1 Overview

This note describes an approach for the Registry to support the discovery of content in the registry based on data contained within the content itself as opposed to within metadata describing content. This has been referred to as content-based queries in past discussions.

2 Problem Description

The Registry contains two types of content:

1. Content that is submitted by Registry clients
2. Metadata that describes content submitted by Registry clients

Current Registry query mechanisms provide support for discovery of content based on metadata. Specifically, the discovery process is based on a flexible content classification mechanism where instances of Classification objects link content to classification schemes defined as a tree of ClassificationNode objects.

The current discovery mechanisms fall short on providing a way to discover Registry content based on the data within the actual content. Even the simplest use case of finding all Collaboration Protol Profiles [CPP] that have a RoleName of “Supplier” is not possible. As ebXML defines Core Components this limitation will become more and more obvious.

3 Alternatives Considered

This section describes the various alternatives that have been considered to support content-based queries.

3.1 Single Query Syntax For Content and Metadata

This approach attempts to define a single query syntax for the registry that can support predicates that operate on both metadata and content. In this approach content is viewed as being virtually derived from its associated metadata as defined by the Registry Information Model [RIM]. Thus a query could predicate on the attributes defined in RIM by the metadata classes or on data defined in content. Two query syntax alternatives have been considered. These include XPATH syntax as defined by W3C and a constrained subset of OQL as defined by ODMG Standard.

The XPATH alternative has the limitation that it only supports content-based queries on XML document content. The OQL approach has the limitation that it has to be mapped to XML content.

3.2 Content Specific Query Syntax

In this approach, it is assumed that there is a global overall query syntax. Alternatives considered are XPATH and OQL as in the previous approach. However, in this approach specific predicates in the query may be in a different query syntax based upon the nature of the content. This approach is undesirable because implementers of the Registry as well as Registry clients would have to implement support for more than one query language syntax.

[Note] Note that many implementers of the Registry are not in favor of support content-based queries because of the totally ad hoc nature of such queries and their potential for being extremely resource intensive in the absence of indexed content.

4 Proposed Solution

The proposed solution is to support content-based queries indirectly through the existing classification mechanism in the Registry. With a relatively simple change in the Registry DTD we can define logical indexes on any schema when it is submitted. An instance of such a logical index would define a link between a specific attribute or element value in an XML document and a ClassificationNode in a classification scheme within the registry. The registry could utilize this index to automatically classify content instances of the schema at the time the content instance is submitted based on the specified classification scheme.

The result of this approach is that content gets classified when submitted and can be subsequently discovered using the existing classification based discovery mechanism of the Registry. Since Registry implantations are expected to optimize discovery based on classifications, this approach is expected to result in efficient discovery and retrieval of content based on the data contained within the content.

This approach is presented here in terms of XML DTDs and XML documents that are instances of these DTDs. It is worth noting that the same approach will support content based queries on anytype of content where there is metadata (schema) describing the content structure and the content is readable by the Registry (i.e. it is not opaque as in a JPEG image). For example, indexes could be defined when submitting a Java class definition and serialized Java objects that are instances of this class may be automatically classified and subsequently discovered based on the automatic classification.

[Note] This approach is conceptually similar to the way databases support indexed retrieval. DBAs define indexes on tables in the schema. When data is added to the table, the data gets

automatically indexed.

4.1 Index Definition

This section describes how the logical index could be defined in the Registry schema. The complete Registry DTD is specified in the appendix. This section simply describes the changes required to support index definition.

A new schema element is defined called ClassificationIndex. The ClassificationIndex element inherits the attributes of the base class Object in [RIM]. It then defines specialized attributes as follows:

1. classificationNode: This attribute references a specific classification node by its ID
2. contentIdentifier: This attribute identifies a specific data element within the content. For XML content this is proposed to be in XPATH syntax. For other types of content other syntaxes may be defined in the future in a content specific manner as needed.

```
<!ENTITY % ObjectAttributes " description CDATA  #IMPLIED
                ID        CDATA  #REQUIRED
                name      CDATA  #REQUIRED">
…
<!ELEMENT ClassificationIndex EMPTY>
<!ATTLIST ClassificationIndex  %ObjectAttributes;
                        classificationNode CDATA  #REQUIRED
                        contentIdentifier  CDATA  #REQUIRED >
<!ELEMENT ClassificationIndexList  (ClassificationIndex  )*>
…
<!ELEMENT SubmittedObject  (ExtrinsicObject? , ClassificationIndexList? ,
ClassificationList? , AssociationList? , ExternalLinkList? , PackageList? )>
```

Finally the existing element SubmittedObject in the Registry DTD is extended to include an optional ClassificationIndexList which is a collection of ClassificationIndex definitions. The ClassificationIndexList is ignored if the content being submitted is not of the SCHEMA objectType.

