ASID Working Group
INTERNET-DRAFT
draft-ietf-asid-rwhois-00.txt
Expires: Jan 30, 1998
Intended category: Standards Track

Network Solutions, Inc.

July 30, 1997

Referral Whois Protocol (RWhois) 2.0

Status of this Memo

Abstract

This memo describes Version 2.0 of the client/server interaction of RWhois. RWhois provides a distributed system for the display of hierarchical and non-hierarchical information. This system is primarily hierarchical by design, allowing for the deterministic reduction of a query and the referral of the user closer to the maintainer of the information. It also identifies the attributes that are non-hierarchical so that they may be indexed into a mesh.

RWhois differentiates itself from other directory service protocols with its ability to reduce the query (narrow in on a hierarchical server that may be able to help) and with its integrated awareness of authority areas.

Table of Contents

* Appendix B: General ABNF definitions

Summary Of Changes Between RWhois Spec 1.5 And 2.0

* Internationalization
    - Character set specification within an object and within a query
    - Encoding specification within an object and within a query
    - Entire protocol _can_ be MIME encoded
* Query Language
    - Per-attribute constraints
    - Boolean operations
    - Nested operations
* Revised Data Model
    - More object oriented
    - Start Of Authority (SOA) and Schema objectified
* Response Formats
* Display Formats
* Generic Query Reduction Rules
    - Standardization of reducible fields
    - Reduction token specified by regular expression match
* Support for Secondarying
    - Complete and partial
* Authority Area Enforcement
* Registration
* Security
    - Enhanced guardian support
    - Per-attribute access control lists

Open Issues

This is an Internet-Draft, which means that more work needs to be done. In the opinion of the original authors, there are several open issues that need to addressed before this protocol can move forward; they include the following.

* Registration
  Registrations need to be examined for the demands they make on implementations with respect to the atomicity of operations. It is the authors intention to not require a system that would need to implement roll backs within its database. It is also desirable for the protocol to be able to support multiple masters so that updates can take place at multiple locations.
* Caching
  Caching should be discussed in relation to schema and how a cache is searched locally given so not to have to implement a server's database in the client.
* Security
  As mentioned in Section 6, security will be reworked.
* Common Indexing Protocol (CIP)
  RWhois 2.0 is intended to work very closely with the CIP. As CIP is standardized, RWhois 2.0 should be modified to operate with it. This

document may need to specify a CIP-index object type.

* Incremental Updates
  To scale in large application domains, a secondary should be able to be updated incrementally. One aspect of this is a 'diff' style update, where only the differences between objects are transported.

1.0 Introduction

Originally, the RWhois protocol [RFC1714] was designed primarily around the data found at the InterNIC and other domain-name and IP registrants such as Internet Service Providers (ISPs). Versions 1.0 [RFC1714] and 1.5 [RFC2167] of the protocol illustrated that the system is useful and works well within certain constraints. During this time, several problems appeared that require an update to the protocol. First, RWhois will be used as a more generic directory services tool in the near future, so the protocol must be able to deal with much more disparately organized data. Some of the data may be in other character sets, its hierarchy may not be '.' delimited, and the users may need a richer query language.

Until Version 1.5 of RWhois, RWhois retained the basic functionality of the whois server and client, making all of the extensions optional. This requirement changed in Version 1.5, however. This version of the RWhois protocol is NOT backward compatible with any of the previous versions. The server MAY respond to a whois or RWhois Version 1.0 or 1.5 client, but it is not required to do so. The RWhois Version 2.0 client MAY also interact with whois [RFC954] servers or RWhois Version 1.0 or 1.5 servers, but it is not required to do so. This is an implementation choice.

RWhois Version 2.0 has been designed as an extensible protocol to ensure that many uses can be accommodated. Public extensions to the protocol sho=
uld
be documented as RFCs. Private extensions can be used with agreement left up to the client and server. Since the protocol is MIME oriented, any extensions should use the x- convention for specifying experimental or non-standard MIME Content-Types. If extensions are not implemented at the server in question, an appropriate error message must be sent.

This document uses words such as SHOULD and SHALL with the special meaning specified in "Key Words For Use In RFCs To Indicate Requirement Levels." [RFC2119]

1.1 Definitions

Class
: A type of object.

Class Definition/Schema
: Identifies all of the attributes allowed within an object of that class.

Attribute
: A variable within a class. (Example: Organization-Name contains the value for the name of an organization.)

Object
: An instance of a class that contains data.

SOA - Start Of Authority
: This object contains the administrative variable of an authority area.

Guardian Object
: This object, when linked to another object, specifies the method of protection for that object.

Authority Area
: An authority area is an identifier for an RWhois database containing data and its schema [1]. It has a hierarchical structure that helps identify the position of the database in the global RWhois data information tree. This version of the protocol specifically allows for hierarchy based on arbitrary and possibly complex criteria. Thus, an authority area can be in any character set as well as a complex set of data. For example, CIDR blocks have fairly complex ideas of how to allocate a given space; this is not easy to specify as a authority area.

Primary RWhois Server
: A primary (or master) RWhois server is where data is registered for an authority area. It answers authoritatively to queries for data in that authority area. There MUST be one and only one primary server for a particular authority area, and an RWhois server may be primary for multiple authority areas.

