Network Working Group
Request for Comments: 3981
Category: Standards Track

A. Newton
VeriSign, Inc.
M. Sanz
DENIC eG

January 2005

IRIS: The Internet Registry Information Service (IRIS) Core Protocol

Status of This Memo

Copyright Notice

Abstract

This document describes an application layer client-server protocol for a framework to represent the query and result operations of the information services of Internet registries. Specified in the Extensible Markup Language (XML), the protocol defines generic query and result operations and a mechanism for extending these operations for specific registry service needs.

Table of Contents

1. Introduction

The specification outlined in this document is based on the functional requirements described in CRISP [17].

1.1. Use of XML

This document describes the specification for the Internet Registry Information Service (IRIS), an XML text protocol intended to describe the query types and result types of various registry information services. IRIS is specified by using the Extensible Markup Language

(XML) 1.0 as described in [2], XML Schema notation as described in [4] and [5], and XML Namespaces as described in [3].

1.2. General Concepts

Each kind of Internet registry is identified by a registry type. The identifier for a registry type is a Uniform Resource Name (URN) used within the XML instances to identify the XML schema that formally describes the set of queries, results, and entity classes allowed within that type of registry.

The structure of these URNs makes no assumptions or restrictions on the types of registries they identify. Therefore, IRIS may support multiple registry types of a disparate or similar nature; it is only a matter of definition. For instance, a single registry type may be defined for domain name registries, and multiple registry types for the various IP address registries.

A registry information server may handle queries and serve results for multiple registry types. Each registry type that a particular registry operator serves is a registry service instance.

IRIS and the XML schema formally describing IRIS do not specify any registry, registry identifier, or knowledge of a particular service instance or set of instances. IRIS is a specification for a framework with which these registries can be defined, used and, in some cases, interoperate. The framework merely specifies the elements for registry identification and the elements that must be used to derive queries and results.

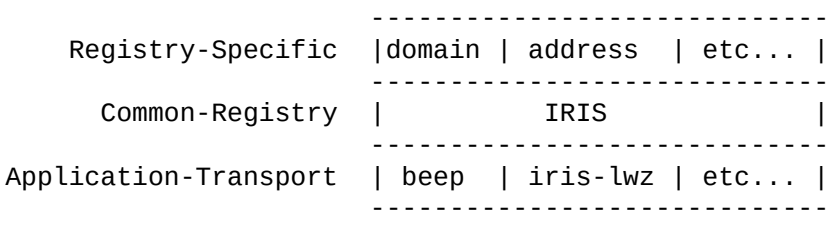
This framework allows a registry type to define its own structure for naming, entities, queries, etc., through the use of XML namespaces and XML schemas (hence, a registry type MUST be identified by the same URI that identifies its XML namespace). To be compliant, a registry type's specification must extend from this framework.

The framework defines certain structures that can be common to all registry types, such as references to entities, search continuations, and entity classes. A registry type may declare its own definitions for all of these, or it may mix its derived definitions with the base definitions.

IRIS defines two types of referrals: an entity reference and a search continuation. An entity reference indicates specific knowledge about an individual entity, and a search continuation allows distributed searches. Both referrals may span differing registry types and instances. No assumptions or specifications are made about the roots, bases, or meshes of entities.

1.3. Framework Layers

The IRIS framework can be thought of as having three layers.



In this figure, "beep" refers to the Blocks Extensible Exchange Protocol (BEEP) (see [20]), and "iris-lwz" refers to a theoretical UDP binding that uses compression.

The differing layers have the following responsibilities:

Registry-Specific :: defines queries, results, and entity classes of a specific type of registry. Each specific type of registry is identified by a URN.
Common-Registry :: defines base operations and semantics common to all registry types such as search sets, result sets, and referrals. It also defines the syntaxes for talking about specific registry types.
Application-Transport :: defines the mechanisms for authentication, message passing, connection and session management, etc. It also defines the URI syntax specific to the application-transport mechanism.

1.4. Definitions

For clarity, the following definitions are supplied:

- registry type -- A registry serving a specific function, such as a domain registry or an address registry. Each type of registry is assigned a URN.

- registry schema -- The definition for a registry type specifying the queries, results, and entity classes.

- authority -- A reference to the server or set of servers containing information.

- resolution method -- The technique used to locate an authority.

- entity class -- A group of entities with a common type or common set of characteristics.

- entity name -- The identifier used to refer to a single entity within an entity class.

- entity reference -- A pointer to an entity composed of an authority, an optional resolution method, a registry type, an entity class, and an entity name. One type of entity reference is the IRIS URI (defined in Section 7).

The terms "derivative", "derive", and "derivation" are used with the same meaning for deriving one type of element from another as specified in XML_SS [5].

1.5. Further Reading

Appendix B contains text answering the question, "Why IRIS?".

This document describes the structure at the core of IRIS. The following documents describe the other aspects of IRIS relevant to CRISP [17]: iris-beep [1] and iris-dreg [18].

2. Document Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14, RFC 2119 [8].

3. Protocol Identification

The root element of all request XML instances MUST be <request>. The root element of all response XML instances MUST be <response>. These elements identify the start of the IRIS elements, the XML namespace used as the identifier for IRIS, and, optionally, the location of the schema. These elements and the associated closing tag MUST be applied to all requests and responses sent by both clients and servers.

The use of the schema location attribute 'xsi:schemaLocation' is OPTIONAL with respect to this specification, and IRIS implementations MAY resolve it to retrieve the schema or MAY use a locally cached version of the schema.

Versioning of the IRIS protocol is the responsibility of the application-transport layer but MUST be associated with the XML namespace [3] URI representing IRIS. A change in this URI indicates a change of the underlying schema and, therefore, a new version of the protocol (and vice versa).

4. Exchange Description

This section describes the request and response exchanges of the protocol. The descriptions contained within this section refer to XML elements and attributes and their relation to the exchange of data within the protocol. These descriptions also contain specifications outside the scope of the formal XML syntax. Therefore, this section will use terms defined by RFC 2119 [8] to describe the specification outside the scope of the formal XML syntax. While reading this section, please reference Section 6 for details on the formal XML syntax.

4.1. Request Format

A <request> element contains an optional <control> element and a set of <searchSet> elements.

The <searchSet> elements enable a client to query a particular registry type by using the URN identifying the registry type. This can be found in one of its two children: <lookupEntity> and <query>.

The <lookupEntity> element describes the lookup of an entity in a specific registry. This element has three attributes: 'registryType', 'entityClass', and 'entityName'. The 'registryType' attribute contains the registry identifier for the registry type in which the lookup operation will take place. The 'entityClass' attribute contains the token identifying the index for which the lookup operation will take place, and the 'entityName' attribute contains the name of the entity to look up.

The <query> element is abstract and may not legally appear in an XML instance. It provides the base type that registry schemas will use to define derived query types. This derivation mechanism is described in Section 4.3.

Each <searchSet> may also contain a <bag> element. When this element appears as a child of <searchSet>, it MUST NOT contain the 'id' attribute. For a description of the <bag> element, see Section 4.4.

The <control> element may contain one child element of any XML namespace. This child element allows a client to signal a server for special states or processing. An example of one such <control> child element may be found in Section 4.3.8.

4.2. Response Format

The <response> element contains an optional <reaction> element, a set of <resultSet> elements, and an optional <bags> element.

The <resultSet> elements are responses to a <searchSet> request. The contents of this element contain an <answer> element, an optional <additional> element, and error elements, if applicable.

The children of the <answer> element are of the following types:

- <result> is an abstract element and may not be legally placed in an XML instance. It provides the base type to be used by registry schemas to define derived result types. This derivation mechanism is described in Section 4.3.

- <entity> is an element specifying an entity reference. See Section 4.3.5.

- The <searchContinuation> element specifies a query referral. Its one child is any element derived from <query> (see Section 4.3.1). To direct the query to a referent server, <searchContinuation> has a mandatory 'authority' attribute and an optional 'resolution' attribute. The <searchContinuation> element may also contain a 'bagRef' attribute. For a description of the 'bagRef' attribute, see Section 4.4.

