INTERNET DRAFT
draft-popp-cnrp-00.txt
February 14, 1999
Expires August 1999

Nicolas Popp
Centraal Corporation
Michael Mealling
Network Solutions

A resolution protocol for Common Name Namespaces

Status of this Memo

Abstract

People often refer to things in the world by a common name or phrase, e.g., a trade name, company name, or a book title. These names are sometimes easier for people to remember and enter than URLs; many people consider URLs hard to remember or type.

This document proposes an abstract protocol for the resolution of common names. The protocol is based on XML, RDF and the Dublin Core metadata elements. The proposed mechanism allows for a very dynamic resolution process where the client has no predefined knowledge of the namespace or the resolution service. The protocol supports an extensible query and response interface with multiple independent XML vocabularies. Furthermore, it allows for multiple concrete implementations (HTTP, LDAP, WHOIS,...). Both the specifics of the query interface and the type of implementation can be discovered at runtime by the resolution client.

This document is intended for discussion at the Common Name

Resolution Protocol BOF (renamed from Human Friendly Identifier BOF). It may be discussed on cnrp-ietf@lists.internic.net. Send in body subscribe cnrp-ietf to listserv@lists.internic.net. The mail archive is at http://lists.netsol.com/lists/cnrp.

1. Introduction: The common name as resource metadata

A CN is a common name used to access a resource on the Internet. A resource can be an HTML Web page, a person's email address, a downloadable software program, basically any type of addressable objects that can reside on a network. In this discussion, any network entity that has an URI [URI] qualifies as a "resource".

The CN is only one of the many characteristics of the resource. For instance, The URI for the resource is another characteristic. Descriptive information about a resource is called metadata [METADATA]. Therefore, a CN is an element of metadata, or metadata property. In addition to a name or a URI, a wide variety of metadata properties are necessary to properly characterize a resource. For example, when the resource is a Web page, a critical piece of information for the user is the "language" of the Web page.

Since a CN will generally not be unique across a namespace, it is important to have other properties to identify a specific resource. For instance, the CN "BMW" can be associated with the Web page at http://www.bmw.com, the official BMW homepage for the English language. At the same time, the CN "BMW" can also be associated with the Web page at http://www.bmw.de, the official homepage for BMW in the German language. As a namespace grows to encompass several thousands of names, it becomes necessary to introduce other metadata properties. These properties are used to give structure to the namespace so that resources can be precisely referred to.

Therefore, it is important to think of a Common Name as one property of a larger metadata object. In that context, a Common Name namespace is a multidimensional space of metadata vectors where each unique property defines a dimension. In such space, finding all the vectors that meet some specific requirements is called the resolution process.

This means that in order to define a general resolution protocol, we first need to consider which properties can be part of the namespace. In fact, these properties may have to be exposed in the resolution query. Hence, the set of admissible properties needs to be understood before a protocol can be developed. The answer to this question is actually straightforward since the resolution protocol should be flexible to support ANY property as part of the query interface. This even includes properties that are unique to a single namespace.

Nevertheless, from a practical standpoint, we believe that there exists a set of core properties that will be present in most namespaces. These properties are the Dublin Core metadata properties [DC]. Although it should not be required that a namespace supports any of these properties, we expect them to play a key role in standard implementations. We expect that many namespaces will standardize on the Dublin Core properties to increase interoperability. At a minimum, we recommend namespace implementers to study the Dublin Core elements before introducing new and proprietary vocabulary. Lastly and to illustrate the use of the query protocol, we will use the Dublin Core properties in all our examples for the remaining of this document.

To formalize our discourse, we define a Common Name namespace as a set metadata. A record in the Common Name namespace database is represented as a metadata vector (or a point) in that space with each meta property defining a new dimension (or axis). Thus, a namespace can be represented as a set of tuples {(P1, P2, ..., Pm)} where Pi (i=1,m) are the meta properties defined by the namespace. An element of the namespace is written as a tuple E (e1, e2, ..., em) where ei (i=1,m) is the value of the ith property for the namespace element.

