Proposal

This is a proposal to ebXML Regrep that it adopt an XML-based syntax to support simple requests to an ebXML Registry. Following the definitions of Focused Query, Totally Ad Hoc Query, Constrained Ad Hoc Query, and Content Based Query given in the proposed requirements document distributed by Farrukh Najmi earlier this week http://lists.ebxml.org/archives/ebxml-regrep/200101/msg00098.html, this proposal is somewhere between a Focused Query and a Constrained Ad Hoc Query.

This proposal is much more flexible than a Focused Query because it allows Boolean predicates on the visible attributes of each class specified in the ebXML Registry Information Model (RIM). We are assuming that ebXML Regrep can agree on a simple syntax for Boolean predicates over a pre-determined set of character string, integer, or date attributes!

The proposal is much easier to implement than Constrained Ad Hoc Query because it does not require the implementation to parse an unfamiliar Query language.

The proposed syntax is more like a script to be followed rather than a language that must be parsed. The script depends on the classes and relationships defined in the RIM, so even very simple syntax may result in quite powerful requests to the Registry. This is demonstrated in the Example section below, which shows how each of the example OQL queries in Farrukh's proposed requirements document can be expressed in this simple XML.

Pre-Conditions

1) Begin with a set of classes that will be the focus of individual queries. Each query will return a set of references to persistent instances of an identified class. Using Figure 1 in RIM, I propose that the set of classes that will support XML queries be: ManagedObject, Organization, and ClassificationScheme. The other classes could be supported too, but it's really not necessary since each instance of one of those classes is so tightly bound to one of these three.

   It is not necessary to consider, separately, Object, Extrinsic, and Instrinsic queries, since they are all easily handled as ManagedObject queries.

   We chose to use ClassificationScheme instead of ClassificationNode as the basis of querying classification nodes. This is because each node is an item in a single tree hierarchy. We choose to identify the root of the tree as the identifier for the entire hierarchy. Thus a ClassificationSchemeQuery will return a list of references to the root nodes of classification schemes. If one also desires references to specific sub-nodes in the tree the XML query will provide those options.

   Choose the following three XML elements as the root of an XML document that declares such a query:

      ManagedObjectQuery
      ClassificationSchemeQuery
      OrganizationQuery

   We will focus on the first of these in what follows. Compare this approach with that specified in Section 7.4 of the OASIS Registry/Repository specification.

2) Each choice of a class pre-determines a virtual XML document that can be queried as a tree. Rules in the Registry Services specification will make clear the structure of this tree. These rules do not yet exist, but they are completely straight-forward because the RIM Information Model is relatively simple. All of associations identified in Figure 1 of RIM identify the obvious paths. The tree structure is also re-inforced by the methods already defined in RIM, a method is either a reference to a single attribute of an instance I'm looking at, or it returns a set of nodes from the next level down in the virtual tree.

3) Next consider Boolean predicates over the visible attributes of each class in RIM. Identify those predicates as follows:

    ManagedObjectFilter
    AssociationFilter
    ClassificationFilter
    ExternalLinkFilter
    ClassificationSchemeFilter -- just on the root nodes of the tree hierarchy
    ClassificationNodeFilter -- on any node in the scheme hierarchy.
    AuditableEventFilter

   In some cases, there are multiple choices for the linkage between two classes. For example, the linkage from ManagedObject to Association can be through the SourceObject or through the TargetObject. We need to provide alternatives for either choice.

    AssociationSourceFilter (AssociationFilter)
    AssociationTargetFilter (AssociationFilter)

   In the first case, linkage from ManagedObject to Association is by linking the ManagedObject instance to the source object of the association. In the second case, linkage from ManagedObject to Association is by linking the ManagedObject instance to the target object of the association.

4) An XML Query only returns a set of references to instances in its base class. Next we need a syntax to build an output document that will be an XML document that is the output of the XML query. We assume the existence of the same document structure as in step 2 above. Then we allow the user to specify which branches of the tree they want returned. Again this can be accomplished in XML by allowing users to specify what they want returned. Just as in step 3) we'd have the following possibilities for the subnodes that get returned with ManagedObject:

    WithAssociations
    WithClassifications
    WithExternalLinks
    WithAuditableLinks
    etc.

