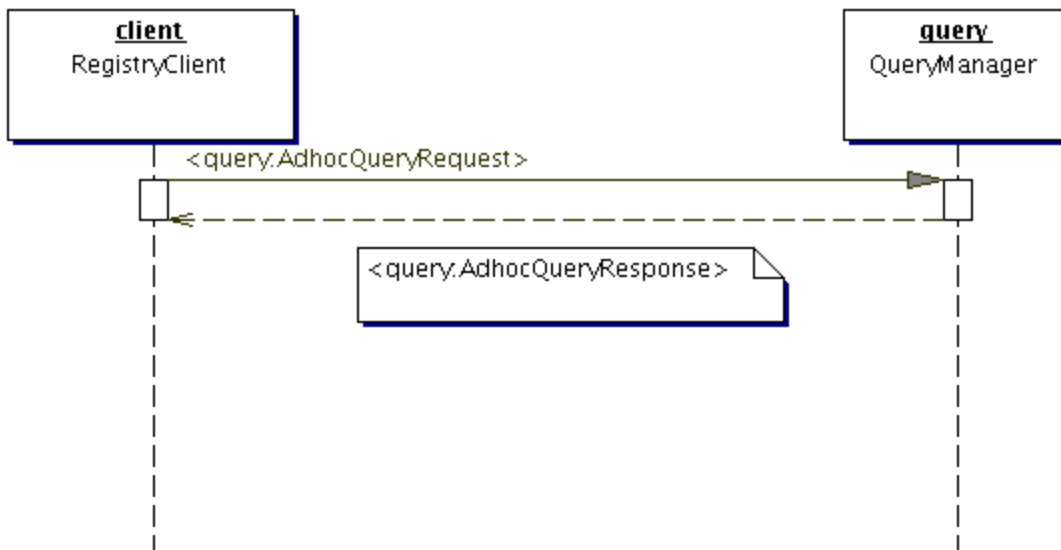


Illustration 1: Query Protocol **Need to update figure??**

332

333 3.3.1 QueryRequest

334 The QueryRequest message is sent by client to the QueryManager interface to invoke a query.

335 3.3.1.1 Syntax

```

336 <element name="QueryRequest">
337   <annotation>
338     <documentation xml:lang="en">Request to invoke a query.</documentation>
339   </annotation>
340   <complexType>
341     <complexContent>
342       <extension base="rs:RegistryRequestType">
343         <sequence>
344           <element maxOccurs="1" minOccurs="1" ref="tns:ResponseOption"/>
345           <element ref="rim:QueryInvocation"/>
346         </sequence>
347         <attribute default="application/ebrim+xml" name="format"
348 type="string" use="optional"/>
349         <attribute name="language" type="string" use="optional"/>
350         <attribute default="0" name="startIndex" type="integer"/>
351         <attribute default="-1" name="maxResults" type="integer"/>
352         <attribute default="0" name="depth" type="integer"/>
353       </extension>
354     </complexContent>
355   </complexType>
356 </element>
  
```

357 3.3.1.2 Example

358 TODO??

359 3.3.1.3 Description

360 **Issue: Need to list items in order of schema, schema should be in alpha order??**

- 361 ● **Element ResponseOption** - This required element allows the client to control the format and
362 content of the QueryResponse generated by the registry in response to this request.
- 363 ● **Element QueryInvocation** - This element identifies a parameterized query and supplies values for
364 its parameters.
- 365 ● **Attribute maxResults** - This optional attribute specifies a limit on the maximum number of results
366 the client wishes the query to return. If unspecified, the registry SHOULD return either all the
367 results, or in case the result set size exceeds a registry specific limit, the registry SHOULD return
368 a sub-set of results that are within the bounds of the registry specific limit.
- 369 ● **Attribute startIndex** - This optional integer value is used to indicate which result must be returned
370 as the first result when iterating over a large result set. The default value is 0, which returns the
371 result set starting with index 0 (first result).
- 372 ● **Attribute format** - This optional attribute specifies the format of the response desired by the client.
373 The value of this attribute MUST be a registered Internet Media Types with IANA. The default
374 value is "application/ebrs+xml?? Nikola to look at RFC 3023" which returns the response in ebRS
375 QueryResponse format.
- 376 ● **Attribute language** - This optional attribute specifies the natural language of the response desired
377 by the client. The default value is to return the response with all available natural languages. **What**
378 **coding standard to base this on. ISO 639, 639-1, 639-2??**
- 379 ● **Attribute depth** - This optional attribute specifies the pre-fetch depth of the response desired by
380 the client. A depth of 0 (default) indicates that the server MUST return only those objects that
381 match the query. A depth of N where N is greater than 0 indicates that the server MUST also
382 return objects that are reachable by N levels of references via attributes that reference other
383 objects. A depth of -1 indicates that the server MUST return all objects within the transitive closure
384 of all references from objects that matches the query.

385 3.3.1.4 Response

386 This request returns QueryResponse as response.

387 3.3.1.5 Exceptions

388 **Issue: Need to define more fine-grained exception hierarchy??**

389 In addition to the exceptions common to all requests defined in ??, the following exceptions MAY be returned:

- 390 ● **InvalidQueryException**: signifies that the query syntax or semantics was invalid. Client must fix the query
391 syntax or semantic error and re-submit the query

392 3.3.2 Element QueryInvocation

393 A client specifies a <rim:QueryInvocation> element within an QueryRequest to specify the parameterized
394 query being being invoked as well as the values for its parameters.

395 3.3.2.1 Syntax

```
396 <complexType name="QueryInvocationType">
397   <annotation>
398     <documentation xml:lang="en">
399       Represents invocation of a parameterized query.
400     </documentation>
401   </annotation>
402
403   <!--
404   References a ParameterizedQuery.
405   Note that reference may be static or dynamic.
406   -->
407   <sequence>
408     <element maxOccurs="unbounded" minOccurs="0" ref="tns:Slot"/>
409   </sequence>
410   <attribute name="query" type="tns:referenceURI" use="required"/>
411 </complexType>
412 <element name="QueryInvocation" type="tns:QueryInvocationType"/>
```

413

414 3.3.2.2 Description:

- 415 ● *Element Slot* - Each Slot element specifies a parameter value for a parameter supported by the
416 query. The slot name MUST match a parameterName attribute within a Parameter's definition
417 within the ParameterizedQuery (inconsistent on use of rim: prefix??) definition. The slot value
418 provides a value for the parameter. The slot value's type MUST match the dataType attribute for
419 the Parameter's definition within the ParameterizedQuery. Order of parameters is not significant.
- 420 ● Attribute query - The value of this attribute must be a reference to a parameterized query that is
421 supported by the server. Need to change type from referenceURI since id may not be a URI any
422 more and also needs to support dynamic references??

423 3.3.3 Element ResponseOption

424 A client specifies a ResponseOption structure within an QueryRequest to indicate the format of the results
425 within the corresponding QueryResponse.

426 3.3.3.1 Syntax

```
427 <complexType name="ResponseOptionType">
428   <attribute default="RegistryObject" name="returnType">
429     <simpleType>
430       <restriction base="NCName">
431         <enumeration value="ObjectRef"/>
432         <enumeration value="RegistryObject"/>
433         <enumeration value="LeafClass"/>
434         <enumeration value="LeafClassWithRepositoryItem"/>
435       </restriction>
436     </simpleType>
437   </attribute>
438   <attribute default="false" name="returnComposedObjects" type="boolean"/>
439 </complexType>
440 <element name="ResponseOption" type="tns:ResponseOptionType"/>
```


441 3.3.3.2 Description:

- 442 ● Attribute `returnComposedObjects` - This optional attribute specifies whether the RegistryObjects
443 returned should include composed objects as defined by Figure 1 in [ebRIM]. The default is to
444 return all composed objects.
- 445 ● Attribute `returnType` - This optional attribute specifies the type of RegistryObject to return within
446 the response. Values for `returnType` are as follows:
 - 447 ○ *ObjectRef* - This option specifies that the QueryResponse MUST contain a collection of
448 `<rim:ObjectRef>` elements. The purpose of this option is to return references to registry
449 objects rather than the actual objects.
 - 450 ○ *RegistryObject* - This option specifies that the QueryResponse MUST contain a collection of
451 `<rim:RegistryObject>` elements.
 - 452 ○ *LeafClass* - This option specifies that the QueryResponse MUST contain a collection of
453 elements that correspond to leaf classes as defined in [RR-RIM-XSD].
 - 454 ○ *LeafClassWithRepositoryItem* - This option is same as LeafClass option with the additional
455 requirement that the response include the RepositoryItems, if any, for every
456 `<rim:ExtrinsicObject>` element in the response.

457 If “returnType” specified does not match a result returned by the query, then the registry *must* use
458 the closest matching semantically valid returnType that matches the result. For example, consider
459 a case where OrganizationQuery is asked to return LeafClassWithRepositoryItem. As this is not
460 possible, QueryManager will assume LeafClass option instead.

461 3.3.4 QueryResponse

462 The QueryResponse message is sent by the QueryManager in response to an QueryRequest when the
463 format requested by the client is the default ebrs format.

464 3.3.4.1 Syntax

```
465 <element name="QueryResponse">  
466   <annotation>  
467     <documentation xml:lang="en">  
468       The response includes a RegistryObjectList which has zero or more  
469       RegistryObjects that match the query specified in QueryRequest.  
470     </documentation>  
471   </annotation>  
472   <complexType>  
473     <complexContent>  
474       <extension base="rs:RegistryResponseType">  
475         <sequence>  
476           <element ref="rim:RegistryObjectList"/>  
477         </sequence>  
478         <attribute default="0" name="startIndex" type="integer"/>  
479         <attribute name="totalResultCount" type="integer" use="optional"/>  
480       </extension>  
481     </complexContent>  
482   </complexType>  
483 </element>
```

484 **3.3.4.2 Example**

485 TODO??

486 **3.3.4.3 Description:**

- 487 ● Element RegistryObjectList - This is the element that contains the RegistryObject instances that
488 matched the specified query.
- 489 ● Attribute startIndex - This optional integer value is used to indicate the index for the first result in
490 the result set returned by the query, within the complete result set matching the query. By default,
491 this value is 0.
- 492 ● Attribute totalResultCount - This optional parameter specifies the size of the complete result set
493 matching the query within the registry. When this value is unspecified, the client should assume it
494 is the size of the result set contained within the result.

495 **3.4 Parameterized Query Definition**

496 A parameterized query is defined by submitting a <rim:ParameterizedQuery> object to the server using
497 the [submitObjects protocol](#). A detailed specification of the <rim:ParameterizedQuery> object is defined in
498 ebRIM. The definition of a parameterized query includes detailed specification of each supported
499 parameter including its name, description, data type, cardinality and domain.