As an example, let us consider a namespace for Web pages with 4 properties. Let these four properties be: CN:CN (the Common Name), DC:IDENTIFIER (the Dublin Core identifier element, the URI for the resource), DC:LANGUAGE (the Dublin Core language element) and DC:DESCRIPTION (the Dublin Core description element). An XML namespace [XMLNAMESPACES] for Common Names (prefixed CN) is presented in the next section. DC is the XML namespace prefix for the Dublin Core properties. This namespace can be represented as {(CN:CN, DC:IDENTIFIER, DC:LANGUAGE, DC:DESCRIPTION)}. Using our notation, an example of an entry for this namespace is ("BMW", "en", http://www.bmw.com, "The homepage for BMW, German car maker.").

2. A simple abstract protocol for the resolution service:

This section describes a generic resolution service for Common Name namespaces. An abstract protocol for the resolution service is proposed. The protocol assumes that all namespaces must implement a minimum of two properties: the CN and the URI properties.

The resolution service is responsible to resolve a query into a list of meta entries or namespace elements. For that purpose, the query interface is explicitly limited and kept as simple as possible. The resolution query interface is defined as a conjunctive query of property-value pairs. The properties allowed in the query are called the query parameters. The resolution protocol presented requires the CN to always be exposed in the query interface. This constraint guarantees that a first order implementation of the protocol exclusively relying on the CN will always work across all namespace implementations. Other properties present in the namespace can be added to the query interface. The namespace authority is solely responsible for deciding which properties to publicly expose as part of the query interface of

the resolution service. The mechanisms for publishing and discovering query properties is described in section 3.

Using the notations introduced previously, for a namespace of m meta properties {(P1, P2, ..., Pm)}, a query Q can be formalized as the tuple Q of property-value pairs:

Q = (Pq1 = v1, Pq2 = v2, ..., Pqi = vi) where (1 <= i <= m) and for each i, 1 <= qi <= m. Each vi is the desired value of the (qi)th property for an entry of the namespace to match the resolution query.

For such a query, the resolution service will return a set of r tuples where each tuple Ej (ej1, ...ejm), (1 <= j <= r) is such that for each qi (1 <= i <= m), ejqi = vi. In other terms, each tuple Ej of the result set satisfies all the conjuncts in the query. Since the response is a results set of metadata, we suggest that the response to a resolution query always be contained in the RDF data model. One encoding of the RDF data model is based on XML [RDF]

Note the following explicit restrictions for the resolution service query interface: The query is limited to conjunctive atomic queries where an atomic query is in the form "property name = property value". The only operator supported by this simple query interface is the "equal" operator. The actual semantic of "equality" will depend on the qualified property (the property type).

For instance, the equality between two Common Names will require more than a byte to byte straight comparison (e.g. UNICODE string equivalence). Beyond the resolution service, we anticipate other services such as "search" services. One can guess that these services will extend the query language to include more elaborate forms of queries (such as Boolean queries and relevance ranking).

For our namespace example {(CN:NAME, DC:IDENTIFIER, DC:LANGUAGE, DC:DESCRIPTION)}, the query for the resource associated with the Common Name "BMW" in the English language will be written as:

Q = (CN:CN = "BMW", DC:LANGUAGE = "en").

For such query, the response returned by the resolver would be a list of namespace elements such as:

```
{
  ("BMW", "en", http://www.bmw.com, "The homepage for BMW."),
  ("BMW", "de", http://www.bmw.de, "BMW Deutschland.")
}
```

Note that the CN XML namespace is introduced in more details in section 2 of this document and the exact form of the RDF response is presented in section 4.

3. An RDF schema for common name namespaces

To support the dynamic definition of all parameters involved in the resolution protocol, we introduce a formal RDF schema [RDFSCHEMAS]. This schema formally defines the Common Name as a core RDF property for CN namespaces.

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/TR/WD-rdf-syntax#"
  xmlns:rdfs="http://www.w3.org/TR/WD-rdf-schema#">

<rdfs:comment>This defines the Common Name (CN) as a resource
property</rdfs:comment>

<rdf:Description ID="cn">
  <rdf:type rdf:resource="http://www.w3.org/TR/WD-rdf-
  syntax#Property"/>
  <rdf:label>Common Name</rdf:label>
</rdf:Description >
```

Furthermore, to enable the dynamic discovery of query and response properties, the schema introduces the concepts of resolution Query and resolution Response.