Secondary RWhois Server
: A secondary (or slave) RWhois server is where data is replicated from a primary server for an authority area. It, like its primary server, answers authoritatively to queries for data in that authority area. There can be multiple secondary servers for a particular authority area. An RWhois server may be secondary for multiple authority areas.

2.0 Connection Model

2.1 Standard Query (TCP)

This document describes how RWhois Version 2.0 would behave using Transmission Control Protocol (TCP).

2.2 Quick Query (UDP)

The overhead incurred by establishing a TCP connection and interacting with an RWhois server may be unnecessary if the client only wishes to ask one question. A separate document will describe the User Datagram Protocol (UDP) facility for RWhois. Adjustments to the query must be made to make this a practical option. The only function allowed while utilizing UDP is a single query.

Unless the client is told that a certain host supports both TCP and UDP the client should attempt UDP first. If the server has not responded after a one-second interval, the client should then attempt to connect via TCP. If the server responds via UDP while the client is connecting/connected via TCP, the client should remember this and set the UDP timeout to the delay

for the returned packet plus one second.

The SRV DNS record provides for a future method of discovering the existence of either TCP or UDP services by querying DNS for tcp.rwhois.<domain> and udp.rwhois.<domain>. RFC2052 states that if a service is not available the target of the SRV record should be "." (NULL in DNS parlance). Thus, an RWhois client can discover which services are available.

3.0 Client/Server Interaction Model

For RWhois Version 2.0 to support multiple character sets and encodings requires that some method be used to identify those encodings and character sets. Instead of developing yet another encoding scheme, an off-the-shelf system can be used. MIME is currently being used by several protocols as a method for encoding objects: therefore, the model for RWhois Version 2.0 will be encoded as a sequence of directives and responses encoded as MIME objects. Most of the client/server interaction concerns three types of objects: directives, responses, and results. Section 2.2 outlines these objects and their relationship to each other. The protocol is the specification of the interaction between the client and the server with respect to when and how these objects are exchanged.

However, the protocol may seem 'ugly' or too 'large' on the wire since every directive is encoded in its own set of MIME headers. Also, it is rare that one element of a client/server interaction would be in a different character set/encoding than the previous element. Both considerations are specifically addressed by two design decisions.

First, while internationalization is a goal, in the near future a majority of users will still be using the English language represented with US-ASCII. To optimize for this and to allow some degree of backward interoperability, the default character set and encoding will be US-ASCII. The default language will be English.

Second, the client can specify its own default encoding, character set, and language in its response to the server's banner. This is specified in Section 3.1.2 as part of the response to the server's banner. For example, when a client contacts a server that supports multiple character sets, the server uses US-ASCII by default. When the client connects, it specifies that it wants to talk Unicode using a UTF-8 encoding for queries in Inuit, the language used by the native people of northern Canada. From that point, the default encoding is as specified by the client.

There are four distinct processes involved in the interaction between a client and a server.

1. Initial connection and subsequent protocol and capability negotiation. This is accomplished with the server sending the Banner and the client sending either a directive or a client capability object. When the client simply sends a directive and not a client capability object, the client implicitly states that it has accepted the servers defaults. See

Section 2.1 Banner for a detailed description of this process.
2. Exchange of zero or more MIME objects containing directives and responses. These are used to set up the environment and specify the query. These objects are listed in Section X.0, Directive and Response Objects.
3. Exchange of zero or more query object. These objects are explained in Section X.0, Query Objects, and in Section X.0, The Search Model.
4. Exchange of zero or more Result objects. Result objects are explained in section X.0 Result Objects.
5. Final disconnect. The final disconnect sequence is explained in Section X.0. While the client SHOULD issue a disconnect sequence, server developers should handle the very common occurrence of clients simply disappearing. It should also be noted that the client can specify a disconnect sequence at any time; it does not need to follow the above sequence before it can disconnect.

3.1 Banner

Even though this version of the RWhois protocol does not require backward compatibility with both whois and whois++, there is some desire to allow older clients to realize that the server they have connected to is of a later revision. It is also common practice for most Internet protocols to greet the client with some sort of banner that fits on one line. This banner must be followed by either a directive from the client or a client capability object.

While it is not a requirement that server developers support all past versions of the RWhois protocol, developers should realize that old clients may (and will) connect to new servers. When a client responds in such a way that illustrates that it does not understand the banner (i.e., it picked a protocol version that is not supported by the server) then the server should respond with an error message the client can understand. For older clients (versions 1.0 and 1.5), the server should send back the appropriate "%error" message as specified in RFC 1714.

Defaults for the MIME attributes and headers are as follows.

MIME header/attribute	Value
Content-Transfer-Encoding	8bit
Content-Language	en-US
charset parameter	US-ASCII

The server MUST assume these values if the client does not specify otherwise. All clients must assume that all RWhois Version 2.0 servers assume these defaults. It should be considered an error for a client to try to interact with a server in anything other than these defaults without first specifying its desire to do so.

3.1.1 From the Server

rwhois-banner =3D "%rwhois" space version-list space host-name=20

```
   [space implementation] crlf
version-list =3D version *("," version)
version =3D version-number [":" capability-id] / "V-2.0" ":" capability-id
version-number =3D "V-" 1*digit "." 1*digit
capability-id =3D response-id ":" extra-id
response-id =3D 6hex-digit
extra-id =3D 2hex-digit
implementation =3D 1*any-char
```

Version-list indicates all of the protocol versions that this server supports. The versions should be listed from oldest to newest so clients do not need to understand potential future changes to the format of this list. The client=92s behavior beyond this point depends on which version of the protocol the client chooses. This document is only concerned with the ensuing dialogue if the version number is 2.0 or lower.