5) Finally, why not offer users the opportunity to filter the branches of the output document by allowing class filters on each of the output choices:

    WithAssociations (AssociationFilter*)
    WithClassifications (ClassificationFilter*)
    WithExternalLinks (ExternalLinkFilter*)
    WithAuditableLinks (AuditableLinkFilter*)
    etc.

Examples

The following example ebXML OQL-based queries are taken from the document distributed by Farrukh Najmi to the ebXML email list on January 20: http://lists.ebxml.org/archives/ebxml-regrep/200101/msg00098.html

Following each OQL query is an XML query based on the very simple approach outlined in the OASIS Registry/Repository specification: ftp://xsun.sdct.itl.nist.gov/regrep/OasisRegrepSpec.pdf

The XML approach is intended to be so simple that it is NOT really a query! Instead, it can be regarded as a script that should be followed to produce a result. The effect of the script is pre-determined by careful definitions in the OASIS and/or ebXML specification. Every implementation of the registry should be able to implement it easily - if not, it has failed its intent! It assumes only that a very simple syntax for declaring predicates over the visible attributes of a single class defined in ebXML RIM has been agreed to by ebXML Regrep. An example of what that syntax might look like, but not necessarily a proposal for it, is: ftp://xsun.sdct.itl.nist.gov/regrep/AttributePredicate.dtd

----

```
SELECT DISTINCT obj FROM Object WHERE  obj.name LIKE ‘%bicycle%’
AND obj.majorVersion >= 1 AND obj.minorVersion >= 3;
```

```
<ManagedObjectQuery>
<ManagedObjectFilter>
        name CONTAINS "bicycle" AND
        majorVersion GREATER_EQUAL "1" AND
        minorVersion LESS_EQUAL "3"
</ManagedObjectFilter>
</ManagedObjectQuery>
```

```
SELECT DISTINCT obj FROM Object WHERE obj.isClassifiedBy(<classificationNode>);
```

```
<ManagedObjectQuery>
<ClassificationFilter>
        classificationScheme="urn:somepath:naics" AND
        levelCode="1" AND itemValue="firstValueInPath" AND
        etc. AND
        levelCode="N" AND itemValue="lastValueInPath"
</ClassificationFilter>
</ManagedObjectQuery>
```

NOTE: Could use <classificationNode>, XPath, or any other "nice" syntax for path declaration, but not really required except for convenience! First order Boolean predicate on visible Classification attributes is sufficient.

```
SELECT DISTINCT eo FROM ExtrinsicObject WHERE eo.isClassifiedBy(ClassificationNode(path:
Industry.Automotive)) AND eo.isClassifiedBy(ClassificationNode(path: Geography.Asia.Japan));
```

```
<ManagedObjectQuery>
<ManagedObjectFilter>
        objectType="extrinsic"
```

```
</ManagedObjectFilter>
<ClassificationFilter>
        classificationScheme="urn:path:naics" AND
        levelCode="1" AND itemValue="Geography" AND
        levelCode="2" AND itemValue="lastValueInPath" AND
        levelCode="3" AND itemValue="lastValueInPath"
</ClassificationFilter>
</ManagedObjectQuery>
```

```
SELECT cn FROM ClassificationNode WHERE cn.classifies(<object’s id>);
```

```
<ClassificationSchemeQuery>
<ManagedObjectFilter>
        URI="urn:mypath:myobject"
</ManagedObjectFilter>
</ClassificationSchemeQuery>
```

NOTE: A <ClassificationSchemeQuery> knows implicitly, i.e. from the associations in RIM, how classification nodes are linked to classifications. Thus this simple XML query makes sense provided that the rules in the specification are clear for how this linkage is defined! They will be! Compare with Section 7.4.1, Semantic Rule 2, of the OASIS spec. The rules in ebXML RS will depend on the "classifies" method; it is not necessary to also name that method in the XML syntax! That's the advantage we have in working from a very simple Information Model.