```
<rdfs:comment>A superclass for all resolution
queries.</rdfs:comment>

<rdf:Description ID="Query">
  <rdf:type rdf:resource="http://www.w3.org/TR/WD-rdf-
  syntax#Class"/>
  <rdfs:subClassOf rdf:resource="http://www.w3.org/TR/WD-rdf-
  schema#Resource"/>
</rdf:Description>

<rdfs:comment>A superclass for all resolution
responses.</rdfs:comment>

<rdf:Description ID="Response">
  <rdf:type rdf:resource="http://www.w3.org/TR/WD-rdf-
  syntax#Class"/>
  <rdfs:subClassOf rdf:resource="http://www.w3.org/TR/WD-rdf-
  schema#Resource" />
</rdf:Description>
```

The Query class encapsulates the resolution query whereas the Response class encapsulates the resolution response. The CN is a required query property for ALL resolution services. The DC:IDENTIFIER is the only required property for the response. To expose more properties in the query or the response each namespace can extend (subclass) the Query or the Response class.

To allow for a precise definition of the query and response parameters, the RDF schema introduces the "parameters" property. This property is scoped to the Query and Response classes. Its value is an unordered list of admissible property classes.

```
<rdfs:comment>The parameters property allows a namespace to
specify the list of properties that belong to the query or to the
response of their resolution service.</rdfs:comment>

<rdf:Description ID="parameters">
  <rdf:type rdf:resource="http://www.w3.org/TR/WD-rdf-
  syntax#Property"/>
  <rdf:domain rdf:resource="#Query"/>
  <rdf:domain rdf:resource="#Response"/>
  <rdf:range rdf:resource="http://www.w3.org/TR/WD-rdf-
  syntax#Class"/>
</rdf:Description>
```

Using this new property, we can now translate the resolution query and response interface constraints into simple RDF statements about the Query and Response classes:

```
<rdfs:comment>The following statement makes the CN property part
of all resolution query interfaces.</rdfs:comment>

<rdf:Description rdf:about="Query">
  <parameters>
    <rdf:bag>
      <rdf:li rdf:resource="#CN" />
    </rdf:bag>
  </parameters>
</rdf:Description>

<rdfs:comment>The following statement makes the DC:IDENTIFIER
property part of all resolution responses.</rdfs:comment>

<rdf:Description rdf:about="Response">
  <parameters>
    <rdf:bag>
```

```
      <rdf:li rdf:resource =
      "http://purl.org/metadata/dublin_core#Identifier" />
    </rdf:bag>
  </parameters>
</rdf:Description>
```

Lastly, we complete the CN RDF schema with the notion of ResolutionService. The ResolutionService class encapsulates the main parameters of an actual implementation of the resolution service. Its first property is the queryInterface. The queryInterface defines the class to be used by the client in order to form a valid resolution query. Thus, the queryInterface is a property whose value is of Class Query (or a subclass). The queryInterface property allows a namespace authority to define the

subclass of Query to use with their resolution service, hence to express which other properties beside the CN should be part of the resolution query interface. The second property of ResolutionService is the responseInterface. Symmetrically, the responseInterface is a property whose value is of Class Response. It allows a namespace authority to fully specify the format of the resolution response.

```
<rdfs:comment>A class for all Resolution services</rdfs:comment>

<rdf:Description ID="ResolutionService">
  <rdf:type rdf:resource="http://www.w3.org/TR/WD-rdf-
  syntax#Class"/>
  <rdfs:subClassOf rdf:resource="http://www.w3.org/TR/WD-rdf-
  schema#Resource"/>
</rdf:Description>

<rdfs:comment>An instance of the class ResolutionService uses the
queryInterface property to specify to the client which Query class
to use for resolution queries. The value of this property is a
Class (Query or a subclass of Query)</rdfs:comment>