500 **3.5 Canonical Query: ArbitraryQuery**

501 **Issue: Nikola has issue with this??**

502 The [canonical query ArbitraryQuery](#) allows clients to invoke a client-specified ad hoc query in a client-
503 specified query expression syntax that is supported by the server. This specification does not require a
504 server to support a specific query expression syntax. It is likely that servers may support one or more
505 common syntaxes such as SQL-92, XQuery, XPath, SPARQL, Search-WS, OGC Filter etc.

506 **3.5.1 Parameter Summary**

Parameter	Description	Data Type	Default Value	Cardinality
queryExpression	This parameter's value is an element of type <rim:QueryExpressionType> as defined by ebRIM. The queryExpression is used to carry the query expression for an ad hoc query. The queryExpression element also specifies the query language syntax via its queryLanguage attribute.	slot		1

507 **3.5.2 Query Semantics**

- 508 ● The server MUST return a InvalidQueryException fault message if the queryLanguage used by
- 509 the queryExpression is not supported by the server
- 510 ● The server SHOULD return an AuthorizationException fault message if the client is not authorized
- 511 to invoke this query
- 512 ● The server MUST return the objects matching the query if the query is processed without any
- 513 exceptions

514 **3.6 Canonical Query: BasicQuery**

515 The canonical query BasicQuery allows clients to query for RegistryObjects by their name, description,
 516 type, status and classifications.

517 **3.6.1 Parameter Summary**

Parameter	Description	Data Type	Default Value	Cardinality
classifications	Set whose elements are path attribute values to ClassificationNodes. Matches RegistryObjects that have a classification whose classificationNode attribute value matches the id of the ClassificationNode where rim:ClassificationNode/@path matches specified value When multiple values are specified it implies a logical AND operation.	string		0..*
description	Matches rim:RegistryObject/rim:Description/rim:LocalizedString/@value	string		0..1
matchOnAnyParameter	If true then use logical OR between predicates for each parameter	boolean	false	0..1
name	Matches rim:RegistryObject/rim:Name/rim:LocalizedString/@value	string		0..1
objectType	Matches RegistryObjects whose objectType attribute matches the id of the ClassificationNode where rim:ClassificationNode/@path matches specified value	string		0..1
status	Matches RegistryObjects whose status attribute matches the id of the ClassificationNode where rim:ClassificationNode/@path matches specified value	string		0..1

518 3.6.2 Query Semantics

- 519 ● This query has several optional parameters
- 520 ● Each parameter implies a predicate within the underlying query
- 521 ● Predicates for each supplied parameter are combined using with an implicit LOGICAL AND if
- 522 matchOnAnyParameter is unspecified or false. If it is specified as true then predicates for each
- 523 supplied parameters are combined using a LOGICAL OR
- 524 ● If an optional parameter is not supplied then its corresponding predicate MUST NOT be included
- 525 in the underlying query

526 3.7 Canonical Query: FindByIdAndType

527 The [canonical query FindByIdAndType](#) allows clients to find RegistryObjects based upon their the value of
528 their id and objectType attributes.

529 3.7.1 Parameter Summary

Parameter	Description	Data Type	Default Value	Cardinality
id	Matches rim:RegistryObject/@id. Allows use of “%” wildcard character to match multiple characters. Allows use of “?” wildcard character to match a single character.	string		0..1
objectType	Matches RegistryObjects whose objectType attribute matches the id of the ClassificationNode where rim:ClassificationNode/@path matches specified value	string		0..1

530 3.7.2 Query Semantics

- 531 ● The server MUST return the objects matching the query if the query is processed without any
- 532 exceptions

533 3.8 Canonical Query: FindAssociations

534 The [canonical query FindAssociations](#) query allows clients to find Associations that match the specified
535 criteria.

536 3.8.1 Parameter Summary

Parameter	Description	Data Type	Default Value	Cardinality
associationType	Matches Associations where	string		0..1

	rim:/Association/@associationType references a RegistryObject whose objectType attribute matches the id of the ClassificationNode where rim:ClassificationNode/@path matches specified value			
matchOnAnyParameter	If true then use logical OR between predicates for each parameter	boolean	false	0..1
sourceObjectId	Matches rim:Association/@sourceObject. Allows use of “%” wildcard character to match multiple characters. Allows use of “?” wildcard character to match a single character.	string		0..1
sourceObjectType	Matches Associations whose sourceObject attribute references a RegistryObject whose objectType attribute matches the id of the ClassificationNode where rim:ClassificationNode/@path matches specified value	string		0..1
targetObjectId	Matches rim:Association/@targetObject. Allows use of “%” wildcard character to match multiple characters. Allows use of “?” wildcard character to match a single character.	string		0..1
targetObjectType	Matches Associations whose targetObject attribute references a RegistryObject whose objectType attribute matches the id of the ClassificationNode where rim:ClassificationNode/@path matches specified value	string		0..1

537 3.8.2 Query Semantics

- 538 ● All parameters are optional
- 539 ● The server MUST return the objects matching the query if the query is processed without any
540 exceptions
- 541 ● Predicates for each supplied parameter are combined using with an implicit LOGICAL AND if
542 matchOnAnyParameter is unspecified or false. If it is specified as true then predicates for each
543 supplied parameters are combined using a LOGICAL OR

544 **3.9 Canonical Query: FindAssociatedObjects**

545 The canonical query [FindAssociatedObjects](#) query allows clients to find RegistryObjects that are
 546 associated with the specified RegistryObject and matched the specified criteria.

547 **3.9.1 Parameter Summary**

Parameter	Description	Data Type	Default Value	Cardinality
associationType	Matches associated RegistryObjects where the Association's associationsType, rim:/Association/@associationType references a RegistryObject whose objectType attribute matches the id of the ClassificationNode where rim:ClassificationNode/@path matches specified value	string		0..1
matchOnAnyParameter	If true then use logical OR between predicates for each parameter	boolean	false	0..1
sourceObjectId	Matches target RegistryObjects of Associations where the source RegistryObject's id matches rim:Association/@sourceObject. Allows use of “%” wildcard character to match multiple characters. Allows use of “?” wildcard character to match a single character.	string		0..1
sourceObjectType	Matches target RegistryObjects of Associations whose sourceObject attribute references a RegistryObject whose objectType attribute matches the id of the ClassificationNode where rim:ClassificationNode/@path matches specified value	string		0..1
targetObjectId	Matches source RegistryObjects of Associations where the target RegistryObject's id matches rim:Association/@targetObject. Allows use of “%” wildcard character to match multiple characters. Allows use of “?” wildcard character to match a single character.	string		0..1
targetObjectType	Matches source RegistryObjects of Associations whose targetObject attribute	string		0..1

	references a RegistryObject whose objectType attribute matches the id of the ClassificationNode where rim:ClassificationNode/@path matches specified value			
--	--	--	--	--

548 **3.9.2 Query Semantics**

- 549 ● All parameters are optional
- 550 ● The server MUST return the objects matching the query if the query is processed without any
551 exceptions
- 552 ● Either sourceObjectId or targetObjectId MUST be specified. If neither are specified then
553 InvalidQueryException fault MUST be returned
- 554 ● Both sourceObjectId and targetObjectId MUST NOT be specified. If both are specified then
555 InvalidQueryException fault MUST be returned
- 556 ● Predicates for each supplied parameter are combined using with an implicit LOGICAL AND if
557 matchOnAnyParameter is unspecified or false. If it is specified as true then predicates for each
558 supplied parameters are combined using a LOGICAL OR

559 **3.10 Canonical Query: GetClassificationSchemesById**

560 The canonical query [GetClassificationSchemesById](#) allows clients to fetch specified
561 ClassificationSchemes.

562 **3.10.1 Parameter Summary**

Parameter	Description	Data Type	Default Value	Cardinality
id	Matches rim:ClassificationScheme/@id. Allows use of “%” wildcard character to match multiple characters. Allows use of “?” wildcard character to match a single character.	string		0..1

563 **3.10.2 Query Semantics**

- 564 ● The server MUST return the objects matching the query if the query is processed without any
565 exceptions
- 566 ● The depth parameter of the QueryRequest may be used to pre-fetch the ClassificationNodes of
567 matches ClassificationSchemes

568 **3.11 Canonical Query: GetRegistryPackagesByMemberId**

569 The canonical query [GetRegistryPackagesByMemberId](#) allows clients to get the RegistryPackages that a
570 specified RegistryObject is a member of.

571 **3.11.1 Parameter Summary**

Parameter	Description	Data Type	Default Value	Cardinality
memberId	Matches RegistryPackages that have a RegistryObject as member where the RegistryObject's id rim:Registry/@id matches the specified value. Allows use of “%” wildcard character to match multiple characters. Allows use of “?” wildcard character to match a single character.	string		0..1

572 **3.11.2 Query Semantics**

- 573 ● The server MUST return the objects matching the query if the query is processed without any
574 exceptions

575 **3.12 Canonical Query: GetMembersByRegistryPackageId**

576 The canonical query [GetMembersByRegistryPackageId](#) allows clients to get the RegistryObjects that are
577 members of a specified RegistryPackage.

578 **3.12.1 Parameter Summary**

Parameter	Description	Data Type	Default Value	Cardinality
packageId	Matches RegistryObjects that are members of a RegistryPackage whose id rim:RegistryPackage/@id matches the specified value. Allows use of “%” wildcard character to match multiple characters. Allows use of “?” wildcard character to match a single character.	string		0..1

579 **3.12.2 Query Semantics**

- 580 ● The server MUST return the objects matching the query if the query is processed without any
581 exceptions

582 **3.13 Canonical Query: KeywordSearch**

583 The canonical query [KeyWordSearch](#) allows clients to find RegistryObjects and RepositoryItems that
584 contain text that matches keywords identified by specified search patterns.

585 **3.13.1 Canonical Indexes**

586 This query defines a set of canonical index names as defined by table below. Each index name is
587 associated with a particular type of information that it indexes. A server MUST index all information that is
588 defined by the canonical indexes below. A server MAY define additional indexes to index information not
589 specified by this section.