CAUTION: The above query only makes sense in Registries containing both the classification node and the classification. This may not be the case, especially when a large classification scheme, e.g. UNSPSC with 10,000+ entries, exists in only one place. But if the classifications were required to reference a managed object in the local registry that carries the metadata for the remote classification scheme, as the OASIS spec requires, then the query would make sense again if reworded as follows:

```
<GetManagedObject>    - - See Section 7.2 of the OASIS spec as example
<ManagedObjectQuery>
<ManagedObjectFilter>
        URI="urn:mypath:myObject"
</ManagedObjectFilter>
</ManagedObjectQuery>  - - The result is a REF to "myObject"
<WithClassifications/>
</GetManagedObject>
```

GetManagedObject returns an XML document with "myObject" linked to all of its classifications. If we didn't want all of them, we could add a ClassificationFilter as a subelement of WithClassifications.

```
SELECT assoc FROM Association WHERE assoc.sourceObject = <uuid>;
```

I don't think it's necessary to support Association queries explicitly, since we can get the same result from the combination of a ManagedObjectQuery and a GetManagedObject service request, as follows:

```
<GetManagedObject>    - - See Section 7.2 of the OASIS spec as example
<ManagedObjectQuery>
<ManagedObjectFilter>
        sourceURI="urn:mypath:myObject"
</ManagedObjectFilter>
```

```
</ManagedObjectQuery>  - - Returns a REF to "myObject"
<WithAssociations/>
</GetManagedObject>
```

The ManagedObjectQuery returns a REF to "myObject". Then GetManagedObject returns an XML document that links "myObject" to <u>all</u> of its associations where it is the source object. If we didn't want <u>all</u> of them, we could add an AssociationFilter as a subelement of WithAssociations.

```
SELECT DISTINCT assoc FROM Association WHERE assoc.targetObject = <uuid>;
```

```
<GetManagedObject>    - - See Section 7.2 of the OASIS spec as example
<ManagedObjectQuery>
<AssociationSourceFilter>
<AssociationFilter>
            targetURI="urn:mypath:myObject"
</AssociationFilter>
</AssociationSourceFilter>
</ManagedObjectQuery>  - - Returns a REFs to all objects associated with "MyObject"
<WithAssociations>
<AssociationFilter>
            targetURI="urn:mypath:myObject"
</AssociationFilter>
</WithAssociations>
</GetManagedObject>
```

GetManagedObject returns an XML document that links selected objects to their associations with "MyObject". The second use of AssociationFilter is needed in order to not return all associations of the selected objects.

```
SELECT DISTINCT assoc FROM Association WHERE assoc.name = <name>;
```

```
<GetManagedObject>    - - See Section 7.2 of the OASIS spec as example
<ManagedObjectQuery>
<AssociationSourceFilter>
<AssociationFilter>
            assocName="MyName"
</AssociationFilter>
</AssociationSourceFilter>
</ManagedObjectQuery>  - - Returns REFs to all objects with "MyName" associations
<WithAssociations>
<AssociationFilter>
            assocName="MyName"
</AssociationFilter>       - - This second AssociationFilter is NOT superfluous!
</WithAssociations>
</GetManagedObject>
```

GetManagedObject will return a large XML document consisting of every managed object that has an association named "MyName" and each managed object will be linked to its collection of such associations - this linkage is an important added benefit since it is likely to be desired anyway.

```
SELECT DISTINCT assoc FROM Association WHERE assoc.sourceRole = <roleName>;
```

```
<GetManagedObject>    - - See Section 7.2 of the OASIS spec as example
<ManagedObjectQuery>
<AssociationSourceFilter>
<AssociationFilter>
            sourceRole="SomeRole"    - - or any other role name.
</Associationfilter>
</AssociationSourceFilter>
</ManagedObjectQuery>    - - Returns REFs to objects having "SomeRole" associations
<WithAssociations>
<AssociationFilter>
            sourceRole="SomeRole"    - - or any other role name.
</Associationfilter>
</WithAssociations>
</GetManagedObject>
```

```
SELECT DISTINCT assoc FROM Association WHERE assoc.targetRole = <roleName>;
```

```
<GetManagedObject>    - - See Section 7.2 of the OASIS spec as example
<ManagedObjectQuery>
<AssociationSourceFilter>
<AssociationFilter>
            sourceRole="inverse_of_targetRole"
</AssociationFilter>
</AssociationSourceFilter>
</ManagedObjectQuery>
<WithAssociations>
<AssociationFilter>
            sourceRole="inverse_of_targetRole"
</AssociationFilter>
</WithAssociations>
</GetManagedObject>
```