<rdf:Description ID="queryInterface">
  <rdf:type rdf:resource="http://www.w3.org/TR/WD-rdf-
  syntax#Property"/>
  <rdf:domain rdf:resource="#ResolutionService"/>
  <rdf:range rdf:resource="http://www.w3.org/TR/WD-rdf-
  syntax#Class"/>
</rdf:Description>

<rdfs:comment>An instance of the class ResolutionService uses the
responseInterface property to specify to the client which Response
class to expect for the response. The value of this property is a
Class (Response or a subclass of Response)</rdfs:comment>

<rdf:Description ID="responseInterface">
```

```
  <rdf:type rdf:resource="http://www.w3.org/TR/WD-rdf-
  syntax#Property"/>
  <rdf:domain rdf:resource="#ResolutionService"/>
  <rdf:range rdf:resource="http://www.w3.org/TR/WD-rdf-
  syntax#Class"/>
</rdf:Description>

</rdf:RDF>
```

The presented RDF schema defines the core objects and vocabulary for all CN namespaces. Using this schema, each namespace can now create its own RDF file. This new RDF file is owned, published and maintained by the namespace authority. It is called the namespace file. The idea behind the namespace file is to allow each namespace organization to expose the specifics of their own namespace. This includes the type of properties that are part of the namespaces, the properties exposed in the resolution query or

response interface, and the implementation parameters for the resolution service (see section 4 of this document for implementation parameters).

The following example shows how a namespace can use the RDF namespace file to expose any set of properties in the query interface of its resolution service. Let us consider our namespace example: {(CN:CN, DC:IDENTIFIER, DC:LANGUAGE, DC:DESCRIPTION)}. Let us also assume that we would like to expose {CN:CN, DC:LANGUAGE} in the resolution query interface. Lastly, let us assume that the response is the default Resolution response as specified in the CN schema. Under these conditions, the namespace file will be written:

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/TR/WD-rdf-syntax#"
  xmlns:rdfs="http://www.w3.org/TR/WD-rdf-schema#"
  xmlns:dc="http://purl.org/metadata/dublin_core#"
  xmlns:cn="http://www.ietf.org/CN/rdf-schema#" >

<rdfs:comment>A subclass of Query to expose the dc:language
property as part of the resolution query interface</rdfs:comment>

<rdf:Description ID="MyQuery">
<rdf:type rdf:resource="http://www.w3.org/TR/WD-rdf-
syntax#Class"/>
<rdfs:subClassOf rdf:resource="http://www.ietf.org/CN/rdf-
 schema#Query" />
</rdf:Description>

<rdfs:comment>Stipulate that the DC:LANGUAGE property is supported
by MyQuery. Since MyQuery is a subclass of Query, cn:cn is
supported as well.</rdfs:comment>
```

```
<rdf:Description rdf:about="MyQuery">
  <parameters>
    <rdf:bag>
      <rdf:li rdf:resource =
      "http://purl.org/metadata/dublin_core#Language" />
      </rdf:bag>
  </parameters>
</rdf:Description>

<rdfs:comment>Create an instance of
ResolutionService</rdfs:comment>

<cn:ResolutionService ID="MyResolutionService" />

<rdfs:comment>Specify to the client that it must pass resolution
queries of class MyQuery and expect responses of class Response.
</rdfs:comment>

<rdf:Description rdf:about="MyResolutionService">
  <cn:queryInterface rdf:resource="#MyQuery" />
  <cn:responseInterface rdf:resource="http://www.ietf.org/CN/rdf-
  schema#Response" />
</rdf:Description>

</rdf:RDF>
```

It is clear that this mechanism provides infinite flexibility for customizing the query and response interfaces. In particular, the properties exposed by the query or the response are not limited to the Dublin Core but can involve any XML vocabulary. The only constraint lies in the publication of the RDF namespace file, a relatively straightforward task as demonstrated in our example.

4. A schema compliant RDF representation for the resolution query and response.

Now that an RDF schema has been defined, it is possible to formally express both the resolution query and response as RDF. The XML encoding of this representation is the recommended transfer mechanism between the client and the server.