590

Index Name	Description
name.localizedString.value	Indexes the value of all localized string in all Name elements of all RegistryObjects
description.localizedString.value	Indexes the value of all localized string in all Description elements of all RegistryObjects
slot.name	Indexes the name of all slots on all RegistryObjects
slot.valueList.value	Indexes the value of all slots on all RegistryObjects
repositoryItem	Indexes the text of all text based repository items associated with ExtrinsicObjects
personName.firstName	Indexes the firstName attribute of PersonName elements in all Person objects
personName.middleName	Indexes the middleName attribute of PersonName elements in all Person objects
personName.lastName	Indexes the lastName attribute of PersonName elements in all Person objects
emailAddress.address	Indexes the address attribute of all EmailAddress objects
postalAddress.city	Indexes the city attribute of all PostalAddress elements contained within any RegistryObject
postalAddress.country	Indexes the country attribute of all PostalAddress elements contained within any RegistryObject
postalAddress.postalCode	Indexes the postalCode attribute of all PostalAddress elements contained within any RegistryObject
postalAddress.stateOrProvince	Indexes the stateOrProvince attribute of all PostalAddress elements contained within any RegistryObject
postalAddress.street	Indexes the street attribute of all PostalAddress elements contained within any RegistryObject

591

592

593 3.13.2 Parameter Summary

Parameter	Description	Data Type	Default Value	Cardinality
keywords	A space separated list of keywords to search for	string		1

594 3.13.3 Query Semantics

595 The value of the keywords parameter may consist of multiple terms where each term is separated
 596 by one or more spaces

597

598 Example: ebxml regrep

599 Semantics: Matches objects containing either “ebxml” or “regrep”

- 601 ● A term may be enclosed in double-quotes to include white space characters as a literal value

602

603 Example: “ebxml regrep”

604 Semantics: Matches objects containing “ebxml regrep”

- 606 ● Terms may be specified using wildcard characters where '*' matches one or more characters and
 607 “?” matches a single character (Issue: what about inconsistency with “%”??)

608

609 Example: eb?ml reg*

- 611 ● Terms may be combined using boolean operators “AND”, “OR” and “NOT”. Absence of a boolean
 612 operator between terms implies an implicit OR operator between them

613

614 Example: ebxml AND regrep

615 Semantics: Matches objects containing “ebxml” and “regrep”

616

617 Example: ebxml NOT regrep

618 Semantics: Matches objects containing “ebxml” and not containing “regrep”

619

620 Example: ebxml OR regrep

621 Semantics: Matches objects containing “ebxml” or “regrep”

622

623 Example: ebxml regrep

624 Semantics: Matches objects containing “ebxml” or “regrep”

- 626 ● Terms may be grouped together using “(“ at the beginning and “)” at the end of the group. Grouping
 627 allowing boolean operators to be applied to a group of terms as a whole and enables more flexible
 628 searches

629

- 630 Example: ebxml AND (registry OR regrep)
631 Semantics: Matches objects containing both “ebxml” and either “registry” or “regrep”
- 632 ● The server MUST return all RegistryObjects that contain indexed data matching the semantics of
633 the keywords parameter
 - 634 ● The server MUST return all ExtrinsicObjects that have a repository item that contains indexed
635 data matching the semantics of the keywords parameter

636 4 LifecycleManager Interface

637 The LifecycleManager interface allows a client to perform various lifecycle management operations on
638 RegistryObjects. These operations include submitting RegistryObjects to the server, updating
639 RegistryObjects in the server, creating new versions of RegistryObjects in the server and removing
640 RegistryObjects from the server.

641 A server MUST implement the LifecycleManager interface as an endpoint.

642 This specification does not specify explicit workflow support beyond basic CRUD operations described
643 here. Such workflow may be implemented using status attribute in combination with [subscription and
644 notification feature](#).

645 4.1 SubmitObjects Protocol

646 The SubmitObjects protocol allows a client to submit RegistryObjects to the server. It also allows a client
647 to completely replace existing RegistryObjects in the server.

648 A client initiates the SubmitObjects protocol by sending an SubmitObjectsRequest message to the
649 LifecycleManager endpoint.

650 The LifecycleManager sends an RegistryResponse back to the client as response.

651

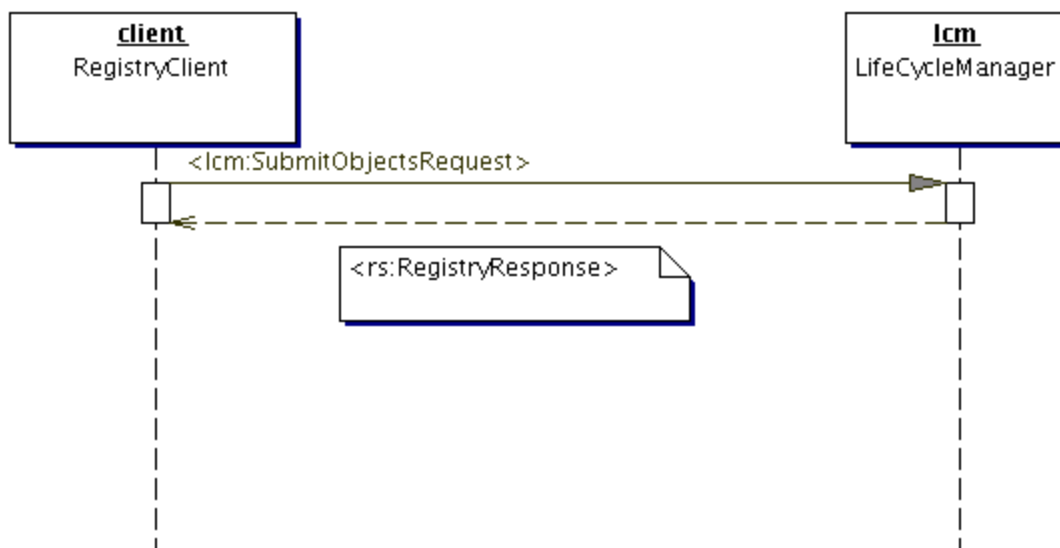


Illustration 2: SubmitObjects Protocol (replace??)

653 4.1.1 SubmitObjectsRequest

654 The SubmitObjectsRequest message is sent by a client to submit RegistryObjects to the server.

655 The server MUST apply any configured validation services for RegistryObjects in a
656 SubmitObjectsRequest before committing the request as described [here](#).

657 The server MUST apply any configured cataloging services for RegistryObjects in a
658 SubmitObjectsRequest before committing the request as described [here](#).

659

660 4.1.1.1 Syntax

```
661 <element name="SubmitObjectsRequest">
662   <annotation>
663     <documentation xml:lang="en">This SubmitObjects protocol allows a client to
664       submit RegistryObjects to the server. It also allows a client to completely
665       replace existing RegistryObjects in the server.</documentation>
666   </annotation>
667   <complexType>
668     <complexContent>
669       <extension base="rs:RegistryRequestType">
670         <sequence>
671           <element ref="rim:RegistryObjectList" minOccurs="0" maxOccurs="1"/>
672         </sequence>
673         <attribute name="mode" use="optional" default="createOrReplace">
674           <simpleType>
675             <restriction base="NCName">
676               <enumeration value="createOrReplace"/>
677               <enumeration value="createOnly"/>
678               <enumeration value="replaceOnly"/>
679             </restriction>
680           </simpleType>
681         </attribute>
682       </extension>
683     </complexContent>
684   </complexType>
685 </element>
```

686 4.1.1.2 Description

- 687 ● Element RegistryObjectList - This element specifies a set of RegistryObject instances that are
688 being submitted to the registry. The RegistryObjects in the list may be brand new objects being
689 submitted to the registry or they may be current objects already existing in the registry.
- 690 ● Attribute mode – **Do we need it, why not use just the default behavior??** This attribute specifies
691 the semantics for how the server should handle RegistryObjects being submitted when they
692 already exist in the server:
 - 693 ○ createOrReplace (default) - If an object does not exist, server MUST create it as a new object.
694 If an object already exists, server must replace the existing object with the submitted object
 - 695 ○ createOnly - If an object already exists, the server must return an Exception fault message
 - 696 ○ updateOnly - If an object does not exist, the server must return an Exception fault message

697 4.1.1.3 Returns

698 This request returns a [RegistryResponse](#).

699 4.1.1.4 Exceptions

700 No additional exceptions are defined beyond the exceptions common (where best to define these??) to all
701 requests.

702

703 4.1.2 RegistryResponse

704 TODO??

705 4.1.3 Unique ID Generation

706 Following needs to be updated for RegRep4??

707 A Submitter MUST supply the id attribute for submitted objects. If the id is not specified then the registry
708 MUST return an InvalidRequestException.

709 If the id and lid match the id and lid of an existing RegistryObject within the home registry, then the
710 registry MUST treat it as an Update action upon the existing RegistryObject.

711 If the id matches the id of an existing RegistryObject within the home registry but the lid does not match
712 that existing object's lid, then the registry MUST return an InvalidRequestException.

713 If the lid matches the lid of an existing RegistryObject within the home registry but the id does not match
714 that existing object's id, then the registry MUST create the newly submitted object as a new version of the
715 existing object.

716 If the Submitter supplies the id and it is a valid URN then the registry MUST honor the Submitter-supplied
717 id value and use it as the value of the id attribute of the object in the registry. If the id is not a valid URN
718 then the registry MUST treat it as a temporary id and replace it, and all references to it within the request,
719 with a registry generated universally unique id. A registry generated universally unique id value MUST
720 conform to the format of a URN that specifies a DCE 128 bit UUID as specified in [UUID]:

721 (e.g. urn:uuid:a2345678-1234-1234-123456789012)

722

723 4.1.4 ID Attribute And Object References

724 Following needs to be updated for RegRep4??

725 The id attribute of an object MAY be used by other objects to reference that object. Within a
726 SubmitObjectsRequest, the id attribute MAY be used to refer to an object within the same
727 SubmitObjectsRequest as well as to refer to an object within the registry. An object in the
728 SubmitObjectsRequest that needs to be referred to within the request document MAY be assigned an id
729 by the submitter so that it can be referenced within the request. The submitter MAY give the object a
730 valid URN, in which case the id is permanently assigned to the object within the registry. Alternatively, the
731 submitter MAY assign an arbitrary id that is not a valid URN as long as the id is a unique anyURI value
732 within the request document. In this case the id serves as a linkage mechanism within the request
733 document but MUST be replaced with a registry generated id upon submission.

734 When an object in a SubmitObjectsRequest needs to reference an object that is already in the registry,
735 the request MAY contain an ObjectRef whose id attribute is the id of the object in the registry. This id is by
736 definition a valid URN. An ObjectRef MAY be viewed as a proxy within the request for an object that is in
737 the registry.

738 4.1.5 Audit Trail Requirements

739 A server MUST create AuditableEvents *after* successfully processing a SubmitObjectsRequest and as
740 part of the same transaction as the request.

741 The server MUST create a single AuditableEvent object with eventType *Created* for all the
742 RegistryObjects created during processing of a SubmitObjectsRequest.

743 The server MUST create a single AuditableEvent object with eventType *Replaced* (*new: need to add to*
744 *schema??*) for all the RegistryObjects replaced during processing of a SubmitObjectsRequest.