Capability-id identifies the server's capabilities. Each optional directive has a bit that is set if the server can understand the directive. Clients should use this information to determine which of the optional capabilities can be used. The value information for each directive can be found in the definition for that directive. All of the capability ids are listed in Appendix B.

Host-name is the host name of the computer hosting the RWhois server.

Implementation contains information about the server implementation. It is recommended that the version number of the software be indicated. This version may differ from the specification version number.

Example of Use:

```
%rwhois V-1.0:04ffff:0f:0f,V-2.0:04ffff:0f root.rwhois.net (NSI V-1.6)
```

3.1.2 From the Client

The client=92s response to this banner depends upon which version of the supported protocol the client picks. As stated earlier, this document is only concerned with the response for Version 2.0 and lower. If the client wishes to respond as a Version 1.0 client, then it will issue the appropriate -RWhois response as specified in RWhois Version 1.0 and 1.5 RFCs. If the client wishes to connect as a Version 2.0 client, it has two options.

The first option is to simply accept the server=92s defaults and to specify the directives it wishes to execute. If the response sent from the client is not a valid MIME object that includes the client capability response, the server should assume that the client has accepted its defaults and that what is being sent is a directive of some type.

The second option is for the client to send a valid "rwhois" directive to the server. See Section 5.X for the definition of the "rwhois" directive. The server MUST either accept the defaults as specified by this directive or

issue the "504 Specified Defaults Unsupported...Goodbye" message.

3.2 Base MIME Objects

There are two design criteria used to decide on the MIME objects. The first is that there be as few MIME objects as possible. This reduces the amount of time spent in the MIME parser and ensures the MIME Content-Type does not bloat. The second criteria generalizes the specific parts of the protocol so that developers can re-use large amounts of code for different directives/responses. Therefore, the following MIME objects will be used for ALL protocol interactions. As soon as this draft has been finalized these Content-Types will be registered with the Internet Address and Naming Authority (IANA).

All MIME objects MUST follow Simple Mail Type Protocol (SMTP)-style body termination. The data contained in the body is sent octet-for-octet, except when the pattern ``<cr><lf>.<cr><lf>''occurs. In that case, the period is repeated, resulting in the following pattern: ``<cr><lf>..<cr><lf>''. When the data is finished, the octet pattern ``<cr><lf>.<cr><lf>'' is transmitted. On the receiver's side, the reverse transformation is applied, and the message read consists of all bytes up to, but not including, the terminating pattern. This object termination is used whether the default MIME encoding is used or not. Thus, the following two representations of a response are valid and MUST be considered equal.

Object given with no header:

```
XXX Message
=2E
```

Both terminator and header specified:

```
Content-Type: application/rwhoisv2-response

XXX Message
=2E
```

3.2.1 Directives

Directives are used to set up the environment that the client needs and to specify the query to execute in that environment. The Content-Type is "application/rwhoisv2-directive". The body of the object contains exactly one directive. An empty body is an error, and more than one directive in a body is also an error.

The body follows a format of the directive name being the first space delimited token on the first line. Arguments on the same line and the format of subsequent lines are defined by the directive. The directive name will be encoded in 7-bit ASCII; each directive definition can specify other values that must be 7-bit ASCII. Arguments that are NOT specified to be 7-bit ASCII follow the character set/language as specified in the MIME header for that set of directives.

For example:

```
Content-Type: application/rwhoisv2-directive;

limit 200
=2E
Content-Type: application/rwhoisv2-directive;
Content-Language: en-UK

query first-name=3D"Winston" and last-name=3D"Churchill"
=2E
```

MIME headers can be defaulted, and the above two directives can also be specified like this:

```
limit 200
=2E
Content-Type: application/rwhoisv2-directive;
Content-Language: en-UK

query first-name=3D"Winston" and last-name=3D"Churchill"
=2E
```

It should also be noted that the limit and language directives also have global constraint counterparts as specified in the query directive. Thus, the previous interaction could be compressed even further to this:

```
query first-name=3D"Winston" and last-
name=3D"Churchill":limit=3D200;language=3Den-UK
=2E
```

Some may say that there are redundant constraints, directives, and MIME attributes. However, these enable one to specify attributes at several levels of granularity within the protocol. While US-ASCII may be the default, some servers may want to specify the French language for certain attributes to allow soundex matches. This flexibility is intentional so that those building fast servers need not be encumbered by a full MIME implementation. Those wanting a truly robust system, therefore, are not crippled by the "need for speed".

3.2.2 Responses

Responses are the return codes and other directive output that is not normally considered to an RWhois data object. The Content-Type is "application/rwhois-response". The body follows the format of the code and its textual representation takes up the first line. Subsequent lines are reserved for each response code to be able to specify its own format.

An application/rwhois-response MUST have one and only one response within the body. Zero responses must be considered an error. More than one response must be considered an error.

Content-Type: application/rwhoisv2-response;

200 Directive ok
=2E

3.2.3 Results

Results are actual data objects returned from a directive. Only certain directives return result sets instead of responses in their normal case. These results can be any MIME-encoded object. The user can request which Content-Type to return by using the 'display' directive.