For a namespace that is satisfied with the default query interface (as specified in the RDF schema), the format of the query is extremely simple and the same for all namespaces. For example, the "default" query for the Common Name "BMW cars" can be written:

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/TR/WD-rdf-syntax#"
  xmlns:cn="http://www.ietf.org/CN/schema"
  xmlns:dc="http://purl.org/metadata/dublin_core#">
  <cn:Query ID="25364.32" />
  <rdf:description rdf:about="25364.32">
    <cn:cn>BMW</cn:cn>
  </rdf:description>
</rdf:RDF>
```

Using the Response class formalized by the RDF schema, the response to our example query can simply be written as:

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/TR/WD-rdf-syntax#"
  xmlns:cn="http://www.ietf.org/CN/schema"
  xmlns:dc="http://purl.org/metadata/dublin_core#">
  <cn:Response ID="25364.32" />
  <rdf:description rdf:about="25364.32">
    <rdf:bag>
      <rdf:li>
          <rdf:description about="http://www.bmw.com">
              <cn:cn>BMW</cn:cn>
              <dc:language>en</dc:language>
              <dc:description>BMW Homepage</dc:description>
          </rdf:description>
      </rdf:li>
        <rdf:li>
        <rdf:description about="http://www.bmw.de">
              <cn:cn>BMW</cn:cn>
              <dc:language>de</dc:language>
              <dc:description>BMW Deutschland</dc:description>
          </rdf:description>
        </rdf:li>
    </rdf:bag>
  </rdf:description>
</rdf:RDF>
```

Since this format will be understood by all namespaces, a trivial client implementation of the protocol is possible. The default query form makes it very simple to use any resolver on the network, independently of the namespace. At the same time, if the targeted namespace has published its own RDF file derived from the schema, and if the client wants to take advantage of the richer APIs, a more advanced implementation is possible.
For example, let us assume now that a client wants to access the resolver from our example namespace. Let us also assume that it wants to use the language property in the query in order to restrict the results set to Web pages in the English language. This translates into the query Q = (CN:CN = "BMW", DC:LANGUAGE = "en"). Q can now be precisely encoded using the class MyQuery found in the RDF namespace file

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/TR/WD-rdf-syntax#"
  xmlns:cn="http://www.ietf.org/CN/schema"
  xmlns:dc="http://purl.org/metadata/dublin_core#"
  xmlns="http://www.myCNExample.com/CN/schema">
  <MyQuery ID="25364.32" />
  <rdf:description rdf:about="25364.32">
    <cn:cn>BMW</cn:cn>
    <dc:language>en</dc:language>
  </rdf:description>
</rdf:RDF>
```

In conclusion, we have defined a formal RDF representation for the resolution query and response. The resolver client can infer the exact RDF representation of the query and response at runtime. This exact representation is derived from an RDF file and can differ from one namespace to another (through the definition of a specific subclass of cn:Query and cn:Response).

5. An implementation of the abstract protocol using HTTP/XML/RDF.

In section 2, we have defined an abstract protocol that can support a wide variety of implementations. In particular, it is worth noting that the abstract resolution protocol could be implemented using LDAP, WHOIS or HTTP. Because of its popularity and simplicity, we anticipate that HTTP will provide the most popular implementation. For that reason, this section specifically looks at a concrete implementation based on HTTP.

HTTP is the transport mechanism. The resolution request is expressed in RDF, encoded in XML and embedded within the HTTP request. To answer the request, a resolver generates an RDF document which is returned to the client within the HTTP response.

In order for a namespace to specify the concrete types of implementations that it supports, the ResolutionService class introduced in section 3 of this document is made an abstract superclass. Hence, the Implementation class becomes a base class from which specific subclasses can be derived in order to describe a concrete implementation. In the case of the HTTP implementation of the resolution service, the concrete subclass is called HTTPResolutionService. This class allows the specification of physical parameters such as the HTTP server URI or the Web server port number.

In the CN schema, these notions translate into:

```
<rdfs:comment>A concrete subclass of ResolutionService to support
an HTTP implementation of the protocol.</rdfs:comment>

<rdf:Description ID="HTTPResolutionService">
  <rdf:type rdf:resource="http://www.w3.org/TR/WD-rdf-
  syntax#Class"/>
  <rdfs:subClassOf rdf:resource="#ResolutionService" />
</rdf:Description>