When following entity references and search continuations, clients SHOULD only follow an <entity> or <searchContinuation> response once. Failure to do so may result in the client process getting stuck in a never-ending query loop, commonly known as a referral loop.

The <additional> element only contains <result> elements, as described above. This element allows a server to indicate to a client results that were not specifically queried but that are related to the queried results, thus enabling the client to display this distinction to a user properly. The <additional> element use is optional.

The following elements, which represent error conditions, may be returned:

- <insufficientResources> -- The corresponding query requires resources unobtainable by the server.

- <invalidName> -- A name given in a query is not syntactically correct.

- <invalidSearch> -- Parameters of the corresponding query are not semantically meaningful.

- <queryNotSupported> -- The corresponding query is not supported by this server.

- <limitExceeded> -- The corresponding query requires more resources than allowed.

- <nameNotFound> -- The name given in a query does not match a known entity.

- <permissionDenied> -- The authentication given does not allow access to a specific result entry.

- <bagUnrecognized> -- The contents of a bag were unrecognized. See Section 4.4.

- <bagUnacceptable> -- The contents of a bag were not and never will be acceptable. See Section 4.4.

- <bagRefused> -- The contents of a bag were not acceptable at this time. See Section 4.4.

- A derivative of <genericCode>, as described in Section 4.3.

The <resultSet> section is divided into the <answer> and <additional> sections to allow easier processing and navigation of the results by a client. Servers MUST return the direct answers to queries in the <answer> element and MAY return results in the <additional> element for which a reference has been made in the <answer> element. Results in the <additional> element MUST have been referenced in the <answer>, either as direct children of the <answer> element or as deeper descendants of the <answer> element.

This serves two purposes. First, it may eliminate a requery by the client for references contained in the <answer> element. Second, it distinguishes between results that are a direct result of a query and those that would have been returned had the client followed the appropriate referrals, thus hinting how clients could process or display the returned results. For instance, clients constructing complex displays with tree navigation widgets will know that results in the <answer> element should all be directly beneath the root node of the tree, while results in the <additional> element are leaf nodes of those produced from the <answer> element.

A <reaction> element (child of <response>) is a response to a <control> element, and provides a means for a server to advise a client of the effect of a <control> element.

The <bags> element (child of <response>) is optional. It contains <bag> elements, and the contents of each <bag> element constitute one element in any XML namespace. Each <bag> element has an 'id' attribute, which is referenced by the 'bagRef' attribute of entity

references (<entity>) and search continuations (<searchContinuation>). See Section 4.4.

4.3. Extension Framework

Because the IRIS schema defines only one query type, and two stand-alone result types, and does not define a registry structure, it is of limited use by itself. Extension of IRIS is accomplished through the use of a base IRIS schema, as defined in XML_SD [4] and XML_SS [5], and through extension of it by schemas constructed on top of IRIS.

4.3.1. Derived Elements

The XML Schema definition of IRIS requires schemas of registry types to derive element types from base types in the IRIS definition. The registry schemas MUST derive elements to define typed queries and results.

While the IRIS schema definition does not prohibit the derivation of any elements, registry schemas SHOULD restrict the derivations to the following types:

- <query> -- As defined, this element contains no content and has no valid attributes. It is abstract and therefore only its derivatives appear in XML instances. Registry schemas derive from this element to define the queries allowed.

- <result> -- As defined, this element contains no content and has five valid attributes: 'authority', 'resolution' (optional), 'registryType', 'entityClass', 'entityName', and 'temporaryReference' (optional, see Section 4.3.6). It is abstract and therefore only its derivatives appear in XML instances. Registry schemas derive from this element to define results that may be returned from a query.

- <genericCode> -- As defined, this element is an instance of <codeType>. It contains the optional elements <explanation> and <language>, which further describe the nature of the error.

- <entity> -- Identifies a reference to an entity. Registry schemas SHOULD use elements derived from <entity> but MAY use <entity> directly. The advantage of deriving from <entity> vs. direct use is the chance to define the name of the element and to use that name descriptively -- for instance, as the role the entity plays with respect to another entity. See Section 4.3.5.

- o <seeAlso> -- Indicates a reference to an entity that has indirect association with a parent element representing an entity. This element is derived from the <entity> element (Section 4.3.5). Registry schemas MAY derive from this element or MAY use it directly.

4.3.2. Registry Type Identifier Requirements

The identifier for a registry type and the XML namespace identifier used by the XML Schema describing the registry MUST be the same. These identifiers MUST be restricted to a URN [7] registered in the 'ns' class of the IANA registry governed by XML_URN [9]. These identifiers are case insensitive.

This is a restriction on XML_NS [3], which specifies that an XML namespace identifier is any valid URI [6].

These identifiers MAY be abbreviated to the part following the class component and its separator of the URN. For example, the full URN "urn:ietf:params:xml:ns:dreg1" may be abbreviated to "dreg1".

In use with IRIS, this abbreviation MUST NOT be used inside of XML instances in which the XML Schema [4] specifies the use of a URI for schema identification or where XML_NS [3] specifies the use of a URI for XML namespace identification.

4.3.3. Entity Classes

IRIS provides entity classes to help avoid collisions with entity names within any given registry type. Their specification in queries also allows server implementations to narrow search or lookup scopes quickly to a single index.

For instance, the entity name "192.0.2.0" might refer to separate entities in the "name-server" and "network" classes. The entity "192.0.2.0" in the "name-server" class may refer to the name server host that is also multi-homed by address 192.0.2.255 and known in DNS as "ns.example.com", whereas the entity "192.0.2.0" in the "network" class may refer to the network 192.0.2/30.

IRIS defines two default entity classes of "local" and "iris", which MUST NOT be redefined. These entity classes MUST be valid in all registry types.

The "local" class is reserved for entities defined locally by a server operator and does not denote any particular type of entity. A lookup in this entity class MAY result in an entity reference or search continuation. For example, "iris:dreg1//example.com/local/

myhosts" may result in a search continuation yielding the nameservers for example.com.

The "iris" class is reserved for entities specific to a particular service instance. It MUST contain the following entity names (see Section 4.3.4):

- "id", which yields a result of <serviceIdentification> (see Section 4.3.7.1).
- "limits", which yields a result of <limits> (see Section 4.3.7.2). This entity class MAY contain other locally defined entities as well.

The names of entity classes in a registry schema are of type token, as defined by XML_SD [4]. Their case sensitivity MUST be defined by the definition of the registry type. In general, they SHOULD be case insensitive.

4.3.4. Names of Entities

The names of entities in a registry schema are of type token, as defined by XML_SD [4].

Names of entities SHOULD be unique within an instance of any particular entity class within a registry. Two entities SHOULD NOT have the same name, but a single entity MAY be known by multiple names. In situations where a single name may result in two entities, the registry schema SHOULD make allowances by defining result types that contain entity references to both entities (e.g., "example.com" can refer to both the domain example.com and the host example.com). However, this type of conflict SHOULD generally be avoided by the proper use of entity classes.

The case sensitivity of entity names is dependent on the entity class in which they reside. The definition of a registry type MUST specify the case sensitivity for entity names. A registry type MAY define the entity names of differing entity classes as having different case sensitivity.

4.3.5. References to Entities

The element <entity> allows references to entities in result sets, either as a direct child of <resultSet> or within a more complex structure deriving from <result>. The <entity> element is defined by 'entityType'. Registry schemas SHOULD define elements derived from <entity> when referencing entities but may use the <entity> element directly. Deriving a new element allows a registry schema to use the

name of the new element to signify the relationship the referenced entity has with the referrer. A derivative of <entity> MUST NOT be used as a substitute when the <entity> element is declared (such as in the <answer> section of the <resultSet>).

The <entity> element (and elements of type 'entityType') can have child elements of <displayName> with an optional 'language' attribute. These are provided so that servers may provide clients with a more human-friendly description of the entity reference. This is often useful to users navigating referral structures.

The <entity> element (and its derivations) have the following attributes:

- 'authority', 'resolution' (optional), 'registryType', 'entityClass', and 'entityName' -- These attributes specify where the entity may be found.

- 'temporaryReference' -- This attribute is optional. See Section 4.3.6.