Appendix A Complete Registry DTD

The following is the complete proposed DTD for the Registry:

```
<?xml version='1.0' encoding='UTF-8' ?>

```

```
<!ELEMENT RequestAcceptedResponse EMPTY>
<!ATTLIST RequestAcceptedResponse  xml:lang      NMTOKEN  #REQUIRED
                        interfaceId    CDATA    #REQUIRED
                        requestMessage CDATA    #REQUIRED
                        actionId       CDATA    #REQUIRED >
<!ENTITY % errorSchema SYSTEM "ebXMLError.dtd">

%errorSchema;

<!ENTITY % ObjectAttributes " description CDATA  #IMPLIED
                ID         CDATA  #REQUIRED
                name       CDATA  #REQUIRED">

<!ENTITY % ManagedObjectAttributes " %ObjectAttributes;
                status          (SUBMITTED | APPROVED | DEPLOYED |
DEPRECATED )  'SUBMITTED'
                majorVersion    CDATA  '1'
                minorVersion    CDATA  '0'">

<!ELEMENT ManagedObject EMPTY>
<!ATTLIST ManagedObject  %ManagedObjectAttributes; >
<!ELEMENT ExtrinsicObject EMPTY>
<!ATTLIST ExtrinsicObject  %ManagedObjectAttributes;
                    contentURN          CDATA    #IMPLIED
                    mimeType            CDATA    #IMPLIED
                    objectType          (PARTY_AGREEMENT |
                                PARTY_PROFILE |
                                PROCESS |
                                ROLE |
                                SERVICE_INTERFACE |
                                SOFTWARE_COMPONENT |
                                TRANSPORT |
                                UML_MODEL |
```

```
                                UNKNOWN |
                                XML_SCHEMA )  #REQUIRED
                opaque            CDATA     'false'
                a-dtype           NMTOKENS  'opaque boolean' >
<!--
A ClassificationIndex is specified on SCHEMA ExtrinsicObjects to define
an automatic Classification of instance objects of the schema using
the specified classificationNode as parent and a ClassificationNode
created or selected by the object content as selected by the contentIdentifier
-->
<!ELEMENT ClassificationIndex EMPTY>
<!ATTLIST ClassificationIndex  %ObjectAttributes;
                    classificationNode CDATA  #REQUIRED
                    contentIdentifier  CDATA  #REQUIRED >
<!-- ClassificationIndexList contains new ClassificationIndexs  -->
<!ELEMENT ClassificationIndexList  (ClassificationIndex )*>

<!ENTITY % IntrinsicObjectAttributes " %ManagedObjectAttributes;">

<!ELEMENT IntrinsicObject EMPTY>
<!ATTLIST IntrinsicObject  %ManagedObjectAttributes; >
<!-- Leaf classes that reflect the concrete classes in RIM -->
<!ELEMENT ManagedObjectList  (Association | Classification |
ClassificationNode | ExternalLink | Organization | ExtrinsicObject )*>

<!-- Reference to an Object via its URN specified by it ID attribute -->
<!ELEMENT ObjectRef EMPTY>
<!ATTLIST ObjectRef  uuid CDATA  #REQUIRED >
<!ELEMENT ObjectRefList  (ObjectRef )*>

<!--
An ExternalLink specifies a link from a ManagedObject and an external URI