<rdfs:comment>The resolver port number, a property of the
 HTTPResolutionService class</rdfs:comment>

<rdf:Description ID="portNumber">
  <rdf:type rdf:resource="http://www.w3.org/TR/WD-rdf-
  syntax#Property"/>
  <rdf:domain rdf:resource="#HTTPResolutionService" />
  <rdf:range rdf:resource="http://www.w3.org/TR/WD-rdf-
  syntax#Literal" />
</rdf:Description>
```

To understand how these new notions will be used, let us consider our example namespace. Let us assume that the resolution service for this namespace is implemented using HTTP. To indicate an HTTP implementation of the service to the resolution clients, the namespace administrator simply adds an instance of HTTPResolutionService to the RDF namespace file:

```
<rdfs:comment>Here, we fully specify the resolution service for
our example namespace. In particular, it includes the description
for the HTTP implementation</rdfs:comment>

<cn:HTTPResolutionService ID="MyHTTPImplementation" />

<rdf:Description rdf:about="MyHTTPImplementation">
  <cn:queryInterface rdf:resource="#MyQuery" />
  <cn:responseInterface rdf:resource="http://www.ietf.org/CN/rdf-
  schema#Response" />
  <dc:identifier>http://www.mycnexample.com/cgi-
  bin/resolver</dc:identifier>
  <cn:portNumber>80</cn:portNumber>
</rdf:Description>
```

This is significant since a client can dynamically discover the type of implementation supported by a specific namespace. It is then up to this client to make a decision regarding which implementation it should access (HTTP, LDAP, WHOIS,...). Everything can happen at runtime since the client can learn the implementation specific parameters as well as the resolution query interface for each particular namespace.

This shows that the resolution process can be extremely dynamic with no predefined knowledge of the namespace. In fact, the only required knowledge for the client is the URI for the RDF namespace file. It is therefore conceivable that these URIs would be registered with a central authority.

6. Conclusion

This document presented a protocol for Common Name resolution. A resolution query has been defined as a conjunctive query of key value pairs. An RDF schema has been introduced. The schema introduces the minimal vocabulary and concepts necessary to provide enough flexibility for dealing with multiple namespaces. Using this schema, the resolution query and response have been expressed using RDF. The notion of a distributed namespace file (one per namespace) has been discussed. This file allows the introduction of arbitrary vocabulary for the resolution query and response interfaces. Furthermore, it allows for multiple concrete implementations of the resolution protocol. Leveraging this file, a resolution client can dynamically discover the implementation details of a targeted resolution service. The client can then use this knowledge to adapt to the implementation. A default form and implementation of the protocol that does not require access to a namespace file has also been presented.

7. References

[URI] Uniform Resource Identifiers (URI): Generic Syntax; Berners-Lee, Fielding, Masinter, Internet Draft Standard August, 1998; RFC2396.

[METADATA] Metadata and Resource description; http://www.w3.org/Metadata/

[DC] The Dublin Core initiative; http://purl.oclc.org/metadata/dublin_core

[XMLNAMESPACES] Namespaces in XML; Bray, Hollander, Layman eds, World Wide Web Consortium Working Draft; http://www.w3.org/TR/1998/PR-xml-names-19981117.

[RDFSCHEMAS] Resource Description Framework (RDF) Model and Syntax; http://www.w3.org/TR/WD-rdf-syntax/

[RDFSchema] Resource Description Framework (RDF) Schemas; Brickley, Guha, Layman eds., World Wide Web Consortium Working Draft; http://www.w3.org/TR/WD-rdf-schema

[XML] Extensible Markup Language (XML) 1.0; World Wide Web Consortium Recommendation; http://www.w3.org/TR/REC-xml.

8. Authors Addresses:

Nicolas Popp Centraal Corporation 811 Hansen Way PO Box 50750 Palo Alto CA 94303 0750 U.S.A. Phone: (650)846-3615 Fax: (650)858-0454 Email: nico@centraal.com

Michael Mealling Network Solutions 505 Huntmar Park Drive Herndon,
VA 22070 voice: (770) 935-5492 fax: (703) 742-9552 email:
michaelm@netsol.com