- 'referentType' -- This attribute contains the expected type of the entity being referenced and may contain the word "ANY" or a qualified XML name. Unlike the other attributes of <entity>, this attribute is qualified and declared in the IRIS XML namespace. Therefore it will also be qualified with the prefix associated with the IRIS XML namespace (e.g., 'iris:referentType'). This allows clients to recognize entity references using an element derived from <entity>.

- 'bagRef' -- This attribute is optional. If present, it must contain an XML identifier to a <bag> element in the <bags> section of the result set. For a description of the 'bagRef' attribute, see Section 4.4.

4.3.6. Temporary Entities

Instances may exist in which an entity reference needs to be temporary. For example, a particular type of result may only have one unique key. If that key contains semantic meaning that may not be exposed to all users, a synthetic key will have to be substituted.

Furthermore, there may be times when data in the data store is not normalized in the same manner as that expressed by the registry schema. In the registry schema, objects of type A may reference objects of type B. But in the data store, objects of type A may contain objects of type B. Again, a synthetic key will have to be temporarily produced.

To support such use cases, results and entity references can be declared temporary by using the 'temporaryReference' attribute. This attribute is of type boolean [4] and has a default value of "false". It is optional for <result> derivatives and elements of type 'entityType'.

When this attribute is used, the entity reference data (e.g., 'entityClass', 'entityName') is only valid within the response in which it appears and may not be consistent with subsequent responses. A server MUST include the referent of any temporary entity reference in the <additional> section of the same <resultSet>

4.3.7. <result> Derived Elements

The base IRIS framework contains three elements directly derived from the <result> element for use by any registry type.

4.3.7.1. <serviceIdentification>

An example of a <serviceIdentification> result:

```
<serviceIdentification
  authority="example.com" registryType="dreg1"
  entityClass="iris"
  entityName="id" >
  <authorities>
    <authority> example.com </authority>
    <authority> example.net </authority>
    <authority> example.org </authority>
  </authorities>
  <operatorName>
    Internet Assigned Numbers Authority
  </operatorName>
  <eMail>
    iana@iana.org
  </eMail>
</serviceIdentification>
```

The <serviceIdentification> element is provided to allow IRIS clients to reference IRIS service instances. It contains the following elements:

- <authorities> -- This element contains one or more <authority> elements. Each <authority> element contains a URI authority component for which the server has results. Although a server MAY only return a partial list of its authority areas, depending on operator policy, it MUST return the authority for which the client has requested.

- <operatorName> -- This element contains the name of the operator of the server.

- <eMail> -- These optional elements contain email addresses of the operator of the service instance.

- <phone> -- These optional elements contain phone numbers of the operator of the service instance.

- <seeAlso> -- See Section 4.3.1 for its definition.

4.3.7.2. <limits>

An example of a <limits> result:

```
<limits
  authority="example.com" registryType="dreg1"
  entityClass="iris" entityName="limits">
  <totalQueries>
    <perHour>2</perHour>
    <perDay>15</perDay>
  </totalQueries>
  <totalResults>
    <perHour>25</perHour>
    <perDay>200</perDay>
  </totalResults>
  <totalSessions>
    <perHour>2</perHour>
    <perDay>15</perDay>
  </totalSessions>
</limits>
```

The <limits> element provides a mechanism allowing a server to inform a client of the limits it may encounter from overuse of the service. The contents describe the service limitations to a client at the current level of access. The contents of this element are as follows:

- <totalQueries> -- This element describes the total number of queries that the server will accept. The children of this element indicate this number per unit of time. The children are <perSecond>, <perMinute>, <perHour>, and <perDay>. Each child MUST only appear once as a child of <totalQueries>, but more than one child MAY be present. For example, a server could indicate that it will accept 15 queries a minute but only 60 queries a day.

- <totalResults> -- This element describes the total number of results that the server will send to a client. The children of this element indicate this number per unit of time in the same manner as <totalQueries>.

- <totalSessions> -- This element describes the total number of sessions that the server will accept from a client. The children of this element indicate this number per unit of time in the same manner as <totalQueries>. The definition of a session is defined the by application transport layer.

- <otherRestrictions> -- This element describes other restrictions that may only be expressible outside of the structured syntax of the other child elements of <limits>. This element may have optional <description> child elements, each with a mandatory 'language' attribute.

- <seeAlso> -- These elements are provided to reference other entities, such as a <simpleEntity> (Section 4.3.7.3) describing a published policy. See <seeAlso> (Section 4.3.1).

All of these child elements are optional, and a server may express that it has no limits by using a <limits> element with no content (e.g., <limits authority=... />).

4.3.7.3. <simpleEntity>

An example of a <simpleEntity> result:

```
<simpleEntity
  authority="example.com" registryType="dreg1"
  entityClass="local"
  entityName="notice" >
  <property name="legal" language="en">
    Example.com is reserved according to RFC 2606.
  </property>
</simpleEntity>
```

The <simpleEntity> element is provided so that service operators may make simple additions to other entities without deriving entirely new registry types. Its definition allows service operators to reference it from other entities (using, for instance, a <seeAlso> element). The <simpleEntity> is meant to represent name and value pairs of strings, allowing each pair to be associated with a specific language qualifier and an optional URI pointing to more information.

Clients may easily display such information in a two-column table. Applications using binary data or richer data structures are out of scope for this element. When such usage scenarios arise, a client will likely need specific knowledge to handle such data, thus calling the need for a new registry type into question.

4.3.8. <control> and <reaction> Elements

The <control> (Section 4.1) and <reaction> (Section 4.2) elements allow the client to request from the server special states for the processing of queries. The intent of these elements is to allow extensibility so that some jurisdictions may adopt policies for query processing without requiring re-versioning of IRIS or any registry type.

This document defines one control, <onlyCheckPermissions>, and its requisite reaction, <standardReaction>, for compliance with CRISP [17].

When a client sends an <onlyCheckPermissions> control, it is only asking the server to check to see whether adequate permissions are available to execute the queries in the associated request. A server MUST respond to this control with a <standardReaction> element.

The <standardReaction> element provides a server with a standard means to respond to controls (it may be used by other controls, but this is left to their definition). It contains four children:

- <controlAccepted> -- the processing or state needed by the control has been accepted.
- <controlDenied> -- the processing or state needed by the control has been denied (a transient failure).
- <controlDisabled> -- the processing or state needed by the control cannot be activated (a permanent failure).
- <controlUnrecognized> -- the control is not recognized (a permanent failure).

If <onlyCheckPermissions> is rejected, then the server MUST return all appropriate result sets (i.e., for every search set in the request), but all result sets MUST be empty of results and MUST contain no errors (a reaction is not part of a result set and is therefore not a result set error). This control applies to all search sets or none of them; therefore a server MUST issue a rejection if <onlyCheckPermissions> cannot be accepted for all search sets in a request.

An example of an IRIS XML exchange using these elements follows:

```
C: <?xml version="1.0"?>
C: <request xmlns="urn:ietf:params:xml:ns:iris1"
C:   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
C:
C:   <control>
C:     <onlyCheckPermissions />
C:   </control>
C:
C:   <searchSet>
C:
C:     <lookupEntity
C:       registryType="dreg1"
C:       entityClass="local"
C:       entityName="AUP" />
C:
C:   </searchSet>
C:
C: </request>

S: <?xml version="1.0"?>
S: <response xmlns="urn:ietf:params:xml:ns:iris1"
S:           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
S:
S:   <reaction>
S:     <standardReaction>
S:       <controlAccepted />
S:     </standardReaction>
S:   </reaction>
S:
S:   <resultSet>
S:     <answer>
S:
S:       <simpleEntity
S:         authority="example.com" registryType="dreg1"
S:         entityClass="local" entityName="AUP" >
S:         <property name="legal" language="en">
S:           It is illegal to use information from this service
S:           for the purposes of sending unsolicited bulk email.
S:         </property>
S:       </simpleEntity>
S:
S:     </answer>
S:   </resultSet>
S:
S: </response>
```

4.4. Relay Bags

IRIS employs bags to allow a server to relay information to a referent server via the client. These bags are generated by the queried server, passed to the client as opaque data, and then passed to the referent server for processing. The contents of the bags are not defined by IRIS, and the client MUST NOT make any assumptions about the contents of a bag when relaying it from one server to another.