When there are multiple, separate objects to be returned, the results will be encoded in a stream of objects in a multipart/mixed MIME object. When an object has sub-parts, such as when an image is sent along with the results, the object must be encoded within a MIME multipart/related object.

Additionally, if a client sets the 'display' directive to 'multipart/related' then no matter how many returned objects or their complexity, the server MUST return a 'multipart/related'. In this case, the server should set the root of the multipart/related object to the TOC Object, which is used to point to the subsequent objects. See Section 4.3.6 for a description of the TOC Object.

When the client does not specify a display type or the data itself does not specify its own encoding, the default MIME type used will be 'text/directory'. This means that text/directory will be used as the standard and canonical exchange format for all native RWhois objects. The text/directory MIME type requires the 'profile' parameter, which is the 'scheme' or class of the body. To alleviate name clashes, the profile namespace for all native RWhois objects is required to start with "rwhois- -". The Attribute Class would use the profile "rwhois-attribute", for example.

3.3 Directive and Response Specifications

All directives encoded within the application/rwhois-directive object are optional with the exception of the rwhois directive that is discussed in Section 3.1.2. This is the only required directive, and without it the server would have no idea with what it was communicating. The directives given here are those that are required to fully cooperate within the RWhois Tree. It is up to implementers to choose whether or not to fully cooperate.

Responses that are specific to a given directive are detailed with that corresponding directive. General responses are listed at the end of this section.

It is also important to note that several directives return objects. Since all objects can be queried it follows that any directive that returns an object could have also been specified via a query. For example, the =91attribute=92 directive returns objects found in the Attribute class. This directive can also be specified by querying directly against the Attribute

class. In each instance where this is true a query form and a directive form are given. These two forms MUST be considered identical. Implementers are STRONGLY encouraged to implement the directive by simply translating it into the query or vice versa. This cannot be stressed enough. Any deviation between the two specifications can reek havoc with the protocol.

3.3.1 rwhois

Description

The "rwhois" directive is used by the client to identify itself as an RWhois client, identify the version of the RWhois protocol that it desires to speak, and the defaults that the client wants the server to use. It is the client=92s version of the server's banner.

ABNF

```
rwhois-dir =3D "rwhois" crlf *(rwhois-av) "." crlf
rwhois-av =3D "Protocol-Version:" space version-val crlf
            "Implementation:" space impl-val crlf
            "Default-Content-Encoding:" space encoding-val crlf
            "Default-charset:" space charset-val crlf
            "Default-Content-Language:" space language-val crlf
            attribute-name ":" attribute-value crlf
version-val =3D version-number
version-number =3D "V-" 1*digit "." 1*digit
response-id =3D 6hex-digit
extra-id =3D 2hex-digit
impl-val =3D 1*any-char
encoding-val =3D 1*any-char
charset-val  =3D 1*any-char
language-val =3D 1*any-char
```

The version-val is a required argument that identifies the client's protocol version and capabilities. Each bit represents the optional responses the client can understand. Servers should use this information to determine which of the optional capabilities can be used. The value information for each server response can be found in Section 7.0 of this document. Note that no list of versions is given as in the server=92s banner. This is because the onus is on the client to pick which version of the protocol will be spoken. This is the client's way of saying "I'll take that one."

The impl-val argument is optional and identifies client implementation information. It is recommended that the implementer maintain a version number separate from the specification version.

The encoding-val is a required argument that specifies the default MIME encoding that will be used by the client. The value MUST be the value of the Content-Transfer-Encoding MIME header as specified in [RFC1521].

The charset-val argument is required and specifies the default character set

that will be used by the client. This value MUST be a valid value for the 'charset' attribute of the text MIME Content Type as specified in [RFC1521].

The language-val is a required argument that specifies the default language that will be used by the client. This value MUST be a valid value for the Content-Language header as defined in [RFC822] and extended in [RFC1766].

Errors

338 Invalid directive syntax
300 Not compatible with version
301 Server not capable of using client defaults

Examples

Content-Type: application/rwhois-directive;

rwhois
Protocol-Version: V-2.0
Implementation: InterNIC B.0.9.7
Default-Content-Encoding: 8bit
Default-charset: ISO-8859-1
Default-Content-Language: en-US
=2E
200 Directive ok
=2E

3.3.2 directive

Description

The "directive" directive can be used by the client to get information about the directives that the server supports. The response must contain the name and description of each specified directive and may be expanded in the future with additional attributes. When no directive name is given, the server must return information about all the directives.

As with other directives, the result is an object of the Directive class and, as such, can also be returned by a standard query against that class.

ABNF

directive-dir =3D "directive" *(space directive-name) crlf
directive-query =3D "query Class-Name=3Ddirective" *(space directive-name) crlf
directive-name =3D 1*id-char

Errors

338 Invalid directive syntax
400 Directive not available
401 Not authorized for directive

Examples

Without parameters:

```
directive
=2E
Content-Type: multipart/mixed; boundary=3D"foo"

- --foo
Content-Type: text/directory; profile=3Drwhois-directive

directive:query
description:RWhois query directive. Used to retrieve an object.
- --foo
Content-Type: text/directory; profile=3Drwhois-directive

directive:directive
description:RWhois directive directive
- --foo
Content-Type: text/directory; profile=3Drwhois-directive

directive:limit
description:RWhois limit directive used to limit the number of hits.
- --foo--
=2E
```

With parameters:

```
directive quit
=2E
Content-Type: text/directory; profile=3Drwhois-directive

directive:quit
description:Quit connection
=2E
```

3.3.3 display

Description

By default, the server returns a text/directory representation of an object in a result set. This default format can be changed with the "display" directive. When no parameter is given, the server must list all the display formats it supports.