745 4.1.6 Sample SubmitObjectsRequest

746 The following simplified example shows a SubmitObjectsRequest that submits a single Organization
747 object to the registry.

748

```
749 <lcm:SubmitObjectsRequest>  
750   <rim:RegistryObjectList>  
751     <rim:RegistryObject xsi:type="rim:OrganizationType" lid="{LOGICAL_ID}"  
752       id="{ID}" ...>  
753  
754     ...  
755  
756   </rim:RegistryObject>  
757 </rim:RegistryObjectList>  
758 </SubmitObjectsRequest>
```

759 4.2 The Update Objects Protocol

760 The UpdateObjectsRequest protocol allows a client to make partial updates to one or more
761 RegistryObjects that already exist in the server. This protocol allows *partial* update of RegistryObjects
762 rather than a *complete replacement*. A client should use the SubmitObjects protocol for complete
763 replacement of RegistryObjects.

764 A server MUST return <rs:InvalidRequestException> fault message if the client attempts to update the id,
765 lid or objectType attribute of a RegistryObject.

766

767 Issue: How to access control who can change status attribute??

768 Full object replacement is an error??

769 Document scope is RegistryObject??

770

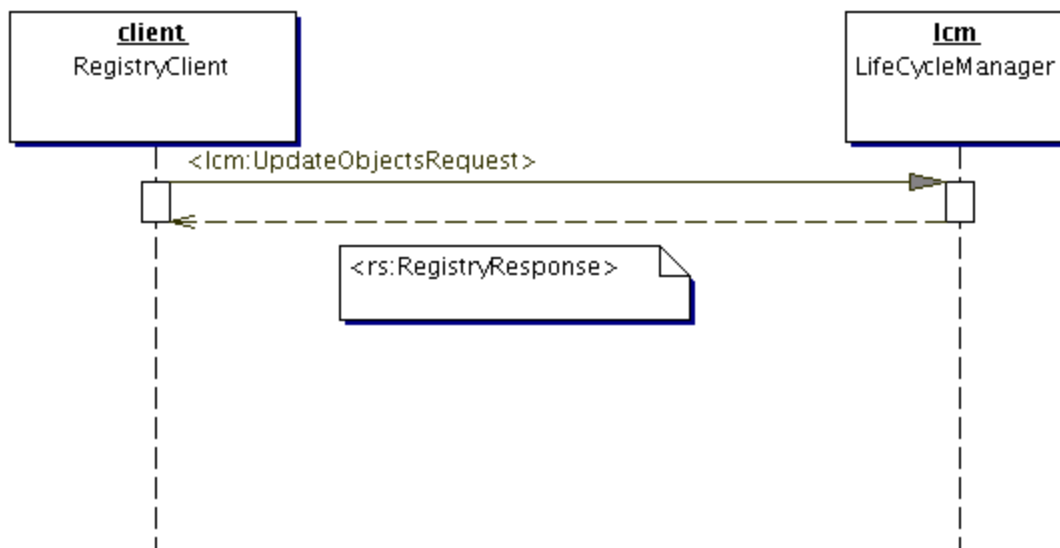


Illustration 3: UpdateObjects Protocol (replace??)

772 4.2.1 UpdateObjectsRequest

773 The UpdateObjectsRequest message is sent by a client to partially update existing RegistryObjects in the
 774 server. An UpdateObjectsRequest identifies a set of RegistryObjects as target objects to be updated by
 775 the request. It also specifies the details of the update action that modifies each target object. Update
 776 actions may insert a fragment within a target object, delete an existing fragment from a target object or
 777 update an existing fragment within the target object.

778 The server MUST apply any configured validation services for RegistryObjects in a
 779 SubmitObjectsRequest before committing the request as described [here](#).

780 The server MUST apply any configured cataloging services for RegistryObjects in a
 781 SubmitObjectsRequest before committing the request as described [here](#).

782 4.2.1.1 Syntax

```

783 <element name="UpdateObjectsRequest">
784   <annotation>
785     <documentation xml:lang="en">This UpdateObjectsRequest allows a client to
786       updates elements and attributes within RegistryObjects already existing in the
787       server.</documentation>
788   </annotation>
789   <complexType>
790     <complexContent>
791       <extension base="rs:RegistryRequestType">
792         <sequence>
793           <!-- QueryInvocation and ObjectRefList select objects to update -->
794           <element ref="rim:QueryInvocation" minOccurs="0" maxOccurs="1" />
795           <element ref="rim:ObjectRefList" minOccurs="0" maxOccurs="1" />
796
797           <!-- Specifies how to update selected objects -->
798           <element name="UpdateAction" type="tns:UpdateActionType" minOccurs="1"
799             maxOccurs="unbounded"/>
  
```



```
800     </sequence>
801     </extension>
802     </complexContent>
803     </complexType>
804 </element>
```

805 4.2.1.2 Description

- 806 ● Element QueryInvocation - Specifies a query to be invoked. A server MUST use all objects that
807 match the specified query in addition to any other objects identified by the ObjectRefList element
808 as targets of the update action.
- 809 ● Element ObjectRefList - Specifies a collection of references to existing RegistryObject instances
810 in the registry. A server MUST use all objects that are referenced by this element in addition to
811 any other objects identified by the QueryInvocation element as targets of the update action.
- 812 ● [Element UpdateAction](#) – Specifies the details of how to update the target objects.

813 4.2.1.3 Returns

814 This request returns a [RegistryResponse](#).

815 4.2.1.4 Exceptions

816 No additional exceptions are defined beyond the exceptions common **(where best to define these??)** to all
817 requests.

818 4.2.2 UpdateAction

819 An UpdateRequest contains one or more UpdateActions. Each UpdateObjectsRequest defines a specific
820 update action to be performed on each target object.

821 4.2.2.1 Syntax

```
822 <complexType name="UpdateActionType">
823   <annotation>
824     <documentation xml:lang="en">
825       </documentation>
826     </annotation>
827   <sequence>
828     <!-- Value for attribute or element -->
829     <element name="Value" type="rim:ValueType" minOccurs="1" maxOccurs="1"/>
830     <!--
831     Value of selector is an XPATH expression that uniquely identifies an
832     attribute
833     or an element within target documents.
834     -->
835     <element name="Selector" type="rim:QueryExpressionType" minOccurs="1"
836     maxOccurs="1"/>
837   </sequence>
838
839   <!--
840   Specifies whether to insert, update or delete a fragment from target
841   document.
```

```

842 -->
843 <attribute name="mode">
844   <simpleType>
845     <restriction base="NCName">
846       <enumeration value="insert"/>
847       <enumeration value="update"/>
848       <enumeration value="delete"/>
849     </restriction>
850   </simpleType>
851 </attribute>
852 </complexType>

```

853 4.2.2.2 Description

854 Element Selector – Is a QueryExpressionType that contains the expression that identifies a
855 fragment of the resource representation to be updated.

856
857 The value of this element MUST conform to the queryLanguage specified in the queryLanguage
858 attribute of the Selector. A resource MUST generate an InvalidExpressionException fault if the
859 expression is invalid. If the expression syntax is not valid with respect to the queryLanguage then
860 a resource SHOULD specify a fault detail of "InvalidExpressionSyntaxException". If the
861 expression value is not valid for the resource type then the resource SHOULD specify a fault
862 detail of "InvalidExpressionValueException".

863
864 A server MUST minimally support XPATH 1.0 as the queryLanguage for Selector element.

- 865 ● Element Value - This element contains the value to be written to the target object. If the mode
866 attribute is "Insert" or "Update" then this element MUST be present. If the mode is "Delete" then
867 this element MUST NOT be present. A resource MUST generate an InvalidRequestException
868 fault if it receives a message with a Value cardinality that is not valid for the Mode attribute.
- 869 ● Attribute mode – This attribute specifies the semantics for how the server should update target
870 objects:
 - 871 ○ insert - Indicates that the Value MUST be added to the target object. If the selector targets a
872 repeated element (maxOccurs > 1), the fragment MUST be added at the end. If the selector
873 targets a non-repeated element (maxOccurs = 1) that already exists, the resource MUST
874 generate a FragmentAlreadyExistsException fault. If the selector targets an existing item of a
875 repeated element, the Value MUST be added before the existing item.
 - 876 ○ update – Indicates that the fragment identified by selector MUST be replaced by the specified
877 Value in its place. If the selector resolves to nothing then this fragment element does not
878 result in any change to the target object.
 - 879 ○ delete - indicates that the fragment identified by selector MUST be deleted from the target
880 object if it is present.

881 4.2.3 Audit Trail Requirements

882 A server MUST create AuditableEvents *after* successfully processing a UpdateObjectsRequest and as
883 part of the same transaction as the request.

884 The server MUST create a single AuditableEvent object with eventType *Updated* for all the
885 RegistryObjects updated during processing of a SubmitObjectsRequest.

886 **4.2.4 Sample UpdateObjectsRequest**

887 **TBD??**

888

889 **4.3 RemoveObjects Protocol**

890 The Remove Objects protocol allows a client to remove or delete one or more RegistryObject instances
891 from the server.

892 A client initiates the RemoveObjects protocol by sending an RemoveObjectsRequest message to the
893 LifecycleManager endpoint.

894 The LifecycleManager sends an RegistryResponse back to the client as response.

895

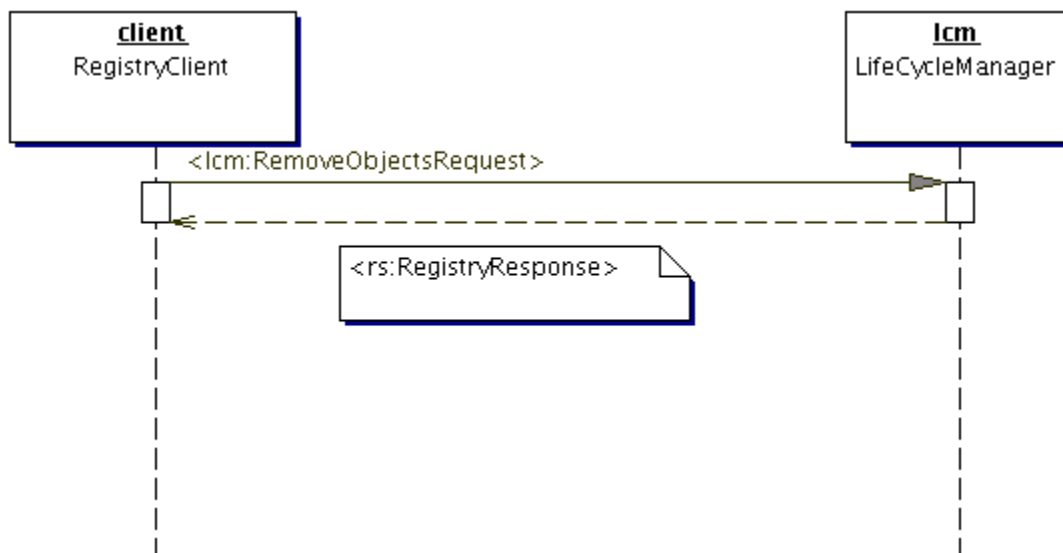


Illustration 4: RemoveObjects Protocol (replace??)