When a server returns a result set to a client, the <response> element may contain a <bags> child element. This child element contains one or more <bag> elements. Each of these MUST contain an 'id' attribute containing the XML data type ID. Entity references and search continuations that have to specify a bag to be used when they are followed MUST have a 'bagRef' attribute containing the XML data type IDREF. See Section 4.2. This allows the response to specify a bag only once but allows each entity reference or search continuation (in all result sets) to have a distinct bag, as needed.

When following an entity reference or search continuation that specifies the use of a bag, the client MUST include the referenced bag in the search set as a child of the <searchSet> element. See Section 4.1.

See Section 4.2 for the list of errors a server may return to a client when a bag is received. A server MUST NOT ignore a bag when it is received. In case a bag cannot be recognized or accepted, one of the errors from Section 4.2 MUST be returned.

An example of an IRIS XML exchange using these elements follows:

```
C: <?xml version="1.0"?>
C: <request xmlns="urn:ietf:params:xml:ns:iris1"
C:   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
C:
C:   <searchSet>
C:
C:     <bag>
C:       <simpleBag xmlns="http://example.com/">
C:         XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
C:       </simpleBag>
C:     </bag>
C:
C:     <lookupEntity
C:       registryType="dreg1"
C:       entityClass="local"
C:       entityName="AUP" />
```

```
C:
C:   </searchSet>
C:
C: </request>

S: <?xml version="1.0"?>
S: <response xmlns="urn:ietf:params:xml:ns:iris1"
S:           xmlns:iris="urn:ietf:params:xml:ns:iris1"
S:           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
S:
S:   <resultSet>
S:     <answer>
S:
S:       <entity authority="example.com" bagRef="x1"
S:         registryType="dreg1"
S:         entityClass="local" entityName="AUP"
S:         iris:referentType="ANY" >
S:         <displayName language="en">
S:           Acceptable Usage Policy
S:         </displayName>
S:       </entity>
S:
S:     </answer>
S:   </resultSet>
S:
S:   <bags>
S:
S:     <bag id="x1">
S:       <simpleBag xmlns="http://example.com/">
S:         AAAAB3NzaC1yc2EAAAABIwAAAIEA0ddD+W3Agl0Lel98G1r77fZ
S:       </simpleBag>
S:     </bag>
S:
S:   </bags>
S: </response>
```

5. Database Serialization

This section describes a method for serializing IRIS registry entities. The descriptions contained within this section refer to XML elements and attributes and their relation to this serialization process. These descriptions also contain specifications outside the scope of the formal XML syntax. This section will use terms defined by RFC 2119 [8] to describe these. While reading this section, please reference Section 6 for needed details on the formal XML syntax.

A database of IRIS entities can be serialized to file storage with XML [2] by using the IRIS defined <serialization> element. This element contains <result> element derivatives and <serializedReferral> elements.

Derivatives of the <result> element are entities. Servers loading these entities MUST place the entity in the entity classes specified by the elements 'registryType', 'entityClass', and 'entityName' attributes and in any entity classes the entities may apply according to explicitly defined children of that element. For instance, if a registry type has two entity classes "foo" and "bar" and a <result> derivative has the attributes entityClass="foo" and entityName="one" and a child element <bar>two</bar>, the server is to enter that entity into the entity class "foo" as the name "one" and into the entity class "bar" as the name "two".

Servers loading entities as serialized derivatives of the <result> element MAY translate the authority attribute. Servers will likely have to do this if the authority for the entity has changed.

<serializedReferral> elements allow the serialization of explicit entity references and search continuations. This element has a child <source> element containing the 'authority', 'resolution' (optional), 'registryType', 'entityClass', and 'entityName' attributes. The attributes of this element are used to signify the entity that can be referenced to yield this referral.

As mentioned above, there may be times when a server needs to translate the authority attribute of a loaded entity. Implementations must also beware of this need for referrals. During deserialization, servers MUST change the authority attribute of a referral (either <entity> or elements derived from <entity> or <source> child of <serializedReferral>) to contain a valid authority of the server if the serialized attribute is empty. During serialization, servers and their related processes MUST leave the authority attribute empty for referrals in which the referent is an entity for which the server answers queries.

The following is an example of serialized IRIS:

```
<iris:serialization
  xmlns:iris="urn:ietf:params:xml:ns:iris1"
  xmlns="urn:ietf:params:xml:ns:iris1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
```

```
    <serviceIdentification
      authority="iana.org" registryType="dreg1"
      entityClass="iris"
      entityName="id" >
      <authorities>
        <authority> iana.org </authority>
      </authorities>
      <operatorName>
        Internet Assigned Numbers Authority
      </operatorName>
      <eMail>
        dbarton@iana.org
      </eMail>
      <seeAlso
        iris:referentType="iris:simpleEntity"
        authority="iana.org" registryType="dreg1"
        entityClass="local"
        entityName="notice">
        <displayName language="en">
          Legal Notice
        </displayName>
      </seeAlso>
    </serviceIdentification>

    <serializedReferral>
      <source
        authority="example.com" registryType="dreg1"
        entityClass="iris"
        entityName="id"/>
      <entity
        iris:referentType="iris:serviceIdentification"
        authority="iana.org" registryType="dreg1"
        entityClass="iris" entityName="id"/>
    </serializedReferral>

    <simpleEntity
      authority="iana.org" registryType="dreg1"
      entityClass="local"
      entityName="notice" >
      <property name="legal" language="en">
        Please use the net wisely!
      </property>
    </simpleEntity>

  </iris:serialization>
```