ABNF

```
display-dir =3D "display" crlf
/ "display" space IMT crlf
display-response =3D *display-record
display-record =3D "name" ":" IMT crlf *display-line crlf crlf
display-line =3D attribute-name ":" attribute-value crlf
```

Errors

338 Invalid directive syntax
400 Directive not available
401 Not authorized for directive
436 Invalid display type

Examples

```
# Get the available display formats.
display
=2E
name:text/directory

name:text/html

name:text/plain
=2E

# Change the active display format.
display text/plain
=2E
200 Directive ok
=2E
```

3.3.4 forward

Description

The "forward" directive instructs the server to follow all referrals and return the results to the client. This directive can be used to run an RWhois server as a proxy server. The default value must be "off". When the value is set to "on", the server must not return referrals.

ABNF

```
forward-dir =3D "forward" space on-off crlf
```

Errors

338 Invalid directive syntax
400 Directive not available
401 Not authorized for directive

Examples

```
forward on
=2E
200 Command ok
=2E
```

```
forward off
=2E
200 Command ok
=2E
```

3.3.5 limit

Description

When returning a query result, the server should limit the number of objects returned to the client. The "limit" directive changes this limit. The default and maximum limit is server-dependent. The client can get the current limit by using the "status" directive (see Section 3.3.13).

ABNF

```
limit-dir =3D "limit" space 1*digit crlf
```

Errors

331 Invalid limit
338 Invalid directive syntax
400 Directive not available
401 Not authorized for directive

Examples

```
limit 100
=2E
200 Command ok
=2E
```

3.3.6 notify

Description

The "notify" directive performs several functions.

* If the server returns a referral that results in an error, the client can report the bad referral to the server using the "badref" option.
* When the client follows referrals and goes through the same referral twice, that referral is a recursive referral and causes a referral loop. The client can report the recursive referral to the server using the "recurref" option.
* When the data in an authority area changes, a master server can use the "update" option to notify its slave servers to update the data.
* The "inssec" option allows an RWhois server to register itself as a slave server for an authority area with a master server. The master server may reject the request on the basis of its registration policy.
* The "delsec" option allows a slave server to cancel its registration with the master server.

ABNF

```
notify-dir =3D "notify" space "badref" space referral-url crlf
        / "notify" space "recurref" space referral-url crlf
        / "notify" space "update" space host-port ":" authority-area crlf
        / "notify" space "inssec" space host-port ":" authority-area crlf
        / "notify" space "delsec" space host-port ":" authority-area crlf
referral-url =3D rwhois-url / uri
```

Errors

338 Invalid directive syntax
340 Invalid authority area
342 Invalid host/port
400 Directive not available
401 Not authorized for directive

Examples

The client reports a bad referral to rwhois.foobar.com to the server:

```
notify badref rwhois://rwhois.foobar.com:4321/class=3Ddomain%20 \
    auth-area=3Dfoobar.com
=2E
200 Directive ok
=2E
```

The client reports a recursive referral to rwhois.foobar.com to the server:

```
notify recurref rwhois://rwhois.foobar.com:4321/class=3Dcontact%20
   auth-area=3Dfoobar.com%20Last-Name=3D"Beeblebrox"
=2E
200 Directive ok
=2E
```

The master server for the foobar.com authority area notifies its slave servers to update the data:

```
notify update master.foobar.com:4321:foobar.com
=2E
200 Command ok
=2E
```

The server rwhois2.foobar.com registers as a slave server for the foobar.com authority area:

```
notify inssec rwhois2.foobar.com:4321:foobar.com
=2E
200 Command ok
=2E
```

The server rwhois2.foobar.com cancels its registration as a slave server for

the foobar.com authority area:

notify delsec rwhois2.foobar.com:4321:foobar.com
=2E
200 Command ok
=2E

3.3.7 quit

Description

The "quit" directive can be used by the client to close the connection. Before the server closes the connection, it must respond with the "203 Goodbye" response.

ABNF

quit-dir =3D "quit" crlf "." crlf
quit-response =3D response

Errors

No errors.

Examples

quit
=2E
203 Goodbye
=2E

3.3.8 register

Description

The "register" directive can be used by the client to add, modify, or delete objects in the server's database.

The "register" directive is slightly different than most directives in that, instead of simply receiving one application/rwhois-directive object, the server receives a multipart/related body whose 'start' parameter points to the application/rwhois-directive object that contains the "register" directive. The "register" directive can contain any number of operations and any combination of those operations. These operations must be considered atomic with respect to the other operations within the directive. For example, a "register" directive contains three adds and three mods. If any one of these operations fails, they ALL fail.

An operation within a register directive can also contain multiple objects on which the operation must be done. These must also be considered atomic; if the operation fails for any object, the whole operation fails (and thus the entire "register" directive fails).

The operations available within a register directive are:

* The "add" option indicates that the object being sent should be added to the server's database. The objects are identified by their content-ids.
* The "mod" option indicates that the object being sent is a modification of an object that already resides on the server's database. These database objects are identified by their IDs and updated times. The objects to replace them are identified by their content-ids.
* The "del" option indicates that the objects being sent should be deleted from the server's database. The database objects are identified by their ID and updated times.