NOTE: We have to work with the inverse role-name because the definition of WithAssociations assumes that linkage from ManagedObject to Association is through the sourceObject. Alternatively we could have two different flavors of WithAssociations just like we have two different flavors of AssociationFilter. The OASIS spec is biased in terms of always assuming that the linkage from Association to ManagedObject is through the sourceObject unless Target linking is explicitly declared.

```
SELECT DISTINCT assoc FROM Association WHERE assoc.associationType = <associationType>;
```

```
<GetManagedObject>    - - See Section 7.2 of the OASIS spec as example
<ManagedObjectQuery>
<AssociationSourceFilter>
            associationType="typeName"
</AssociationSourceFilter>
</ManagedObjectQuery>
<WithAssociations>
<AssociationFilter>
```

```
            associationType="typeName"
</AssociationFilter>     - - This 2nd filter on associationType is NOT redundant!
</WithAssociations>
</GetManagedObject>
```

```
SELECT DISTINCT assoc FROM Association WHERE Assoc.sourceObject = "sourceObj"
assoc.sourceRole = "buysFrom" AND assoc.sourceRole = "sellsTo";
```

The above query is not well formed, but it would be handled just like the previous several queries in the proposed simple XML syntax.

```
SELECT DISTINCT pkg FROM Package WHERE pkg IN obj.getPackages();

<ManagedObjectQuery>
<ManagedObjectFilter>
            objectType="registry package"
</ManagedObjectFilter>
</ManagedObjectQuery>
```

```
SELECT DISTINCT obj FROM Object WHERE obj IN pkg.getMemberObjects();

<GetManagedObject>    - - See Section 7.2 of the OASIS spec as example
<ManagedObjectQuery>
<ManagedObjectFilter>
            URI="urn:somepath:myPackage"
</ManagedObjectFilter>
</ManagedObjectQuery>  - - Returns a REF to "myPackage"
<WithAssociations>
<AssociationFilter>
            associationType="Contains"
</AssociationFilter>
</WithAssociations>
</GetManagedObject>
```

GetManagedObject returns an XML document that links "myPackage" to each of its member elements.

EXTENSION: But what if some of these member objects are themselves packages? How would we get the collection of objects that is the transitive closure? Using syntax introduced in the OASIS spec Section 7.1, this extension is relatively straight-forward to declare!

```
SELECT DISTINCT pkg FROM Package WHERE pkg IN obj.getPackages() AND
pkg.name LIKE ‘%RosettaNet%’ AND pkg.status != ‘DEPRECATED’;

<ManagedObjectQuery>
<ManagedObjectFilter>
            objectType="registry package" AND
            name CONTAINS "RosettaNet" AND
            status NOT_EQUAL "Deprecated"
```

```
</ManagedObjectFilter>
</ManagedObjectQuery>
```

```
SELECT DISTINCT link FROM ExternalLink WHERE link IN obj.getExternalLinks();
```

NOTE: Where does the obj REF come from? The following query gets it first, using its URN, then retrieves all ExternalLinks linked to it:

```
<GetManagedObject>
<ManagedObjectQuery>
 <ManagedObjectFilter>
            URN="urn:somepath:myObject"
</ManagedObjectFilter>
</ManagedObjectQuery>    - - Returns a REF to "myObject"
<WithExternalLinks/>
</GetManagedObject>
```

GetManagedObject returns an XML document that links "myObject" to all of its declared external links. If instead we wanted just some of the links we could add an ExternalLinksFilter as a subelement of WithExternalLinks. This works especially well if external links are classified according to some classification scheme so that you can just get the ones you're really interested in e.g. all links that locate a "WhitePaper" describing "myObject".

```
SELECT DISTINCT obj FROM Object WHERE obj IN link.getLinkedObjects();
```