6. Formal XML Syntax

IRIS is specified in XML Schema notation. The formal syntax presented here is a complete schema representation of IRIS suitable for automated validation of IRIS XML instances.

```
<?xml version="1.0"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:iris="urn:ietf:params:xml:ns:iris1"
        targetNamespace="urn:ietf:params:xml:ns:iris1"
        elementFormDefault="qualified" >

  <annotation>
    <documentation>
      Internet Registry Information Service (IRIS) Schema v1
    </documentation>
  </annotation>

  <!-- ========================================= -->
  <!--                                           -->
  <!-- The Transactions                          -->
  <!--                                           -->
  <!-- ========================================= -->

  <element name="request">
    <complexType>
      <sequence>
        <element
          name="control"
          type="iris:controlType"
          minOccurs="0"
          maxOccurs="1" />
        <element
          name="searchSet"
          type="iris:searchSetType"
          minOccurs="1"
          maxOccurs="unbounded" />
      </sequence>
    </complexType>
  </element>

  <element name="response">
    <complexType>
      <sequence>
        <element
          name="reaction"
          type="iris:reactionType"
          minOccurs="0"
```

```
            maxOccurs="1" />
          <element
            name="resultSet"
            type="iris:resultSetType"
            minOccurs="1"
            maxOccurs="unbounded" />
          <element
            name="bags"
            type="iris:bagsType"
            minOccurs="0"
            maxOccurs="1" />
        </sequence>
      </complexType>
    </element>

    <!-- ======================================== -->
    <!--                                          -->
    <!-- Search Sets and Result Sets              -->
    <!--                                          -->
    <!-- ======================================== -->

    <complexType
      name="searchSetType" >
      <sequence>
        <element
          name="bag"
          type="iris:bagType"
          minOccurs="0"
          maxOccurs="1" />
        <choice>
          <element
            name="lookupEntity"
            type="iris:lookupEntityType" />
          <element
            ref="iris:query" />
        </choice>
      </sequence>
    </complexType>

    <complexType
      name="resultSetType" >
      <sequence>
        <element
          name="answer"
          minOccurs="1"
          maxOccurs="1">
          <complexType>
            <sequence>
```

```
            <element
              ref="iris:result"
              minOccurs="0"
              maxOccurs="unbounded" />
            <element
              ref="iris:entity"
              minOccurs="0"
              maxOccurs="unbounded" />
            <element
              ref="iris:searchContinuation"
              minOccurs="0"
              maxOccurs="unbounded" />
          </sequence>
        </complexType>
      </element>
      <element
        name="additional"
        minOccurs="0"
        maxOccurs="1">
        <complexType>
          <sequence>
            <element
              ref="iris:result"
              minOccurs="1"
              maxOccurs="unbounded" />
          </sequence>
        </complexType>
      </element>
      <choice
        minOccurs="0"
        maxOccurs="1" >
        <element
          name="insufficientResources"
          type="iris:codeType" />
        <element
          name="invalidName"
          type="iris:codeType" />
        <element
          name="invalidSearch"
          type="iris:codeType" />
        <element
          name="queryNotSupported"
          type="iris:codeType" />
        <element
          name="limitExceeded"
          type="iris:codeType" />
        <element
          name="nameNotFound"
```

```
            type="iris:codeType" />
          <element
            name="permissionDenied"
            type="iris:codeType" />
          <element
            name="bagUnrecognized"
            type="iris:codeType" />
          <element
            name="bagUnacceptable"
            type="iris:codeType" />
          <element
            name="bagRefused"
            type="iris:codeType" />
          <element
            ref="iris:genericCode"/>
        </choice>
      </sequence>
    </complexType>

    <!-- ======================================== -->
    <!--                                          -->
    <!-- Controls and Reactions                   -->
    <!--                                          -->
    <!-- ======================================== -->

    <complexType
      name="controlType">
      <sequence>
        <any
          namespace="##any"
          processContents="skip"
          minOccurs="1"
          maxOccurs="1" />
      </sequence>
    </complexType>

    <complexType
      name="reactionType">
      <sequence>
        <any
          namespace="##any"
          processContents="skip"
          minOccurs="1"
          maxOccurs="1" />
      </sequence>
    </complexType>

    <!-- ======================================== -->
```

```
<!--                                                -->
<!-- Queries and Lookups                            -->
<!--                                                -->
<!-- ============================================== -->

<complexType
  name="queryType" />

<element
  name="query"
  type="iris:queryType"
  abstract="true" />

<complexType
  name="lookupEntityType" >
  <attribute
    name="registryType"
    type="anyURI"
    use="required" />
  <attribute
    name="entityClass"
    type="token"
    use="required" />
  <attribute
    name="entityName"
    type="token"
    use="required" />
</complexType>

<!-- ============================================== -->
<!--                                                -->
<!-- Results                                        -->
<!--                                                -->
<!-- ============================================== -->

<complexType
  name="resultType">
  <attribute
    name="authority"
    use="required"
    type="token" />
  <attribute
    name="resolution"
    type="token" />
  <attribute
    name="registryType"
    use="required"
    type="anyURI" />
```

```
      <attribute
        name="entityClass"
        use="required"
        type="token" />
      <attribute
        name="entityName"
        use="required"
        type="token" />
      <attribute
        name="temporaryReference"
        default="false"
        type="boolean" />
    </complexType>

    <element
      name="result"
      type="iris:resultType"
      abstract="true" />

    <!-- ======================================== -->
    <!--                                          -->
    <!-- Errors                                   -->
    <!--                                          -->
    <!-- ======================================== -->

    <complexType
      name="codeType">
      <sequence
        minOccurs="0"
        maxOccurs="unbounded">
        <element
          name="explanation">
          <complexType>
            <simpleContent>
              <extension
                base="string">
                <attribute
                  use="required"
                  name="language"
                  type="language" />
              </extension>
            </simpleContent>
          </complexType>
        </element>
      </sequence>
    </complexType>
    <element
      name="genericCode"
```

```
    type="iris:codeType"
    abstract="true" />

  <!-- ======================================== -->
  <!--                                          -->
  <!-- Entity References and                    -->
  <!-- Search Continuations                     -->
  <!--                                          -->
  <!-- ======================================== -->

  <complexType
    name="entityType">
    <sequence>
      <element
        name="displayName"
        minOccurs="0"
        maxOccurs="unbounded">
        <complexType>
          <simpleContent>
            <extension
              base="string">
              <attribute
                name="language"
                use="required"
                type="language" />
            </extension>
          </simpleContent>
        </complexType>
      </element>
    </sequence>
    <attribute
      name="authority"
      use="required"
      type="token" />
    <attribute
      name="resolution"
      type="token" />
    <attribute
      name="registryType"
      use="required"
      type="anyURI" />
    <attribute
      name="entityClass"
      use="required"
      type="token" />
    <attribute
      name="entityName"
      use="required"
```

```
        type="token" />
      <attribute
        name="referentType"
        use="required"
        form="qualified"
        type="iris:referentTypeType" />
      <attribute
        name="temporaryReference"
        default="false"
        type="boolean" />
      <attribute
        name="bagRef"
        type="IDREF" />
    </complexType>

    <element
      name="entity"
      type="iris:entityType" />

    <simpleType
      name="referentTypeType">
      <union
        memberTypes="QName iris:anyLiteralType" />
    </simpleType>

    <simpleType
      name="anyLiteralType">
      <restriction
        base="string">
        <enumeration
          value="ANY" />
      </restriction>
    </simpleType>

    <complexType
      name="searchContinuationType">
      <sequence>
        <element ref="iris:query" />
      </sequence>
      <attribute
        name="bagRef"
        type="IDREF" />
      <attribute
        name="authority"
        type="token"
        use="required" />
      <attribute
        name="resolution"
```

```
        type="token" />
    </complexType>

    <element
      name="searchContinuation"
      type="iris:searchContinuationType" />

    <!-- ======================================== -->
    <!--                                          -->
    <!-- Bags                                     -->
    <!--                                          -->
    <!-- ======================================== -->

    <complexType
      name="bagsType">
      <sequence>
        <element
          name="bag"
          minOccurs="1"
          maxOccurs="unbounded">
          <complexType>
            <complexContent>
              <extension
                base="iris:bagType">
                <attribute
                  use="required"
                  name="id"
                  type="ID" />
              </extension>
            </complexContent>
          </complexType>
        </element>
      </sequence>
    </complexType>

    <complexType
      name="bagType">
      <sequence>
        <any
          namespace="##any"
          processContents="skip"
          minOccurs="1"
          maxOccurs="1" />
      </sequence>
    </complexType>
    <!-- ======================================== -->
    <!--                                          -->
    <!-- Derived Results for use with all         -->
```

```
<!-- registry types.                                  -->
<!--                                                  -->
<!-- ================================================ -->

<!--                                                  -->
<!-- See Also                                         -->
<!--                                                  -->

<element
  name="seeAlso"
  type="iris:entityType" />

<!--                                                  -->
<!-- Service Identification                           -->
<!--                                                  -->

<complexType
  name="serviceIdentificationType">
  <complexContent>
    <extension
      base="iris:resultType">
      <sequence>
        <element
          name="authorities"
          minOccurs="1"
          maxOccurs="1">
          <complexType>
            <sequence>
              <element
                name="authority"
                type="token"
                minOccurs="1"
                maxOccurs="unbounded" />
            </sequence>
          </complexType>
        </element>
        <element
          name="operatorName"
          type="string"
          minOccurs="0"
          maxOccurs="1" />
        <element
          name="eMail"
          type="string"
          minOccurs="0"
          maxOccurs="unbounded" />
        <element
          name="phone"
```

```
            type="string"
            minOccurs="0"
            maxOccurs="unbounded" />
          <element
            ref="iris:seeAlso"
            minOccurs="0"
            maxOccurs="unbounded" />
        </sequence>
      </extension>
    </complexContent>
  </complexType>

  <element
    name="serviceIdentification"
    type="iris:serviceIdentificationType"
    substitutionGroup="iris:result" />

  <!--                                                -->
  <!-- Limits                                         -->
  <!--                                                -->

  <complexType
    name="limitsType">
    <complexContent>
      <extension
        base="iris:resultType">
        <sequence>
          <element
            name="totalQueries"
            minOccurs="0"
            maxOccurs="1" >
            <complexType>
              <group
                ref="iris:timeLimitsGroup"
                minOccurs="1"
                maxOccurs="4" />
            </complexType>
          </element>
          <element
            name="totalResults"
            minOccurs="0"
            maxOccurs="1" >
            <complexType>
              <group
                ref="iris:timeLimitsGroup"
                minOccurs="1"
                maxOccurs="4" />
            </complexType>
```

```
            </element>
            <element
              name="totalSessions"
              minOccurs="0"
              maxOccurs="1" >
              <complexType>
                <group
                  ref="iris:timeLimitsGroup"
                  minOccurs="1"
                  maxOccurs="4" />
              </complexType>
            </element>
            <element
              name="otherRestrictions"
              minOccurs="0"
              maxOccurs="1">
              <complexType>
                <sequence>
                  <element
                    name="description"
                    minOccurs="0"
                    maxOccurs="unbounded">
                    <complexType>
                      <simpleContent>
                        <extension
                          base="string">
                          <attribute
                            name="language"
                            type="language"
                            use="required" />
                        </extension>
                      </simpleContent>
                    </complexType>
                  </element>
                </sequence>
              </complexType>
            </element>
            <element
              ref="iris:seeAlso"
              minOccurs="0"
              maxOccurs="unbounded" />
          </sequence>
        </extension>
      </complexContent>
    </complexType>
    <element
      name="limits"
      type="iris:limitsType"
```

```
      substitutionGroup="iris:result" />

    <group
      name="timeLimitsGroup">
      <choice>
        <element
          name="perSecond"
          type="nonNegativeInteger" />
        <element
          name="perMinute"
          type="nonNegativeInteger" />
        <element
          name="perHour"
          type="nonNegativeInteger" />
        <element
          name="perDay"
          type="nonNegativeInteger" />
      </choice>
    </group>

    <!--                                          -->
    <!-- Simple Entity                            -->
    <!--                                          -->

    <complexType
      name="simpleEntityType">
      <complexContent>
        <extension
          base="iris:resultType">
          <sequence>
            <element
              name="property"
              minOccurs="1"
              maxOccurs="unbounded">
              <complexType>
                <simpleContent>
                  <extension
                    base="string">
                    <attribute
                      name="name"
                      type="string"
                      use="required" />
                    <attribute
                      name="language"
                      type="language"
                      use="required" />
                    <attribute
                      name="uri"
```

```
                    type="anyURI" />
                </extension>
              </simpleContent>
            </complexType>
          </element>
        </sequence>
      </extension>
    </complexContent>
  </complexType>

  <element
    name="simpleEntity"
    type="iris:simpleEntityType"
    substitutionGroup="iris:result" />

  <!-- ======================================== -->
  <!--                                          -->
  <!-- Derived Controls and Reactions           -->
  <!--                                          -->
  <!-- ======================================== -->

  <!--                                          -->
  <!-- Only Check Permissions                   -->
  <!--                                          -->

  <element
    name="onlyCheckPermissions" >
    <complexType />
  </element>

  <!--                                          -->
  <!-- Standard Reaction                        -->
  <!--                                          -->

  <element
    name="standardReaction" >
    <complexType>
      <choice>
        <element
          name="controlAccepted">
          <complexType/>
        </element>
        <element
          name="controlDenied">
          <complexType/>
        </element>
        <element
          name="controlDisabled">
```

```
            <complexType/>
          </element>
          <element
            name="controlUnrecognized">
            <complexType/>
          </element>
        </choice>
      </complexType>
    </element>

    <!-- ======================================== -->
    <!--                                          -->
    <!-- Serialization                            -->
    <!--                                          -->
    <!-- ======================================== -->

    <complexType
      name="serializedReferralType">
      <sequence>
        <element name="source">
          <complexType>
            <attribute
              name="authority"
              use="required"
              type="token" />
            <attribute
              name="resolution"
              type="token" />
            <attribute
              name="registryType"
              type="anyURI"
              use="required" />
            <attribute
              name="entityClass"
              type="token"
              use="required" />
            <attribute
              name="entityName"
              type="token"
              use="required" />
          </complexType>
        </element>
        <choice>
          <element
            ref="iris:searchContinuation" />
          <element
            ref="iris:entity" />
        </choice>
```

```
      </sequence>
    </complexType>

    <element
      name="serialization">
      <complexType>
        <choice
          minOccurs="1"
          maxOccurs="unbounded">
          <element
            ref="iris:result" />
          <element
            name="serializedReferral"
            type="iris:serializedReferralType" />
        </choice>
      </complexType>
    </element>

  </schema>
```