```

```
The sourceObjectRef is ref to the ManagedObject

The sourceObjectRef is optional when Association is defined part of
a SubmittedObject.
-->
<!ELEMENT ExternalLink EMPTY>
<!ATTLIST ExternalLink  %IntrinsicObjectAttributes;
                sourceObjectRef         CDATA  #IMPLIED
                uri                     CDATA  #IMPLIED >
<!-- ExternalLinkList contains new ExternalLinks or refs to pre-existing
ExternalLinks -->
<!ELEMENT ExternalLinkList  (ExternalLink | ObjectRef )*>

<!--
An Association specifies references to two previously submitted
managed objects.

The sourceObjectRef is ref to the sourceObject in association
The targetObjectRef is ref to the targetObject in association

The sourceObjectRef is optional when Association is defined part of
a SubmittedObject.
-->
<!ELEMENT Association EMPTY>
<!ATTLIST Association  %IntrinsicObjectAttributes;
                fromLabel               CDATA    #IMPLIED
                toLabel                 CDATA    #IMPLIED
                associationType         (CLASSIFIED_BY |
                                        CONTAINED_BY |
                                        CONTAINS |
                                        EXTENDS |
                                        IMPLEMENTS |
                                        INSTANCE_OF |
```

```
                                RELATED_TO |
                                SUPERSEDED_BY |
                                SUPERSEDES |
                                USED_BY |
                                USES )  #FIXED 'RELATED_TO'
                bidirection             CDATA    'false'
                sourceObjectRef         CDATA    #REQUIRED
                targetObjectRef         CDATA    #REQUIRED
                a-dtype                 NMTOKENS  'bidirection boolean' >
<!ELEMENT AssociationList  (Association )*>

<!--
A Classification specifies references to two previously submitted

managed objects.

The sourceObjectRef is ref to the sourceObject in Classification
The targetObjectRef is ref to the targetObject in Classification

The sourceObjectRef is optional when Classification is defined part of
a SubmittedObject.
-->
<!ELEMENT Classification EMPTY>
<!ATTLIST Classification  %IntrinsicObjectAttributes;
                    sourceObjectRef         CDATA  #REQUIRED
                    targetObjectRef         CDATA  #REQUIRED >
<!ELEMENT ClassificationList  (Classification )*>

<!ELEMENT Package EMPTY>
<!ATTLIST Package  %IntrinsicObjectAttributes; >
<!-- PackageList contains new Packages or refs to pre-existing Packages -->
<!ELEMENT PackageList  (Package | ObjectRef )*>

```

```
<!ENTITY % TelephoneNumberAttributes " areaCode   CDATA  #REQUIRED
                contryCode CDATA  #REQUIRED
                extension  CDATA  #IMPLIED
                number     CDATA  #REQUIRED
                url        CDATA  #IMPLIED">

<!ELEMENT TelephoneNumber EMPTY>
<!ATTLIST TelephoneNumber  %TelephoneNumberAttributes; >
<!ELEMENT FaxNumber EMPTY>
<!ATTLIST FaxNumber  %TelephoneNumberAttributes; >
<!ELEMENT MobileTelephoneNumber EMPTY>
<!ATTLIST MobileTelephoneNumber  %TelephoneNumberAttributes; >
<!-- PostalAddress -->
<!ELEMENT PostalAddress EMPTY>
<!ATTLIST PostalAddress  city      CDATA  #REQUIRED
                  country    CDATA  #REQUIRED
                  postalCode CDATA  #REQUIRED
                  state      CDATA  #REQUIRED
                  street     CDATA  #REQUIRED >
<!-- PersonName -->
<!ELEMENT PersonName EMPTY>
<!ATTLIST PersonName  firstName  CDATA  #REQUIRED
                middleName CDATA  #REQUIRED
                lastName   CDATA  #REQUIRED >
<!-- Contact -->
<!ELEMENT Contact  (PostalAddress , PersonName , FaxNumber? ,
TelephoneNumber , MobileTelephoneNumber? )>
<!ATTLIST Contact  email CDATA  #REQUIRED >
<!-- Organization -->
<!ELEMENT Organization  (PostalAddress , Contact , FaxNumber? ,
TelephoneNumber )>
<!ATTLIST Organization  %IntrinsicObjectAttributes;
                  parent               CDATA  #IMPLIED >
```

```
<!--
ClassificationNode is used to submit a Classification tree to the Registry.
Note that this is a recursive schema definition.

The parent attribute of a node in tree is implied by the enclosing
ClassificationNode
The children nodes of a node are implied by enclosing immediate child elements
of type ClassificationNode.
-->
<!ELEMENT ClassificationNode EMPTY>
<!ATTLIST ClassificationNode %IntrinsicObjectAttributes;>

<!--
parent is the URN to the parent node. parent is optional if ClassificationNode is
enclosed
in a parent ClassificationNode or if root ClassificationNode
-->
<!ATTLIST ClassificationNode parent                CDATA  #IMPLIED>

<!ELEMENT ClassificationNodeList  (ClassificationNode )*>

<!--
End information model mapping.

Begin Registry Services Interface
-->
<!--
The SubmittedObject provides meta data for submitted object
Note object being submitted is in a separate document that is not
in this DTD.
-->
<!ELEMENT SubmitObjectsRequest  (SubmittedObject+ )>