After a register operation (add, modify, or delete an object) in an authority area, the server should update the "Serial-Number" variable in the SOA information for the authority area. This is useful for data replication, because a slave server checks the "Serial-Number" variable to detect a data change at the master server (see Section 3.6.2).

ABNF

```
register-dir =3D "register" crlf *(register-line crlf)
register-line =3D  "add:" register-add-tuple *( register-add-tuple )
        / "mod:" space register-mod-tuple *( space register-mod-tuple )
        / "del:" space register-del-tuple *( register-del-tuple )
register-add-tuple =3D space cid
register-del-tuple =3D rwhois-id "," rwhois-updated
register-mod-tuple =3D rwhois-id "," rwhois-updated "," cid

register-updated =3D time-stamp
register-response =3D register-response-header crlf  *(register-operation- -
result)
register-response-header =3D "241" space register-response-text
register-response-text =3D 1*any-char
register-operation-result =3D "object:" space cid rwhois-id rwhois-updated crlf
```

* For the "add" operation, the client must send all required attributes for the object, including the Class-Name and Auth-Area attributes. However, the client must not send the ID and Updated attributes. These attributes are assigned by the server and returned in the response.
* The rwhois-response contains references only to those objects who were added. The cid is used to identify the object to the client. The ID and rwhois-updated fields are those that were assigned by the server at the moment the object was inserted into the database.

Errors

120 Registration deferred
241 Register complete
320 Invalid attribute
321 Invalid attribute syntax

322 Required attribute missing
323 Object reference not found
324 Primary key not unique
325 Failed to update outdated object
336 Object not found
338 Invalid directive syntax
340 Invalid authority area
341 Invalid class
400 Directive not available
401 Not authorized for directive

Examples

```
# Add an object.
Content-Type: multipart/related; type=3Dapplication/rwhois-directive;=20
 start=3Dcid1@foo.com; boundary=3Dexample-1

- --example-1
Content-Type: application/rwhois-directive
Content-ID: <cid1@foo.com

register
Add: <cid2@foo.com>
Mod: 23456789.a.com,19961205124403000,<cid3@foo.com>
Del: 23456789.a.com,19961205224403000
- --example-1
Content-Type: text/directory
Content-ID: <cid2@foo.co>

Class-Name:contact
Auth-Area:a.com
First-Name:Scott
Last-Name:Williamson
Name:Williamson, Scott
Email:scottw@a.com
- --example-1
Content-Type: text/directory
Content-ID: <cid3@foo.com>

Class-Name:contact
Auth-Area:a.com
ID:23456789.a.com
First-Name:Scott
Last-Name:Williamson
Name:Williamson, Scott
Email:sw@a.com
- --example-1--
=2E
```

241 Register complete

```
Object: <cid2@foo.com> 23456789.a.com 19961205224403000
Object: <cid3@foo.com> 56789.a.com 19961205224404444
```

=2E

3.3.9 class

Description

The "class" directive is used by the client to get the meta-information for one or more classes in an authority area. The response is an object of the type Class. This directive and the Attribute directive work together to specify other classes.

The version number of each specified class may be expanded in the future with additional attributes. When no class name is given, the server must return the meta-information for all of the classes in the authority area.

As with the Attribute directive, this directive can also be handled via a query of the Class class. Thus, the following two ABNF grammars must be considered equal:

ABNF

```
class-dir =3D "class" space authority-area *(space class-name) crlf
class-query =3D "query Class-Name=3DClass auth-area=3D" authority-area
*(space "Name=3D" class-name space "or") crlf
```

See Section 3.3 for information on directive and query equality. The response to this directive/query is either an error or a result set of CLASS objects. See Section 4.2.2 for the definition of a CLASS object.

Errors

338 Invalid directive syntax
340 Invalid authority area
341 Invalid class
400 Directive not available
401 Not authorized for directive

Examples

```
class rwhois.net domain host
=2E
```

or

```
query Class-Name=3DClass and auth-area=3Drwhois.net and Name=3Ddomain or
Name=3Dhost
=2E
```

3.3.10 attribute

Description

The "attribute" directive can be used by the client to get attribute definitions. Each definition is complete by itself. It is up to the Class object to aggregate Attribute objects together to define a new Class. As with the "class" directive, this directive can be handled via a standard query of the Attribute class.

ABNF

```
attribute-dir =3D "attribute" space authority-area
       *(space class-name:attribute-name) crlf
attribute-query =3D "query Class-Name=3Dattribute"
                  (" and auth-area=3D" authority-area)
*(space "and Name=3D" attribute-name) crlf
```

Errors

338 Invalid directive syntax
340 Invalid authority area
341 Invalid class
400 Directive not available
401 Not authorized for directive

Examples

attribute map

```
=2E
Content-Type: multipart/mixed; boundary=3Dexample1

- --example1
Content-Type: text/directory

attribute:Class-Name
description:Type of the object
type:TEXT
format:re:[a-zA-Z0-9-]+
indexed:OFF
required:ON
multi-line:OFF
repeatable:OFF
primary:OFF
hierarchical:OFF
private:OFF
- --example1
Content-Type: text/directory

attribute:ID
description:Globally unique object identifier
type:TEXT
format:re:[0-9]+\.[a-zA-Z0-9\.-]+
indexed:ON
required:ON
multi-line:OFF
```

```
repeatable:OFF
primary:ON
hierarchical:OFF
private:OFF
# This is an abbreviated example, more attributes usually follow.
- --example1--
=2E
```

3.3.11 security

Description

The "security" directive enables either a client request or a server response to be authenticated and/or encrypted. This directive can be used to securely access or update any information (meta or data) in an authority area that is protected by one or more guardian objects.