897 **4.3.1 RemoveObjectsRequest**

898 The RemoveObjectsRequest message is sent by a client to remove one or more existing RegistryObjects
899 from the server.

900 **4.3.1.1 Syntax**

```
901 <element name="RemoveObjectsRequest">  
902   <annotation>  
903     <documentation xml:lang="en">  
904       The ObjectRefList is the list of  
905       refs to the registry entrys being removed  
906     </documentation>  
907   </annotation>  
908   <complexType>
```

```
909     <complexContent>
910         <extension base="rs:RegistryRequestType">
911             <sequence>
912                 <element ref="rim:QueryInvocation" minOccurs="0" maxOccurs="1" />
913                 <element ref="rim:ObjectRefList" minOccurs="0" maxOccurs="1" />
914             </sequence>
915             <attribute name="deletionScope" default="urn:oasis:names:tc:ebxml-
916 regrep:DeletionScopeType>DeleteAll" type="rim:referenceURI" use="optional"/>
917         </extension>
918     </complexContent>
919 </complexType>
920 </element>
```

921

922 4.3.1.2 Description

- 923
- 924 ● Attribute `deletionScope` - This attribute specifies the scope of impact of the
925 RemoveObjectsRequest. The value of the `deletionScope` attribute MUST be a reference to a
926 ClassificationNode within the canonical DeletionScopeType ClassificationScheme as described in
927 ebRIM. A server MUST support the deletionScope types as defined by the canonical
928 DeletionScopeType ClassificationScheme. The canonical DeletionScopeType
929 ClassificationScheme may easily be extended by adding additional ClassificationNodes to it.

930 The following canonical ClassificationNodes are defined for the DeletionScopeType
931 ClassificationScheme:

- 932
- 933 ○ DeleteRepositoryItemOnly - Specifies that the registry MUST delete the RepositoryItem for
the specified ExtrinsicObjects but MUST NOT delete the specified ExtrinsicObjects
 - 934 ○ DeleteAll (default) - Specifies that the request MUST delete both the RegistryObject and the
935 RepositoryItem (if any) for the specified objects
 - 936 ● Element QueryInvocation - Specifies a query to be invoked. A server MUST remove all objects
937 that match the specified query in addition to any other objects identified by the ObjectRefList
938 element.
 - 939 ● Element ObjectRefList - Specifies a collection of references to existing RegistryObject instances
940 in the registry. A server MUST remove all objects that are referenced by this element in addition
941 to any other objects identified by the QueryInvocation element.

942 Issue: Should we not do referential integrity checks and allow dangling refs to achieve flexibility and
943 performance?? Should we add a canonical query FindDanglingReferences(type)??

944 4.3.1.3 Returns:

945 This request returns a [RegistryResponse](#).

946 4.3.1.4 Exceptions:

947 In addition to the exceptions common to all requests, the following exceptions MAY be returned:

- 948
- 949 ● UnresolvedReferenceException - Indicates that the requestor referenced an object within the
request that was not resolved during the processing of the request.

- 950 ● ReferencesExistException - Indicates that the requestor attempted to remove a RegistryObject
951 while references to it still exist. Note that it is valid to remove a RegistryObject and all
952 RegistryObjects that refer to it within the same request. In such cases the
953 ReferencesExistException MUST not be thrown.

954 **4.3.2 Audit Trail Requirements**

955 A server MUST create AuditableEvents *after* successfully processing a RemoveObjectsRequest and as
956 part of the same transaction as the request.

957 The server MUST create a single AuditableEvent object with eventType *Updated* for all the
958 RegistryObjects updated during processing of a SubmitObjectsRequest.

959 **4.3.3 Sample UpdateObjectsRequest**

960 **TBD??**

961 5 Validator Interface

962 The Validator interface allows a client to validate objects already in the server. A server MUST implement
963 the Validator interface as an endpoint. The Validator interface validates objects using [Validator Plugins](#)
964 specific to the type of object being validated.

965 5.1 ValidateObjects Protocol

966 A client validates RegistryObjects residing in the server using the *ValidateObjects* protocol supported by
967 the validateObjects operation of the Validator interface.

968 A client initiates the ValidateObjects protocol by sending an ValidateObjectsRequest message to the
969 Validator endpoint.

970 The Validator endpoint sends an ValidateObjectsResponse back to the client as response. The
971 ValidateObjectsResponse contains information on whether the objects were valid and if invalid objects
972 were found it includes any validation errors that were encountered.

973

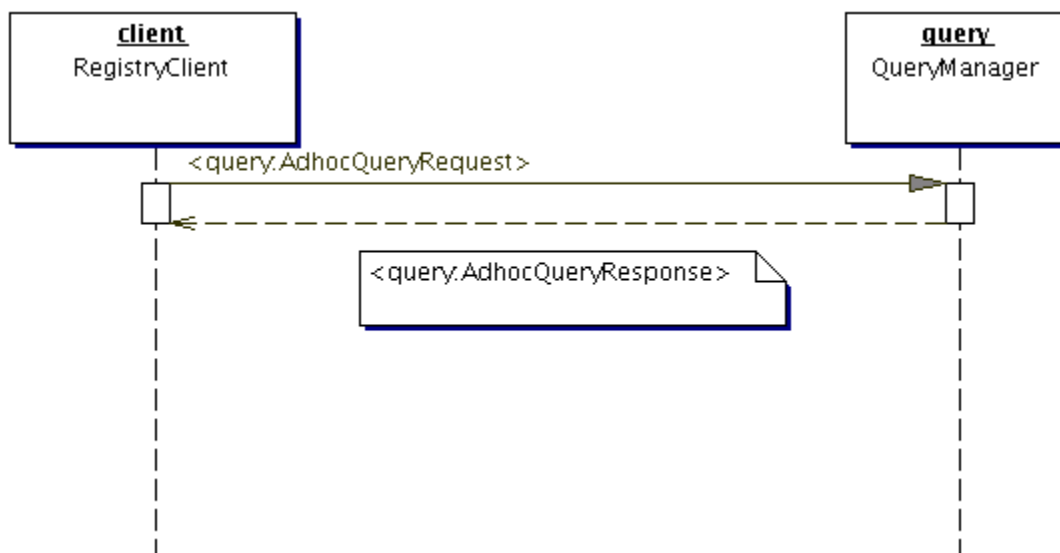


Illustration 5: ValidateObjects Protocol **Need to update figure??**

975

976 5.1.1 ValidateObjectsRequest

977 The ValidateObjectsRequest message is sent by client to the Validator interface to validate objects that
978 are already resident in the server.

979 5.1.1.1 Syntax

980 `<element name="ValidateObjectsRequest">`

```

981     <annotation>
982         <documentation xml:lang="en">Request to validate specified
983 objects.</documentation>
984     </annotation>
985     <complexType>
986         <complexContent>
987             <extension base="rs:RegistryRequestType">
988                 <sequence>
989                     <!-- QueryInvocation and ObjectRefList select objects to validate
990 -->
991                     <element ref="rim:QueryInvocation" minOccurs="0" maxOccurs="1" />
992                     <element ref="rim:ObjectRefList" minOccurs="0" maxOccurs="1" />
993                     <element name="OriginalObjects" type="rim:RegistryObjectListType"/>
994                 </sequence>
995                 <!-- The Invocation Control File (optional). -->
996                 <element maxOccurs="unbounded" minOccurs="0"
997 name="InvocationControlFile" type="rim:ExtrinsicObjectType"/>
998             </extension>
999         </complexContent>
1000     </complexType>
1001 </element>
1002

```

1003 5.1.1.2 Example

1004 TODO??

1005 5.1.1.3 Description

1006 **Issue: Editorial - Need to list items in order of schema, schema should be in alpha order??**

- 1007 ● Element QueryInvocation - Specifies a query to be invoked. A server MUST validate all objects
1008 that match the specified query.
- 1009 ● Element ObjectRefList - Specifies a collection of references to existing RegistryObject instances
1010 in the registry. A server MUST validate all objects that are referenced by this element.
- 1011 ● Element OriginalObjects - Specifies a collection of RegistryObject instances. A server MUST
1012 validate all objects that are contained in this element.
- 1013 ● Element InvocationControlFile – Specifies an ExtrinsicObject that is used to control the validation
1014 process in a type specific manner. See [Canonical XML Validator plugin](#) for an example.

1015 5.1.1.4 Response

1016 This request returns [ValidateObjectsResponse](#) as response.

1017 5.1.1.5 Exceptions

1018 **Issue: Need to define more fine-grained exception hierarchy??**

1019 In addition to the exceptions common to all requests defined in ??, the following exceptions MAY be returned:

- 1020 ● ValidationException: signifies that an exception was encountered during the validateObjects operation

1021 **5.1.2 ValidateObjectsResponse**

1022 Currently ValidateObjectsResponse is a simple extension to RegistryResponseType and does not define
1023 additional attributes or elements. Issue: **How to return all ValidationExceptions in a single response??**

1024

1025 6 Cataloger Interface

1026 The Cataloger interface allows a client to catalog or index objects already in the server. A server MUST
1027 implement the Cataloger interface as an endpoint. The Cataloger interface catalogs objects using
1028 [Cataloger Plugins](#) specific to the type of object being validated.

1029 6.1 CatalogObjects Protocol

1030 A client catalogs RegistryObjects residing in the server using the *CatalogObjects* protocol supported by
1031 the catalogObjects operation of the Cataloger interface.

1032 A client initiates the CatalogObjects protocol by sending an CatalogObjectsRequest message to the
1033 Cataloger endpoint.