NOTE: I don't think this query makes sense. The LinkedObjects are external objects. How can one of them also be an object in the Registry?

```
SELECT DISTINCT link FROM ExternalLink WHERE link IN obj.getExternalLinks() AND
link.description LIKE ‘%legal%’ AND link.externalURI LIKE ‘%http://%’;
```

NOTE: Where does the obj REF come from? The following query gets it first, using its URN, then retrieves all ExternalLinks linked to it, filtered on the desired attributes:

```
<GetManagedObject>
<ManagedObjectQuery>
 <ManagedObjectFilter>
            URN="urn:somepath:myObject"
</ManagedObjectFilter>
</ManagedObjectQuery>    - - Returns a REF to "myObject"
<WithExternalLinks>
<ExternalLinkFilter>
            description CONTAINS "legal" AND
            externalURI CONTAINS "http://"
</ExternalLinkFilter>
</WithExternalLinks>
</GetManagedObject>
```

GetManagedObject returns an XML document that links "myObject" to only those declared external links that satisfy the predicate of the ExternalLinkFilter.

```
SELECT DISTINCT ev FROM AuditableEvent WHERE ev IN obj.getAuditTrail();
```

NOTE: Where does the obj REF come from? The following query gets it first, using its URN, then retrieves all AuditableEvents linked to it:

```
<GetManagedObject>
<ManagedObjectQuery>
 <ManagedObjectFilter>
            URN="urn:somepath:myObject"
</ManagedObjectFilter>
</ManagedObjectQuery>    - - Returns a REF to "myObject"
<WithAuditableEvents/>
</GetManagedObject>
```

GetManagedObject returns an XML document that links "myObject" to <u>all</u> auditable events associated with "myObject". This is pretty weakly specified in the current ebXML specification. Compare with the notion of Request and Impact in the OASIS specification.

```
SELECT DISTINCT obj FROM ManagedObject, Package WHERE obj.status == ‘DEPRECTAED’ AND
exists pkgs IN obj.getPackages() : pkgs.name LIKE ‘%Acme%’ ;
```

NOTE: I don't think this query makes sense. The FROM clause mandates a Join between ManagedObject and Package. But according to Figure 2 of RIM, a Package ISA ManagedObject. How can this be? I would prefer a specification where a package and the metadata describing a package are two separate objects so that it's easy to decide when you are referencing the package and when you are referencing the metadata that describes the package, e.g. its "status", "stability", "expirationdate", etc.

Let me bring up my Sept '99 proposal one more time! Please make a distinction between an object submitted to the registry for registration (e.g. RegisteredObject) and the metadata that describes the object (e.g. RegistryEntry). Then the funny joins in the above OQL query would be unnecessary!

I think the above query is asking for references to the set of all deprecated packages that have a package member with a name containing the string "Acme". Such a list of references would be returned by the following ManagedObjectQuery:

```
<ManagedObjectQuery>
<ManagedObjectFilter>
            objectType="registry package" AND
            status="deprecated"
</ManagedObjectFilter>     - - Returns a list of REFs to deprecated packages
<AssociationSourceFilter>
<AssociationFilter>
            associationType="Contains" AND
            targetURN CONTAINS "Acme"
</AssociationFilter>
</AssociationSourceFilter>
</ManagedObjectQuery>
```

The ManagedObjectQuery returns an XML document with managed objects that reference deprecated packages, that are the sourceObject in a Contains Association, and have a member object with a name like Acme.

DISCLOSURE: If the above query had requested any other conditions on the package members, involving various other attributes of ManagedObject, I would NOT have been able to do it with the proposed simple syntax. This is because it involves a Join of ManagedObject with itself - something the simple XML was not designed to handle. But -- if it is really that important, certain predefined self-joins could be supported without too much additional effort. But ad hoc join conditions would never be supported using this simple XML -- that's when it would be appropriate to support a more general purpose query language like SQL, OQL, or XML Query. I do not advocate that as a requirement for Phase I.

THE END