This is an area that needs further work. The examples below that use encrypted passwords and Pretty Good Privacy (PGP) are for illustrative purposes only.

ABNF

```
security-dir =3D "security" space "on" space direction space security-method=20
  [space security-data] crlf security-payload ["-security" space "off" crlf]
direction =3D "request" / "response"
security-method =3D "password" / "pgp" / 1*id-char
security-data =3D password-data / pgp-data / 1*any-char
password-data =3D 1*any-char
pgp-data =3D "signed" / "encrypt" [space key-id] / "signed-encrypt" [space key-
id]
security-payload =3D *(*any-char crlf)
security-response =3D "251" / "252" security-response-text crlf "." crlf
security-response-text =3D 1*any-char
```

* The "password" security-method is available in the "request" direction only. For password, the security-data is a cleartext password.
* The "pgp" security-method is available in both the "request" and "response" directions. For PGP, the security-data indicates how to treat the security-payload: signed, encrypted, or signed and encrypted. To encrypt the security-payload in the "response" direction, the security-data must include the public key ID with which to encrypt it.

Errors

251 Security mode on
252 Security mode off
338 Invalid directive syntax
352 Invalid security method
353 Authentication failed
354 Encryption failed
400 Directive not available

401 Not authorized for directive

Examples

```
# Authenticate a request using password.
security on request password hello!1
=2E
251 Security mode on
=2E

# Authenticate a PGP signed request.
security on request pgp signed
=2E
251 Security mode on
=2E
Content-Type: multipart/related; type=3Dapplication/rwhois-directive;=20
  start=3D<cid1@foo.com; bounday=3Dexample1

- --example1
Content-Type: application/rwhois-directive
Content-ID: <cid1@foo.com

register
Add: <cid2@foo.com
- --example1
Content-Type: application/rwhois-directive
Content-ID: <cid2@foo.com

- -----BEGIN PGP MESSAGE-----
Version: 2.6.2

owHrZJjKzMpgdP9D9crUhdpBYnwHGRnPbmVhmHlV7Hef9je/n7vyzhmE6589/+Dg
jPpVm59tNz92vPSmrFB/4ankBRz+xgY+7z9OUYjefGahbWSNwzzxbw6TpWZGerU+
uOUg/Cygs33JBdHqjwEc+wyfZPp+N5p2bu+ywoaOu8eLPyn+m2Mt/T9p1UaG68vP
Zd2d9EPw+Ywpio7dco6yh3b/v7zmQxJHcWpyaVFmSSUDEHi6WBkZm5iamVtY6iXq
JefnKnCFFqQklqSmWBlaWpoZGhmYGhqZmBgYGxgYKHA55yQWF+v6JeamWiXn55Uk
JpcocDmWlmToOhalJlpB9cf7uYbHE6kWi/VumUXFJRB9wcn5JUBdPokwgfDMnJzM
xNzi/DwFLjQBHQWoatfcxMwcq+JyB6h5AA=3D=3D
=3Da0sQ
- -----END PGP MESSAGE-----
- --example1--
=2E
241 Register complete
Object: <cid2@foo.com 23456789.a.com 19961205224403000
=2E
security off
=2E
252 Security mode off
=2E

# Encrypt a response using PGP. 52160EC1 is the public key ID with which
```

```
# the response is encrypted.
security on response pgp encrypt 52160EC1
=2E
```
251 Security mode on
```
=2E
xfer com class=3Ddomain attribute=3DDomain-Name attribute=3DOrganization-Name
=2E
Content-Type: application/pgp-signed-XXX

- -----BEGIN PGP MESSAGE-----
Version: 2.6.2

hIwDqWWhK1IWDsEBBACOXssTzD2CbB7Vjj2cNURScpJc2as2TbUDOQiwkT+8qFgG
ZyRfktpwNNTawRIcGOk1Kcs84z8a3vvTA/oje9vZexHtzfJwBHFdiIZxPuCEpvgv
2ppK7WqlmHGcQKVBJJHYw7Fq83CUkeGJB9P1M3CQiXeW8h8MwAuhxSgbgt23PKYA
AABuhknJrXeh9Owm81+MvyzgLOyM7sjDYmttU9sj/yuOYmAhS9V+34MT/Mwn4wO8
2BCsJqBHXbwOuYKs02p0se4jyKFtZR8MDPWNm9QyAP+oNMTjsufy6ZRa9PegUC6t
HDhXymkiP03mKMMVK1//7X0=3D
=3DvZ2x
- -----END PGP MESSAGE-----
=2E
security off
=2E
```
252 Security mode off
```
=2E
```

3.3.12 soa

Description

The "soa" directive can be used by the client to retrieve the SOA information for one or more authority areas. When no authority area name is given, the server must return the SOA information for all the authority areas.