1034 The Cataloger endpoint sends a CatalogObjectsResponse back to the client as response.

1035

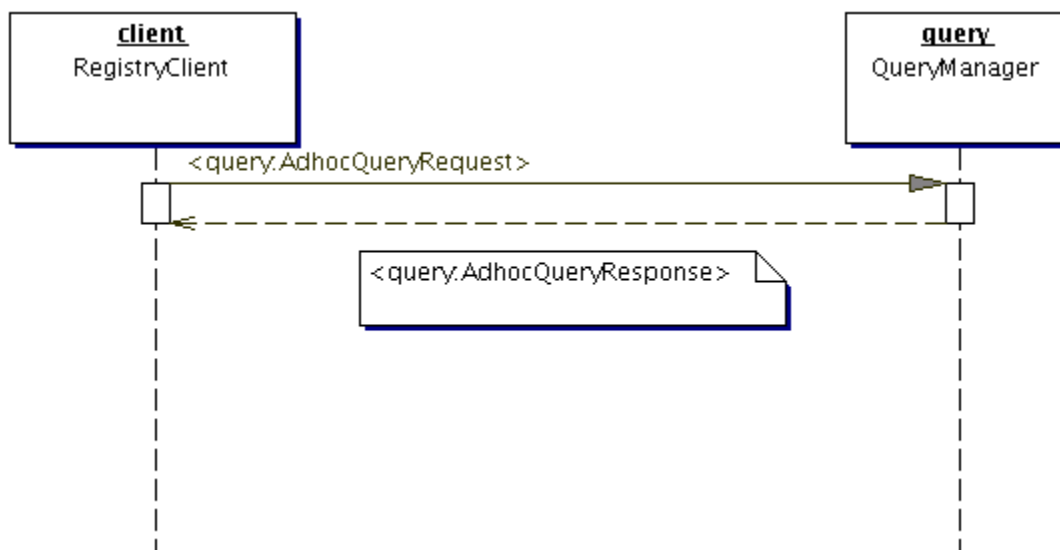


Illustration 6: CatalogObjects Protocol **Need to update figure??**

1037

1038 6.1.1 CatalogObjectsRequest

1039 The CatalogObjectsRequest message is sent by client to the Cataloger interface to catalog objects that
1040 are already resident in the server.

1041 6.1.1.1 Syntax

```
1042 <element name="CatalogObjectsRequest">  
1043 <annotation>
```

```

1044     <documentation xml:lang="en">Request to catalog specified
1045 objects.</documentation>
1046     </annotation>
1047     <complexType>
1048         <complexContent>
1049             <extension base="rs:RegistryRequestType">
1050                 <sequence>
1051                     <!-- QueryInvocation and ObjectRefList select objects to catalog
1052 -->
1053                     <element ref="rim:QueryInvocation" minOccurs="0" maxOccurs="1" />
1054                     <element ref="rim:ObjectRefList" minOccurs="0" maxOccurs="1" />
1055                     <element name="OriginalObjects" type="rim:RegistryObjectListType"/>
1056                 </sequence>
1057                 <!-- The Invocation Control File (optional). -->
1058                 <element maxOccurs="unbounded" minOccurs="0"
1059 name="InvocationControlFile" type="rim:ExtrinsicObjectType"/>
1060             </extension>
1061         </complexContent>
1062     </complexType>
1063 </element>
1064

```

1065 **6.1.1.2 Example**

1066 TODO??

1067 **6.1.1.3 Description**

- 1068 ● Element QueryInvocation - Specifies a query to be invoked. A server MUST catalog all objects
1069 that match the specified query.
- 1070 ● Element ObjectRefList - Specifies a collection of references to existing RegistryObject instances
1071 in the registry. A server MUST catalog all objects that are referenced by this element.
- 1072 ● Element OriginalObjects - Specifies a collection of RegistryObject instances. A server MUST
1073 catalog all objects that are contained in this element.
- 1074 ● Element InvocationControlFile – Specifies an ExtrinsicObject that is used to control the validation
1075 process in a type specific manner. See [Canonical XML Catalogor plugin](#) for an example.

1076 **6.1.1.4 Response**

1077 This request returns [CatalogObjectsResponse](#) as response.

1078 **6.1.1.5 Exceptions**

1079 In addition to the exceptions common to all requests defined in ??, the following exceptions MAY be returned:

- 1080 ● CatalogException: signifies that an exception was encountered during the catalogObjects operation

1081 **6.1.2 CatalogObjectsResponse**

1082 The CatalogObjectsResponse message is sent by the Cataloger endpoint in response to an
1083 CatalogObjectsRequest.

1084 6.1.2.1 Syntax

```
1085 <element name="CatalogObjectsResponse">
1086   <annotation>
1087     <documentation xml:lang="en">Response to request to catalog specified
1088 objects.</documentation>
1089   </annotation>
1090   <complexType>
1091     <complexContent>
1092       <extension base="rs:RegistryResponseType">
1093         <sequence>
1094           <element name="CatalogedObjects"
1095 type="rim:RegistryObjectListType"/>
1096         </sequence>
1097       </extension>
1098     </complexContent>
1099   </complexType>
1100 </element>
```

1101 6.1.2.2 Example

1102 TODO??

1103 6.1.2.3 Description

1104 In addition to elements and attributes defined by RegistryResponseType the following are defined:

- 1105 ● Element CatalogedObjects – Contains the RegistryObjects that are produced as output of the
1106 catalogObjects operation. Typically this list contains the objects that were input to the
1107 catalogObjects operation, as well as new objects that were the output of the catalogObjects
1108 operation. The input objects MAY be modified by the cataloger as a result of the catalogObjects
1109 operation.

1110 7 Server Plugin SPI

1111 Deployments of the server MAY extend the core functionality of the server by using function-specific
1112 software modules called plugins. A plugin extends the server by adding additional functionality to it. A
1113 plugin MUST conform to standard interfaces called Service Provider Interfaces (SPI) as defined by this
1114 specification.

1115 This chapter specifies the Service Provider Interfaces (SPI) that defines the standard interface for various
1116 types of server plugins. The interfaces are described in form of WSDL specification. A server may
1117 implement these interfaces as external web services invoked by the server using SOAP or as plugin
1118 modules that share the same process as the server and are invoked by local function calls.

1119 Examples of types of server plugins include, but are not limited to query plugin, validator plugin and
1120 cataloger plugin.

1121 This specification does not define how a plugin is configured within a server. Nor does it define whether or
1122 how, plugin configuration functionality is made available by the server to clients.

1123 7.1 Query Plugins

1124 Query plugins allow a server to implement support for a parameterized query as a plugin. Since query
1125 plugins are software modules, they able to handle highly specialized query semantics that cannot be
1126 expressed in most query languages. A specific instance of a query plugin is designed and configured to
1127 handle a specific parameterized query.

1128 7.1.1 Query Plugin Interface

1129 A Query plugin implements the [QueryManager interface](#). A QueryManager endpoint MUST delegate an
1130 executeQuery operation to a Query plugin if a Query plugin has been configured for the requested
1131 parameterized query. A Query plugin MUST process the query and return a QueryResponse or fault
1132 message to the QueryManager. The QueryManager MUST then deliver that response to the client.

1133 7.2 Validator Plugins

1134 Validator plugins allow a server to validate objects being submitted during the processing of a
1135 SubmitObjectsRequest or being validate during the processing of a ValidateObjectsRequest.

1136 A specific instance of a Validator plugin is designed and configured to validate a specific type of object.
1137 For example, The canonical XML Validator plugin is designed and configured to validate XML Objects
1138 using Schematron documents as InvocationControlFile.

1139 7.2.1 Validator Plugin Interface

1140 A Validator plugin implements the [Validator interface](#). The server's Validator endpoint MUST delegate a
1141 validateObjects operation to any number of Validator plugins using the following algorithm:

- 1142 ● The server selects the RegistryObjects that are the target of the validateObjects operations using
1143 the <rim:QueryInvocation> and <rim:ObjectRefList> elements. Any objects specified by the
1144 OriginalObjects element MUST be ignored by the server.

- 1145 ● The server partitions the set of target objects into multiple sets based upon the objectType
1146 attribute value for the target objects
- 1147 ● The server determines whether there is a Validator plugin configured for each objectType for
1148 which there is a set of target objects
- 1149 ● For each set of target objects that share a common objectType and for which there is a configured
1150 Validator plugin, the server MUST invoke the Validator plugin. The Validator plugin invocation
1151 MUST specify the target objects for that set using the OriginalObjects element. The server MUST
1152 NOT specify <rim:QueryInvocation> and <rim:ObjectRefList> elements when invoking
1153 validateObjects operation on a Validator plugin
- 1154 ● Each Validator plugin MUST process the ValidateObjectsRequest and return a
1155 ValidateObjectsResponse or fault message to the server's Validator endpoint.
- 1156 ● The server's Validator endpoint MUST then combine the results of the individual
1157 ValidateObjectsRequest to Validator plugins into a single unified ValidateObjectsResponse and
1158 return it to the client.

1159 **7.2.2 Canonical XML Validator Plugin**

1160 The canonical XML Validator plugin is a validator plugin that validates XML content using a Schematron
1161 file as InvocationControlFile. The Schematron file specifies validation rules using [Schematron] language
1162 to validate XML content. The server may configure the canonical XML Validator plugin such that it is
1163 invoked with an appropriate schematron file as InvocationControlFile based upon the objectType of the
1164 object being validated.

1165 A server MUST implement the canonical XML Validator plugin.

1166 **7.3 Cataloger Plugins**

1167 Cataloger plugins allow a server to catalog objects being submitted during the processing of a
1168 SubmitObjectsRequest or being cataloged during the processing of a CatalogObjectsRequest.

1169 A specific instance of a Cataloger plugin is designed and configured to catalog a specific type of object.
1170 For example, The canonical XML Cataloger plugin is designed and configured to catalog XML Objects
1171 using XSLT documents as InvocationControlFile.

1172 **7.3.1 Cataloger Plugin Interface**

1173 A Cataloger plugin implements the [Cataloger interface](#). The server's Cataloger endpoint MUST delegate a
1174 catalogObjects operation to any number of Cataloger plugins using the following algorithm:

- 1175 ● The server selects the RegistryObjects that are the target of the catalogObjects operations using
1176 the <rim:QueryInvocation> and <rim:ObjectRefList> elements. Any objects specified by the
1177 OriginalObjects element MUST be ignored by the server.
- 1178 ● The server partitions the set of target objects into multiple sets based upon the objectType
1179 attribute value for the target objects
- 1180 ● The server determines whether there is a Cataloger plugin configured for each objectType for
1181 which there is a set of target objects

- 1182 ● For each set of target objects that share a common objectType and for which there is a configured
1183 Cataloger plugin, the server MUST invoke the Cataloger plugin. The Cataloger plugin invocation
1184 MUST specify the target objects for that set using the OriginalObjects element. The server MUST
1185 NOT specify <rim:QueryInvocation> and <rim:ObjectRefList> elements when invoking
1186 catalogObjects operation on a Cataloger plugin
- 1187 ● Each Cataloger plugin MUST process the CatalogObjectsRequest and return a
1188 CatalogObjectsResponse or fault message to the server's Cataloger endpoint.
- 1189 ● The server's Cataloger endpoint MUST then combine the results of the individual
1190 CatalogObjectsRequest to Cataloger plugins and commit these objects as part of the transaction
1191 associated with the request. It MUST then combine the individual CatalogObjectsResponse
1192 messages into a single unified CatalogObjectsResponse and return it to the client.