```

```
<!--
The ExtrinsicObject provides meta data about the object being submitted
ClassificationList can be optionaly be specified to define Classifications
for the SubmittedObject

AssociationList can be optionaly be specified to define Associations
for the SubmittedObject

The ExternalLinkList provides zero or more external objects related to
the object being submitted.
-->
<!ELEMENT SubmittedObject  (ExtrinsicObject? , ClassificationIndexList? ,
ClassificationList? , AssociationList? , ExternalLinkList? , PackageList? )>

<!--
The ObjectRefList is the list of
refs to the managed objects being approved.
-->
<!ELEMENT ApproveObjectsRequest  (ObjectRefList )>

<!--
The ObjectRefList is the list of
refs to the managed objects being deprecated.
-->
<!ELEMENT DeprecateObjectsRequest  (ObjectRefList )>

<!--
The ObjectRefList is the list of
refs to the managed objects being removed
-->
<!ELEMENT RemoveObjectsRequest  (ObjectRefList )>

<!ELEMENT GetRootClassificationNodesRequest EMPTY>
```

```
<!--
The namePattern follows SQL-92 syntax for the pattern specified in
LIKE clause. It allows for selecting only those root nodes that match
the namePattern. The default value of '*' matches all root nodes.
-->
<!ATTLIST GetRootClassificationNodesRequest namePattern CDATA  "*">

<!--
The response includes a ClassificationNodeList which has zero or more
ClassificationNodes
-->
<!ELEMENT GetRootClassificationNodesResponse  (ClassificationNodeList |
ebXMLError )>

<!--
Get the classification tree under the ClassificationNode specified parentRef.

If depth is 1 just fetch immediate child
nodes, otherwise fetch the descendant tree upto specified depth level.
If depth is 0 that implies fetch entire sub-tree
-->
<!ELEMENT GetClassificationTreeRequest EMPTY>
<!ATTLIST GetClassificationTreeRequest  parent CDATA  #REQUIRED
                                depth  CDATA  '1' >
<!--
The response includes a ClassificationNodeList which includes only
immediate ClassificationNode children nodes if depth attribute in
GetClassificationTreeRequest was 1, otherwise the decendent nodes
upto specifed depth level are returned.
-->
<!ELEMENT GetClassificationTreeResponse  (ClassificationNodeList |
ebXMLError )>
```

```
<!--
Get refs to all managed objects that are classified by all the
ClassificationNodes specified by ObjectRefList.
Note this is an implicit logical AND operation
-->
<!ELEMENT GetClassifiedObjectsRequest  (ObjectRefList )>

<!--
objectType attribute can specify the type of objects that the registry
client is interested in, that is classified by this ClassificationNode.
It is a String that matches a choice in the type attribute of ExtrinsicObject.
The default value of '*' implies that client is interested in all types
of managed objects that are classified by the specified ClassificationNode.
-->
<!ATTLIST GetClassifiedObjectsRequest objectType CDATA  "*">

<!--
The response includes a ManagedObjectList which has zero or more
ManagedObjects that are classified by the ClassificationNodes
specified in the ObjectRefList in GetClassifiedObjectsRequest.
-->
<!ELEMENT GetClassifiedObjectsResponse  (ManagedObjectList | ebXMLError
)>

<!--
An Ad hoc query request specifies a query string as defined by [RS] in the
queryString attribute
-->
<!ELEMENT AdhocQueryRequest EMPTY>
<!ATTLIST AdhocQueryRequest  queryString CDATA  #REQUIRED >
<!--
The response includes a ManagedObjectList which has zero or more
```

```
ManagedObjects that metach the query specified in AdhocQueryRequest.
-->
<!ELEMENT AdhocQueryResponse  (ManagedObjectList | ebXMLError )>

<!--
Gets the actual content (not metadata) specified by the ObjectRefList
-->
<!ELEMENT GetContentRequest  (ObjectRefList )>

<!--
The GetObjectsResponse will have no sub-elements if there were no errors.
The actual contents will be in the other payloads of the message.
If any errors were encountered the message will contain the ebXMLError and
the content payloads will be empty.
-->
<!ELEMENT GetContentResponse  (ebXMLError? )>

<!--
The contrived root node
-->
<!ELEMENT RootElement  (RequestAcceptedResponse | ebXMLError |
SubmitObjectsRequest | ApproveObjectsRequest | DeprecateObjectsRequest |
RemoveObjectsRequest | GetRootClassificationNodesRequest |
GetRootClassificationNodesResponse | GetClassificationTreeRequest |
GetClassificationTreeResponse | GetClassifiedObjectsRequest |
GetClassifiedObjectsResponse | AdhocQueryRequest | AdhocQueryResponse |
GetContentRequest | GetContentResponse )>

```