Figure 8

7. The IRIS URI

The IRIS URI has a very rigid structure. All IRIS URIs have the same fields and look similar to users.

But the IRIS URIs are flexible because they allow different methods to be employed to find servers and allow the use of multiple transports (with BEEP being the default).

7.1. URI Definition

An IRIS URI [6] has the following general syntax.

iris:<registry>/<resolution>/<authority>/<class>/<name>

The full ABNF [11] follows, with certain values included from RFC 2396 [6] and RFC 2732 [15].

```
   iris-uri           = scheme ":" registry-urn "/"
                        [ resolution-method ] "/" authority
                        [ "/" entity-class "/" entity-name ]
   scheme             = "iris"
   authority          = // as specified by RFC2396
   registry-urn       = // as specified by IRIS
   resolution-method  = *(unreserved | escaped)
   entity-class       = *(unreserved | escaped)
```

```
entity-name        = *(unreserved | escaped)
unreserved         = // as specified by RFC2396
escaped            = // as specified by RFC2396
```

An IRIS URI MUST NOT be a relative URI. The resolution method, entity class, and entity name MUST be of the UTF-8 [12] character set encoded with "application/x-www-form-urlencoded", as specified by URL_ENC [14].

When the entity-class and entity-name components are not specified, the defaults "iris" and "id" MUST be implied. For example, "iris:dreg1//com" is interpreted as "iris:dreg1//com/iris/id".

When the resolution-method is not specified, the default is the direct resolution method described in Section 7.3.2.

7.2. Transport Specific Schemes

The "iris" scheme name is not application transport specific. The URI resolution process MAY determine the application transport. An example of such a process is direct resolution (Section 7.3.2), which uses the steps outlined in Section 7.3.3 to determine the application transport.

A mapping between an application transport and IRIS MAY define a scheme name signifying its use with the semantics of the IRIS URI.

The rules for determining which application transport to use are as follows:

- If an application transport specific scheme name is present, the application transport it signifies SHOULD be used if possible.
- If a client has a preferred transport and the resolution process allows for its use, the client MAY use that application transport.
- Otherwise, the default application transport specified by IRIS-BEEP [1] MUST be used.

7.3. URI Resolution

7.3.1. Registry Dependent Resolution

Interpretation and resolution of the authority component of an IRIS URI may be altered with the specification of a resolution-method in the URI. If no resolution-method component is specified in the URI, the default is the direct resolution method (see Section 7.3.2).

Alternate resolution methods MAY be specified by registry types. The identifiers for these methods MUST conform to the ABNF in Section 7.1.

7.3.2. Direct Resolution

In the direct resolution process, the authority component of an IRIS URI may only contain a domain name, a domain name accompanied by a port number, an IP address, or an IP address accompanied by a port number. The authority component of the scheme indicates the server or set of servers authoritatively responsible for a domain according to records in DNS (Section 7.3.3) if a domain is specified. If an IP address is specified, it indicates the specific server to be queried.

The rules for resolution are as follows:

- If the authority component is a domain name accompanied by a port number as specified by RFC 2396, the domain name is converted to an IP address via an A or AAAA record to the DNS.

- If the authority component is a domain name by itself, the service/transport location (Section 7.3.3) process is used. If this process produces no results, then the DNS is queried for the A or AAAA RRs corresponding to the domain name, and the port number used is the well-known port of the transport used according to Section 7.2.

- If the authority component is an IP address, then the DNS is not queried, and the IP address is used directly. If the port number is present, it is used directly; otherwise, the port number used is the well-known port of the transport used according to Section 7.2.

The use of an IPv6 address in the authority component MUST conform to RFC 2732 [15].

7.3.3. Transport and Service Location

The direct resolution method (Section 7.3.2) uses the profiled use of the NAPTR and SRV resource records defined in S-NAPTR [10] to determine both the location of a set of servers for a given service and the set of possible transports that may be used. It is RECOMMENDED that any resolution method not making explicit use of the direct resolution process should use S-NAPTR [10] in whatever process it does define.