1193 **7.3.2 Canonical XML Cataloger Plugin**

- 1194 The canonical XML Cataloger plugin is a Cataloger plugin that catalogs XML content using an XSLT file
1195 as InvocationControlFile. The XSLT file specifies transformations rules using [XSLT] language to catalog
1196 XML content. The server may configure the canonical XML Cataloger plugin such that it is invoked with an
1197 appropriate XSLT file as InvocationControlFile based upon the objectType of the object being validated.
- 1198 A server MUST implement the canonical XML Cataloger plugin.

1199 8 Subscription and Notification

1200 Subscription and Notification is a normative though optional feature.

1201 A client MAY subscribe to events that transpire in the server by creating a Subscription. A server
1202 supporting Subscription and Notification feature MUST deliver a Notification to the subscriber when an
1203 event transpires that matches the event selection criteria specified by the client.

1204 8.1 Server Events

1205 Activities within the server result in events. [ebRIM] defines the AuditableEvent class, instances of which
1206 represent server events. Typically, a server creates AuditableEvent instances during the processing of
1207 client requests.

1208 8.1.1 Pruning of Events

1209 A server MAY periodically prune AuditableEvents in order to manage its resources. It is up to the server
1210 when such pruning occurs. It is up to the server to determine when undelivered events are purged. A
1211 server SHOULD perform such pruning by removing the older AuditableEvents first. **What about CREATED**
1212 **events which establish owner for RegistryObject??**

1213 8.2 Notifications

1214 A Notification message is used by the server to notify clients of events they have subscribed to. A
1215 Notification contains the RegistryObjects, or references to the RegistryObjects, that are affected by the
1216 event for which the Notification is being sent

1217 Details for the Notification element are defined in [ebRIM].

1218 8.3 Creating a Subscription

1219 A client MAY create a subscription within a server if it wishes the server to send it a Notification when a
1220 specific type of event transpires. A client creates a subscription by submitting a <rim:Subscription>
1221 instance to the server using the standard [SubmitObjects protocol](#). If a Subscription is submitted to the
1222 server that does not support event notification then the server MUST return an
1223 UnsupportedCapabilityException.

1224 Submission of a Subscription object follows the same rules as submission of any other RegistryObject.
1225 Details for the Subscription element are defined in [ebRIM].

1226 8.3.1 Subscription Authorization

1227 A deployment MAY use custom Access Control Policies to decide which users are authorized to create a
1228 subscription and to what events. A server MUST return an AuthorizationException in the event that an
1229 unauthorized user submits a Subscription to a server.

1230 **8.3.2 Subscription Quotas**

1231 A server MAY use server specific policies to decide an upper limit on the number of Subscriptions a user
1232 is allowed to create. A server MUST return a QuotaExceededException in the event that an authorized
1233 user submits more Subscriptions than allowed by their server-specific quota.

1234 **8.3.3 Subscription Expiration**

1235 Each subscription defines a startTime and an endTime attribute which determines the period within
1236 which a Subscription is active. Outside the bounds of the active period, a Subscription MAY exist in an
1237 expired state within the server. A server MAY remove an expired Subscription at any time.

1238 A Registry MUST NOT deliver notifications for an event to an expired Subscriptions. An expired
1239 Subscription MAY be renewed by updating the startTime and / or endTime for the Subscription using the
1240 [UpdateObjects protocol](#).

1241 **8.3.4 Event Selection**

1242 A client MUST specify a Selector element within the Subscription to specify its criteria for selecting events
1243 of interest. The Selector element is of type <rim:QueryInvocationType> and specifies a parameterized
1244 query to be invoked with specified query parameters.

1245 A server MUST process AuditableEvents and determine which Subscriptions match the event using the
1246 following algorithm:

- 1247 ● **TODO??**

1248

1249 **8.4 Event Delivery**

1250 A client MUST specify a DeliveryInfo element within the Subscription to specify how the server should
1251 deliver events matching the subscription to the client. The DeliveryInfo element includes a NotifyTo element
1252 which specifies an EndPoint Reference (EPR) as defined by [WS-Addressing]. The NotifyTo element
1253 contains a <wsa:Address> element which contains a URI to the endpoint.

1254 Details for the DeliveryInfo element are defined in [ebRIM].

1255 **8.4.1 Delivery Mode**

1256 A client MAY specify a mode attribute in DeliveryInfo element of a Subscription. The mode attribute
1257 specifies how the client wishes to be notified of events. This specification defines the following modes and
1258 allows a server to support additional modes:

- 1259 ● urn:oasis:names:tc:ebxml-regrep:rim:DeliveryInfo:mode:push - This mode specifies that the client
1260 wishes the server to include complete RegistryObjects within the Notifications.
- 1261 ● urn:oasis:names:tc:ebxml-regrep:rim:DeliveryInfo:mode:push-pull - This mode specifies that the
1262 client wishes the server to include only ObjectRefs to RegistryObjects within the Notifications. The
1263 client MAY then “pull” the actual RegistryObjects from the server if needed using the [Query](#)
1264 [protocol](#).

- 1265 ● urn:oasis:names:tc:ebxml-regrep:rim:DeliveryInfo:mode:pull - This mode specifies that the client
1266 wishes the server to not deliver any Notification to it. Instead the client MAY “pull” AuditableEvents
1267 of interest using the [Query protocol](#).

1268 **8.4.2 Delivery to NotificationListener Web Service**

1269 If the URI in the <wsa:Address> element is a URL that uses the http protocol then the server MUST use
1270 this URL as the web service endpoint to deliver the Notification to. The target web service in this case
1271 MUST implement the NotificationListener interface.

1272 **8.4.3 Delivery to Email Address**

1273 If the URI in the <wsa:Address> element is a URL that uses the mailto protocol then the server MUST use
1274 this URL as the email address to deliver the Notification to via email. This specification does not define
1275 how a server is configured to send Notifications via email.

1276 **8.4.3.1 Processing Email Notification Via XSLT**

1277 A client MAY specify an XSLT style sheet within a DeliveryInfo element to process a Notification prior to it
1278 being delivered to an email address. The XSLT style sheet MAY be specified using a Slot in DeliveryInfo
1279 element where the Slot's name is “urn:oasis:names:tc:ebxml-regrep:rim:DeliveryInfo:email:xslt” and the
1280 Slots value is the id of an ExtrinsicObject whose repository item is the XSLT. The ExtrinsicObject and
1281 repository item MUST be submitted prior to or at the same time as the Subscription.

1282 **8.5 NotificationListener Interface**

1283 The NotificationListener interface allows a client to receive Notifications from the server for their
1284 Subscriptions. A client MUST implement the NotificationListener interface as an endpoint if they wish to
1285 receive Notifications via SOAP.

1286 **8.6 Notification Protocol**

1287 A server sends a Notification to a client using the *Notification* protocol supported by the onNotification
1288 operation of the NotificationListener interface.

1289 A server initiates the Notification protocol by sending an Notification message to the NotificationListener
1290 endpoint registered within the Subscription for which the Notification is being delivered.

1291 The onNotification operation does not send a response back to the server.

1292

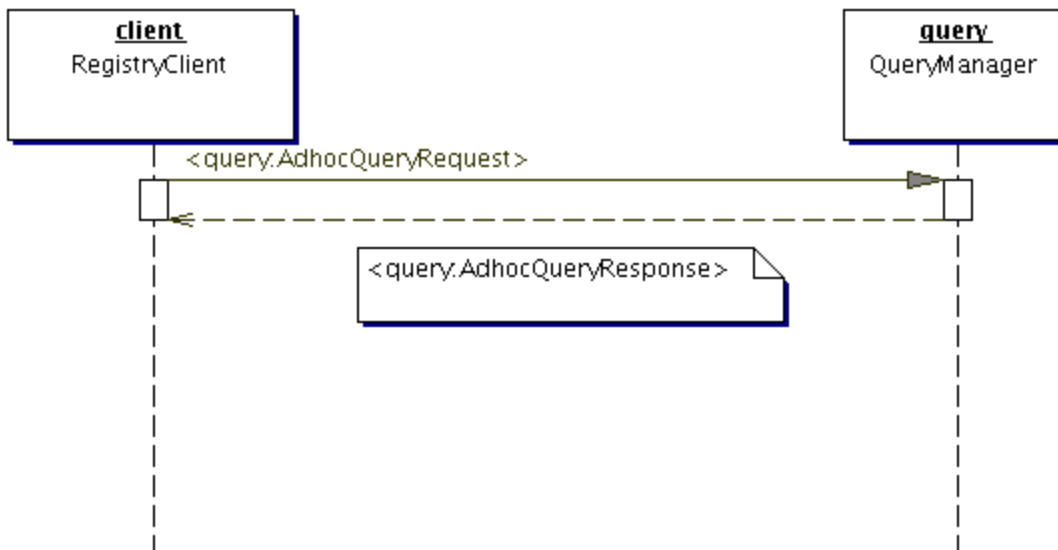


Illustration 7: CatalogObjects Protocol **Need to update figure??**

1294

1295 8.6.1 CatalogObjectsRequest

1296 The CatalogObjectsRequest message is sent by client to the Cataloger interface to catalog objects that
1297 are already resident in the server.

1298 8.6.1.1 Syntax

```

1299 <element name="CatalogObjectsRequest">
1300   <annotation>
1301     <documentation xml:lang="en">Request to catalog specified
1302 objects.</documentation>
1303   </annotation>
1304   <complexType>
1305     <complexContent>
1306       <extension base="rs:RegistryRequestType">
1307         <sequence>
1308           <!-- QueryInvocation and ObjectRefList select objects to catalog
1309 -->
1310           <element ref="rim:QueryInvocation" minOccurs="0" maxOccurs="1" />
1311           <element ref="rim:ObjectRefList" minOccurs="0" maxOccurs="1" />
1312           <element name="OriginalObjects" type="rim:RegistryObjectListType"/>
1313         >
1314           <!-- The Invocation Control File (optional). -->
1315           <element maxOccurs="unbounded" minOccurs="0"
1316 name="InvocationControlFile" type="rim:ExtrinsicObjectType"/>
1317         </sequence>
1318       </extension>
1319     </complexContent>
1320   </complexType>
1321 </element>
  
```

1322 8.6.1.2 Example

1323 TODO??

1324 8.6.1.3 Description

- 1325 ● Element QueryInvocation - Specifies a query to be invoked. A server MUST catalog all objects
1326 that match the specified query.
- 1327 ● Element ObjectRefList - Specifies a collection of references to existing RegistryObject instances
1328 in the registry. A server MUST catalog all objects that are referenced by this element.
- 1329 ● Element OriginalObjects - Specifies a collection of RegistryObject instances. A server MUST
1330 catalog all objects that are contained in this element.
- 1331 ● Element InvocationControlFile – Specifies an ExtrinsicObject that is used to control the validation
1332 process in a type specific manner. See [Canonical XML Catalogor plugin](#) for an example.

1333 8.6.1.4 Response

1334 This request returns [CatalogObjectsResponse](#) as response.

1335 8.6.1.5 Exceptions

1336 In addition to the exceptions common to all requests defined in ??, the following exceptions MAY be returned:

- 1337 ● CatalogException: signifies that an exception was encountered during the catalogObjects operation

1338 8.6.2 CatalogObjectsResponse

1339 The CatalogObjectsResponse message is sent by the Cataloger endpoint in response to an
1340 CatalogObjectsRequest.

1341 8.6.2.1 Syntax

```
1342 <element name="CatalogObjectsResponse">  
1343   <annotation>  
1344     <documentation xml:lang="en">Response to request to catalog specified  
1345 objects.</documentation>  
1346   </annotation>  
1347   <complexType>  
1348     <complexContent>  
1349       <extension base="rs:RegistryResponseType">  
1350         <sequence>  
1351           <element name="CatalogedObjects"  
1352 type="rim:RegistryObjectListType"/>  
1353         </sequence>  
1354       </extension>  
1355     </complexContent>  
1356   </complexType>  
1357 </element>
```

1358 **8.6.2.2 Example**

1359 TODO??

1360 **8.6.2.3 Description**

1361 In addition to elements and attributes defined by RegistryResponseType the following are defined:

- 1362 ● Element CatalogedObjects – Contains the RegistryObjects that are produced as output of the
1363 catalogObjects operation. Typically this list contains the objects that were input to the
1364 catalogObjects operation, as well as new objects that were the output of the catalogObjects
1365 operation. The input objects MAY be modified by the cataloger as a result of the catalogObjects
1366 operation.

1367 **8.7 Deleting a Subscription**

1368 A client MAY terminate a Subscription with a registry if it no longer wishes to be notified of events related
1369 to that Subscription. A client terminates a Subscription by deleting the corresponding Subscription object
1370 using the standard [RemoveObjects protocol](#).

1371 Removal of a Subscription object follows the same rules as removal of any other RegistryObject.

1372

9 Cooperating Registries

1373

Canonical objects should have a spec defined home attribute to a virtual "spec" RegRep instance.

1374

Issue: Should we remove Federation Infomodel classes and instead rely on local queries against remoted

1375

registers?? Decisions here may impact Query Protocol??

1376 **10 Security Features**

1377 No longer defining registry as identity provider or authentication authority??.

1378 Support pluggable authentication??

1379

1380 11 Native Language Support (NLS)

1381 This chapter describes the Native Languages Support (NLS) features of ebXML Registry.

1382 11.1 Terminology

1383 The following terms are used in NLS.

NLS Term	Description
Coded Character Set (CCS)	CCS is a mapping from a set of abstract characters to a set of integers. [RFC 2130]. Examples of CCS are ISO-10646, US-ASCII, ISO-8859-1, and so on.
Character Encoding Scheme (CES)	CES is a mapping from a CCS (or several) to a set of octets. [RFC 2130]. Examples of CES are ISO-2022, UTF-8.
Character Set (charset)	<ul style="list-style-type: none">• charset is a set of rules for mapping from a sequence of octets to a sequence of characters.[RFC 2277],[RFC 2278]. Examples of character set are ISO-2022-JP, EUC-KR.• A list of registered character sets can be found at [IANA].

1384

1385 11.2 NLS and Registry Protocol Messages

1386 For the accurate processing of data in both registry client and registry services, it is essential for the
1387 recipient of a protocol message to know the character set being used by it.

1388 A Registry Client SHOULD specify charset parameter in MIME header when they specify text/xml as
1389 Content-Type. A server MUST specify charset parameter in MIME header when they specify text/xml as
1390 Content-Type.

1391 The following is an example of specifying the character set in the MIME header.

1392

1393

1394

```
Content-Type: text/xml; charset=ISO-2022-JP
```

1395

1396 If a registry receives a protocol message with the charset parameter omitted then it MUST use the default
1397 charset value of "us-ascii" as defined in [RFC 3023].

1398 Also, when an application/xml entity is used, the charset parameter is optional, and registry client and
1399 registry services MUST follow the requirements in Section 4.3.3 of [REC-XML] which directly address this
1400 contingency.

1401 If another Content-Type is used, then usage of charset MUST follow [RFC 3023].

1402 **11.3 NLS Support in RegistryObjects**

1403 The information model XML Schema [RR-RIM-XSD] defines the <rim:InternationalStringType> for defining
1404 elements that contains a locale sensitive string value.

1405

```
1406 <complexType name="InternationalStringType">  
1407   <sequence maxOccurs="unbounded" minOccurs="0">  
1408     <element ref="tns:LocalizedString"/>  
1409   </sequence>  
1410 </complexType>
```

1411

1412 An InternationalStringType may contain zero or more LocalizedStrings within it where each
1413 LocalizedString contain a string value is a specified local language and character set.

1414

```
1415 <complexType name="LocalizedStringType">  
1416   <attribute ref="xml:lang" default="en-US"/>  
1417   <attribute default="UTF-8" name="charset"/>  
1418   <attribute name="value" type="tns:FreeFormText" use="required"/>  
1419 </complexType>
```

1420

1421 Examples of such attributes are the “name” and “description” attributes of the RegistryObject class defined
1422 by [ebRIM] as shown below.

```
1423 <complexType name="InternationalStringType">  
1424   <sequence maxOccurs="unbounded" minOccurs="0">  
1425     <element ref="tns:LocalizedString"/>  
1426   </sequence>  
1427 </complexType>  
1428 <element name="InternationalString" type="tns:InternationalStringType"/>  
1429 <element name="Name" type="tns:InternationalStringType"/>  
1430 <element name="Description" type="tns:InternationalStringType"/>  
1431  
1432 <complexType name="LocalizedStringType">  
1433   <attribute ref="xml:lang" default="en-US"/>  
1434   <!--attribute name = "lang" default = "en-US" form = "qualified" type =  
1435 "language"/-->  
1436   <attribute default="UTF-8" name="charset"/>  
1437   <attribute name="value" type="tns:FreeFormText" use="required"/>  
1438 </complexType>  
1439 <element name="LocalizedString" type="tns:LocalizedStringType"/>
```

1440

1441 An element InternationalString is capable of supporting multiple locales within its collection of
1442 LocalizedStrings.

1443 The above schema allows a single RegistryObject instance to include values for any NLS sensitive
1444 element in multiple locales.

1445 The following example illustrates how a single RegistryObject can contain NLS sensitive <rim:Name> and
1446 <rim:Description> elements with their value specified in multiple locales. Note that the <rim:Name> and
1447 <rim:Description> use the <rim:InternationalStringType> as their type.

```
1448 <rim:ExtrinsicObject id="{ID}" mimeType="text/xml">  
1449 <rim:Name>  
1450 <rim:LocalizedString xml:lang="en-US" value="customACP1.xml"/>  
1451 <rim:LocalizedString xml:lang="fi-FI" value="customACP1.xml"/>  
1452 <rim:LocalizedString xml:lang="pt-BR" value="customACP1.xml"/>  
1453 </rim:Name>  
1454 <rim:Description>  
1455 <rim:LocalizedString xml:lang="en-US" value="A sample custom ACP"/>  
1456 <rim:LocalizedString xml:lang="fi-FI" value="Esimerkki custom ACP"/>  
1457 <rim:LocalizedString xml:lang="pt-BR" value="Exemplo de ACP  
1458 customizado  
1459 "/>  
1460 </rim:Description>  
1461 </rim:ExtrinsicObject>
```

1462

1463 Since locale information is specified at the sub-element level there is no language or character set
1464 associated with a specific RegistryObject instance.

1465 **11.3.1 Character Set of *LocalizedString***

1466 The character set used by a locale specific String (LocalizedString) is defined by the charset attribute.
1467 Registry Clients SHOULD specify UTF-8 or UTF-16 as the value of the charset attribute of
1468 LocalizedStrings for maximum interoperability.

1469 **11.3.2 Language of *LocalizedString***

1470 The language MAY be specified in xml:lang attribute (Section 2.12 [REC-XML]).

1471 **11.4 NLS and Repository Items**

1472 While a single instance of an ExtrinsicObject is capable of supporting multiple locales, it is always
1473 associated with a single repository item. The repository item MAY be in a single locale or MAY be in
1474 multiple locales. This specification does not specify any NLS requirements for repository items.

1475 **11.4.1 Character Set of Repository Items**

1476 When a submitter submits a repository item, they MAY specify the character set used by the repository
1477 item using the MIME *Content-Type* mime header for the mime multipart containing the repository item as
1478 shown below:

```
1479 Content-Type: text/xml; charset="UTF-8"  
1480  
1481  
1482
```

1483 Registry Clients SHOULD specify UTF-8 or UTF-16 as the value of the charset attribute of
1484 LocalizedStrings for maximum interoperability. A server MUST preserve the charset of a repository item
1485 as it is originally specified when it is submitted to the registry.

1486 **11.4.2 Language of Repository Items**

1487 The Content-language mime header for the mime bodypart containing the repository item MAY specify the
1488 language for a locale specific repository item. The value of the Content-language mime header property
1489 MUST conform to [RFC 1766].

1490 This document currently specifies only the method of sending the information of character set and
1491 language, and how it is stored in a registry. However, the language information MAY be used as one of
1492 the query criteria, such as retrieving only DTD written in French. Furthermore, a language negotiation
1493 procedure, like registry client is asking a favorite language for messages from registry services, could be
1494 another functionality for the future revision of this document.

1495 **12 SOAP Binding**

1496 REST binding will be defined in a separate specification which will be based upon ATOM Syndication
1497 Format Atom Publishing Protocol and OpenSearch??.

1498

1499

1500

1501

1502

1503

1504

1505

1506

1507

1508 **Appendix A. Acknowledgments**

1509 The following individuals have participated in the creation of this specification and are gratefully
1510 acknowledged

1511 **Participants:**

- 1512 • [Participant name, affiliation | Individual member]
- 1513 • [Participant name, affiliation | Individual member]
- 1514 • [Participant name, affiliation | Individual member]

1515

1516 **Appendix B. Revision History**

1517 [optional; should not be included in OASIS standards]

1518 **Appendix C. Non-Normative Text**

1519