S-NAPTR [10] requires an application service label. The direct resolution method (Section 7.3.2) uses the abbreviated form of the registry URN as the application service label. Other resolution methods MAY specify other application service labels.

See Appendix A for sample uses of S-NAPTR.

7.4. IRIS URI Examples

Here are some examples of IRIS URIs and their meaning:

- iris:dreg1//example.com/domain/example.com
  - Finds a server authoritative for "example.com" according to the rules of direct resolution (Section 7.3.2).
  - The server is asked for "example.com" in the "domain" index, or entity class, of the "dreg1" registry.

- iris:dreg1//example.com
  - Finds a server authoritative for "example.com" according to the rules of direct resolution (Section 7.3.2).
  - The server is asked for "id" in the "iris" index, or entity class, of the "dreg1" registry.

- iris:dreg1//com/domain/example.com
  - Finds a server authoritative for "com" according to the rules of direct-resolution (Section 7.3.2).
  - The server is asked for "example.com" in the "domain" index, or entity class, of the "dreg1" registry.

- iris:dreg1//192.0.2.1:44/domain/example.com
  - Following the rules of direct-resolution (Section 7.3.2), the server at IP address 192.0.2.1 on port 44 is queried by using BEEP.
  - The server is asked for "example.com" in the "domain" index, or entity class, of the "dreg1" registry.

- iris.lwz:dreg1//192.0.2.1:44/domain/example.com
  - Following the rules of direct-resolution (Section 7.3.2), the server at IP address 192.0.2.1 on port 44 is queried by using a lightweight application transport.
  - The server is asked for "example.com" in the "domain" index, or entity class, of the "dreg1" registry.

- iris.beep:dreg1//com/domain/example.com
  - Finds a server authoritative for "com" according to the rules of direct-resolution (Section 7.3.2).
  - Uses the BEEP application transport.
  - The server is asked for "example.com" in the "domain" index, or entity class, of the "dreg1" registry.

- iris:dreg1/bottom/example.com/domain/example.com
  - Finds a server authoritative for "example.com" according to the rules of the resolution method 'bottom' as defined by the registry type urn:ietf:params:xml:ns:dreg1.
  - The application transport used is determined by the 'bottom' resolution method.
  - The server is asked for "example.com" in the "domain" index, or entity class, of the "dreg1" registry.

- iris.beep:dreg1/bottom/example.com/domain/example.com
  - Finds a server authoritative for "example.com" according to the rules of the resolution method 'bottom' as defined by the registry type urn:ietf:params:xml:ns:dreg1.
  - Uses the BEEP application transport.
  - The server is asked for "example.com" in the "domain" index, or entity class, of the "dreg1" registry.

8. Checklists

8.1. Registry Definition Checklist

Specifications of registry types MUST include the following explicit definitions:

- Formal XML syntax deriving from the IRIS XML.

- An identifying registry URN.

- Any registry specific resolution methods.

- A registration of the abbreviated registry URN as an application service label for compliance with S-NAPTR [10]. Note that this is a different IANA registry than the registry type URN IANA registry.

- A list of well-known entity classes.

- A statement regarding the case sensitivity of the names in each entity class.

8.2. Transport Mapping Checklist

Specifications of transport mappings MUST include the following explicit definitions:

- A URI scheme name specific to the transport.
- An application protocol label for compliance with S-NAPTR [10]. See Section 7.3.3. Note that although this is a different IANA registry than the URI scheme name IANA registry, it is RECOMMENDED that they be the same string of characters.
- The set of allowable character set encodings for the exchange of XML (see Section 9).
- The set of security mechanisms.

9. Internationalization Considerations

IRIS is represented in XML. XML processors are obliged to recognize both UTF-8 and UTF-16 [12] encodings. XML provides for mechanisms to identify and use other character encodings by means of the "encoding" attribute in the <xml> declaration. Absence of this attribute or a byte order mark (BOM) indicates a default of UTF-8 [13] encoding. Thus, for compatibility reasons and per RFC 2277 [16], use of UTF-8 [13] is RECOMMENDED with IRIS.

The complete list of character set encoding identifiers is maintained by IANA at [21].

The application-transport layer MUST define a common set of character set encodings to be understood by both client and server.

Localization of internationalized strings may require additional information from the client. Entity definitions SHOULD use the "language" type defined by XML_SD [4] to aid clients in the localization process. See Section 4.3.7.3 for an example.

10. IANA Considerations

This document uses a proposed XML namespace and schema registry specified in XML_URN [9]. Accordingly, the following registration information is provided for the IANA:

- URN/URI:
  - urn:ietf:params:xml:ns:iris1
- Contact:
  - Andrew Newton <andy@hxr.us>
  - Marcos Sanz <sanz@denic.de>
- XML:
  - The XML Schema specified in Section 6

11. Security Considerations

The IRIS XML layer provides no authentication or privacy facilities of its own. It relies on the application-transport layer for all of these abilities. Application-transports should explicitly define their security mechanisms (see Section 8.2).

Referral IRIS registry results may contain entity lookups and search continuations that result in a client query operation against another registry service. Clients SHOULD NOT use authentication credentials and mechanisms subject to replay attacks to conduct subsequent entity lookups and search continuations.

12. References

12.1. Normative References

[1] Newton, A. and M. Sanz, "Using the Internet Registry Information Service (IRIS) over the Blocks Extensible Exchange Protocol (BEEP)", RFC 3983, January 2005.

[2] World Wide Web Consortium, "Extensible Markup Language (XML) 1.0", W3C XML, February 1998, <http://www.w3.org/TR/1998/REC-xml-19980210>.

[3] World Wide Web Consortium, "Namespaces in XML", W3C XML Namespaces, January 1999, <http://www.w3.org/TR/1999/REC-xml-names-19990114>.

[4] World Wide Web Consortium, "XML Schema Part 2: Datatypes", W3C XML Schema, October 2000, <http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/>.

[5] World Wide Web Consortium, "XML Schema Part 1: Structures", W3C XML Schema, October 2000, <http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/>.

[6] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, August 1998.

[7] Moats, R., "URN Syntax", RFC 2141, May 1997.

[8] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[9] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.

[10] Daigle, L. and A. Newton, "Domain-based Application Service Location Using SRV RRs and the Dynamic Delegation Discovery Service (DDDS)", RFC 3958, January 2005.

[11] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997.

[12] The Unicode Consortium, "The Unicode Standard, Version 3", ISBN 0-201-61633-5, 2000, <The Unicode Standard, Version 3>.

[13] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

[14] Connolly, D. and L. Masinter, "The 'text/html' Media Type", RFC 2854, June 2000.

[15] Hinden, R., Carpenter, B., and L. Masinter, "Format for Literal IPv6 Addresses in URL's", RFC 2732, December 1999.

[16] Alvestrand, H., "IETF Policy on Character Sets and Languages", BCP 18, RFC 2277, January 1998.

12.2. Informative References

[17] Newton, A., "Cross Registry Internet Service Protocol (CRISP) Requirements", RFC 3707, February 2004.

[18] Newton, A. and M. Sanz, "IRIS: A Domain Registry (dreg) Type for the Internet Registry Information Service (IRIS)", RFC 3982, January 2005.

[19] Daigle, L., "WHOIS Protocol Specification", RFC 3912, September 2004.

[20] Rose, M., "The Blocks Extensible Exchange Protocol Core", RFC 3080, March 2001.

URIs

[21] <http://www.iana.org/assignments/character-sets>

Appendix A. S-NAPTR and IRIS Uses

A.1. Example of S-NAPTR with IRIS

This section shows an example of S-NAPTR [10] use by IRIS. In this example, there are two registry types: REGA and REGB. There are also two IRIS application transports: iris-a and iris-b. Given this, the use of S-NAPTR offers the following:

1. A means by which an operator can split the set of servers running REGA from the set of servers running REGB. This is to say, the operator is able to split out the set of servers serving up data for REGA from the set of servers serving up data for REGB.

2. A means by which an operator can distinguish the set of servers running iris-a from the set of servers running iris-b. This is to say, the operator is able to split out the set of servers running protocol iris-a serving REGA and REGB data from the set of servers running protocol iris-b serving REGA and REGB data.

3. A means by which an operator can specify which set of the servers to operate and which set of the above servers to delegate to another operator.

To implement the first feature, the operator deploys the following in his or her DNS zone:

```
example.com.
;;        order  pref  flags service               re replacement
IN NAPTR  100    10    ""    "REGA:iris-a:iris-b"  "" rega.example.com
IN NAPTR  100    10    ""    "REGB:iris-a:iris-b"  "" regb.example.com
```

To implement the second feature, the operator then adds the following in their DNS zone:

```
rega.example.com.
;;        order  pref flags service        re  replacement
IN NAPTR  100    10   "s"   "REGA:iris-a"  ""  _iris-a._udp.example.com
regb.example.com.
IN NAPTR  100    10   "s"   "REGA:iris-b"  ""  _iris-b._tcp.example.com

_iris-a._udp.example.com.
;;        pref  weight port  target
IN SRV    10    0      34    big-a.example.com.
IN SRV    20    0      34    small-a.example.com.
```

```
_iris-b._tcp.example.com.
;;        pref  weight port  target
IN SRV    10    0      34    big-b.example.com.
IN SRV    20    0      34    small-b.example.com.
```

Finally, an operator may decide to operate the REGA services while delegating the REGB services to somebody else. Here is how that is done:

```
example.com.
;;       order pref flags service              re replacement
IN NAPTR 100   10   ""    "REGA:iris-a:iris-b" "" rega.example.com
IN NAPTR 100   10   ""    "REGB:iris-a:iris-b" "" somebodyelse.com
```

Or the operator may decide to operate REGB services under the iris-a protocol/transport while delegating the REGB services under the iris-b protocol/transport to somebody else.

```
example.com.
;;       order pref flags service       re replacement
IN NAPTR 100   10   ""    "REGB:iris-a:iris-b" "" regb.example.com
IN NAPTR 100   10   "s"   "REGB:iris-a" "" _iris-a._udp.example.com
IN NAPTR 100   10   "s"   "REGB:iris-b" "" _iris-b._tcp.somebodyelse.com

_iris-a._udp.example.com.
;;        pref  weight port  target
IN SRV    10    0      34    big-a.example.com.
IN SRV    20    0      34    small-a.example.com.
```

Note that while this last example is possible, it is probably not advisable because of the operational issues involved in synchronizing the data between example.com and somebodyelse.com. It is provided here as an example of what is possible.

A.2. Using S-NAPTR for Cohabitation

Given the examples in Appendix A.1, the use of S-NAPTR could be part of a transition strategy for cohabitation of protocols solving the problems of CRISP [17].

For example, the type of data for domain information could be given the application service label of "DREG1". Given this, the service field of an S-NAPTR compliant NAPTR record could read

"DREG1:whois:iris-beep"

This service field conveys that domain data, as defined by CRISP, is available via both the iris-beep protocol and the whois protocol. The whois application protocol label refers to RFC 954 [19].

Appendix B. IRIS Design Philosophy

Beyond the concrete arguments that could be placed behind a thoughtful analysis of the bits flying across the ether, there are other abstract reasons for the development of IRIS. This section attempts an explanation.

B.1. The Basic Premise

IRIS has been designed as a directory service for public-facing registries of Internet resources. The basic premise is this:

- A client should be able to look up any single piece of data from any type of registry. This lookup should involve a straight-forward and consistent definition for finding the registry and should entail a hit to a single data index in the registry.

- Anything more, such as searches up and down the DNS tree to find the registry or searches across multiple indexes in a registry, requires a client with special knowledge of the data relationships contained within a registry.

Therefore, IRIS does the following:

- It specifies the basic schema language used by all registries to specify their schemas.

  - It provides the basic framework for a registry to make a reference to an entity in another type of registry.

And, therefore, IRIS does not do the following:

- It does not specify a common query language across all types of registries. A common query language imposed across multiple types of registries usually results in the disabling of certain functions by a server operator in order to meet acceptable levels of performance, leaving a common query language that does not commonly work.

- It does not impose any relationship between sets of data in any type of registry, such as specifying a tree. There are many types of Internet resources, and they do not all share the same style of

relationship with their contained sets of data. When it is not a natural fit, an imposition of a common relationship is often a concern and not a benefit.

B.2. The Lure of a Universal Client

The design premise of IRIS signifies that, for directory services, there is no such thing as a universal client (or that if there is one, it is commonly called the "web browser").

For IRIS, the closest thing to a universal client is one that may "look up" data and may be able to display the data in a rudimentary fashion. For a client to be able to "search" data or display it in a truly user-friendly manner, it must have specific knowledge about the type of data it is retrieving.

Attempts to outfit a universal client with a common query language are also not very useful. A common query language may be applied to a specific problem domain, which would require a user to have expertise in both the common query language and the problem domain. In the end, the outcome is usually the development of a client specific to the problem domain but saddled with translation of the user's desires and the lowest-common-denominator aspect of the query language.

B.3. Server Considerations

As mentioned above, IRIS was designed for the directory service needs of public-facing registries. In this light, certain aspects of more generalized directory services are a hindrance in an environment that does not have the same control and safety considerations as a managed network.

For instance, a common query language can provide great flexibility to both the power user and the abusive user. An abusive user could easily submit a query across multiple indexes with partial values. Such a query would have no utility other than to cause denial of service to other users. To combat this, a service operator must restrict the types of queries that cause harm to overall performance, and this act obsoletes the benefit of a common query language.

Another consideration for server performance is the lack of a required data relationship. Because sets of data often have differing relationships, a one-size-fits-all approach does not fit well with all types of registries. In addition, public-facing services tend to have service level requirements that cannot reasonably be met by transforming complete data stores from a native format into a format enforcing an artificial set of relationships.

To combat these issues, operators of public-facing services tend to create their own custom query parsers and back-end data stores. But doing so brings into question the use of a generalized directory service.

Finally, IRIS is built upon a set of standard technological layers. This allows service operators to switch components to meet the needs of their particular environment.

B.4. Lookups, Searches, and Entity Classes

IRIS supports both lookups and searches. Conceptually, the difference between the two is as follows:

> A "lookup" is a single query with a discrete value on a single index.
>
> Anything more, such as partial value queries, queries across multiple indexes, or multiple queries to a single index is a "search".

Lookups are accomplished through the defined query <lookupEntity>. This query specifies a discrete name, called the entity name, to be queried in a single index, called the entity class. Therefore, implementations may consider a type of registry to be composed of multiple indexes, one for each defined entity class.

There are no standard searches in IRIS. Each type of registry defines its own set of searches.

B.5. Entities References, Search Continuations, and Scope

Due to its effect on client behavior and the side effects such behavior may have on servers, IRIS makes a clear distinction between entity references (<entity>) and search continuations (<searchContinuation>). It is not an add-on, but a fundamental core of the protocol.

The distinction is very important to a client:

> "Go look over there and you will find what you seek." "Go look over there and you may find what you seek, or you may find some other stuff, or you may find nothing."

Finally, because IRIS makes no assumptions about and places no requirements on the relationship of data in a registry, this also extends to data of the same registry type spread across multiple authority areas. This means that IRIS makes no requirements as to

the scope of entity references or search continuations. The scope is determined by what the registry type needs and by what the registry type allows a service operator.

Appendix C. Acknowledgments

The terminology used in this document to describe namespaces and namespaces of namespaces is now much clearer thanks to the skillful debate tactics of Leslie Daigle. Previously, it was much more confusing. In addition, Leslie has provided great insight into the details of URIs, URNs, and NAPTR/SRV resource records.

Many other technical complexities were proved unnecessary by David Blacka and have been removed. And his IRIS implementation has helped smooth out the rougher edges.

Authors' Addresses

Andrew L. Newton
VeriSign, Inc.
21345 Ridgetop Circle
Sterling, VA 20166
USA

Phone: +1 703 948 3382
EMail: anewton@verisignlabs.com; andy@hxr.us
URI: http://www.verisignlabs.com/

Marcos Sanz
DENIC eG
Wiesenhuettenplatz 26
D-60329 Frankfurt
Germany

EMail: sanz@denic.de
URI: http://www.denic.de/

Full Copyright Statement

Intellectual Property

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.