As with the Class and Attribute directives, this directive can also be handled via a query of the SOA class. Thus, the following two ABNF grammars must be considered equal:

ABNF

```
soa-dir =3D "soa" *(space authority-area) crlf
soa-query =3D "query" space "auth-area=3D" authority-area crlf "." crlf
```

The server must return the following SOA information for an authority are=
a.
This information is also included in Section 4.3.2, which describes the SOA class.

attribute-name	attribute-value	Comments
authority	authority-area	This is the name of the

		authority area.
ttl	1*digit	This is the default time-to-live for the data in the authority area.
serial	time-stamp	This is the serial number of the data in the authority area; it changes when the data changes.
refresh	1*digit	This is the time interval before a slave server checks for complete replication.
increment	1*digit	This is the time interval before a slave server checks for incremental replication.
retry	1*digit	This is the time interval before a slave server tries again to connect to a master server that appears to be out-of-service.
tech-contact	email	This is the contact for the operation of the master server.
admin-contact	email	This is the contact for the data integrity at the master server.
hostmaster	email	This is the contact for sending update requests at the master server.
primary	host-port	This is the host name (or IP address) and port number of the master server.

Errors

338 Invalid directive syntax
340 Invalid authority area
400 Directive not available
401 Not authorized for directive

Examples

```
soa org
=2E
Content-Type: text/directory; profile=3D"rwhois-soa"

authority:org
ttl:86400
serial:19961119111535000
refresh:3600
increment:1800
retry:180
tech-contact:tech@internic.net
admin-contact:admin@internic.net
```

```
hostmaster:hostmaster@internic.net
primary:rs.internic.net:4321
=2E
```

3.3.13 status

Description

The "status" directive can be used by the client to retrieve an object belonging to the Status class, which is used to contain various status flags for the server. The result must include the number of objects in all of the authority areas, the current display format, the server contact information, and the status flags for the state-oriented directives: "limit" and "forward".

Since the directive returns a result, it can also be handled by a query against the Status class. Since the client has no way of knowing the values in advance, the only thing the client can ask for is the Class itself.

ABNF

```
status-dir =3D "status" crlf
status-query =3D "query Class-Name=3Dstatus"
```

Errors

338 Invalid directive syntax
400 Directive not available
401 Not authorized for directive

Examples

```
status
=2E
Content-Type: text/directory; profile=3Drwhois-status
limit: 20
forward: OFF
objects: 12345
display: text/directory
contact: joe@rwhois.net
=2E
```

3.3.14 xfer

Description

The "xfer" directive can be used by the client (generally, a slave server) to transfer the data in an authority area. The client can control the amount of data transferred using one of the following options. NOTE: The "xfer" directive is identical to the "query" directive. It is included for backward interoperability.

Updating secondary servers is a work in progress. Specifically, some method is needed for sending information on what was deleted. Some of these same issues arose during discussions of incremental updates in CIP and the same problems developed.

In RWhois 1.5, the "xfer" directive had several options. Below are descriptions of how these options should be handled:

* serial-number: In 1.5, the client can transfer all the objects that have been added, modified, or deleted since a certain time, specifying the serial-number that indicates that time. This option was used for incremental replication. In 2.0, this function is handled by querying for objects that have an updated time that is after the time specified in the serial-number.
* class: In 1.5, the client could limit the data transfer to one or more classes, using the "class=3D<class-name>" option. This is handled simply by specifying the class or classes to which the query would apply.
* attribute: In 1.5, the client could limit the data transfer to one or more attributes of a class. In 2.0, this is handled by the INCLUDE global constraint.

Implementers are encouraged to implement security surrounding queries that exhibit security problems. The XFER behavior is one of these.

3.3.15 X-

Description

The "X-" directive is used to specify an additional, non-standard directive. It can be implemented by executing an external program, by internal functions, or by other means. It may interact with the client or simply produce output like one of the standard directives.

ABNF

```
x-dir =3D "X-" x-directive [space x-arguments] crlf *x-line "." crlf
x-directive =3D 1*id-char
x-arguments =3D *any-char
x-response =3D "X6X" x-response-text *(*any-char crlf) crlf "." crlf
x-response-text =3D 1*any-char
x-line =3D *any-char crlf
```

Errors

338 Invalid directive syntax
400 Directive not available
401 Not authorized for directive

Examples

The following example uses an implementation that executes an external program, the UNIX "date" command. The server runs the "date" command and

returns its output to the client.

X-date
=2E
261 X-date output
Mon Jan 6 13:21:20 EST 1997
=2E

4.0 Data Model

The RWhois protocol is primarily concerned with directory-services-type information. Historically, this has been formatted in a series of colon-delimited attribute/value pairs. Thus, the RWhois protocol's primary interaction will be via encoded A/V pairs. In addition to these types, it is also apparent that RWhois will need to deal with objects of different types such as the image of a particular user, a company=92s logo, or an encoded representation of a sales pitch. It is also recognized that in the future other directory services encoding formats that have a richer markup language may become popular. Astute readers may pay attention to the interaction between SGML and MIME for future application to the world of directory services.

The A/V type objects have a specific base class from which all objects are derived. In addition to the base class, there is the schema object, which is used to define other objects that inherit from the base class.

4.1 Base Class

While RWhois does not have or advocate using a specific, standardized schema, it does impose a few requirements. It requires that all defined classes inherit attributes from a particular base class (or base schema). The RWhois specification does not require the actual implementation of inheritance. Instead, all classes must include the attributes defined in the base class.

The base class has the following attributes.

Class-Name	This attribute contains the name of the class to which the object belongs. It is the type of the object itself. It is of type TEXT and is required.
Auth-Area	This attribute contains the name of the authority area to which the object belongs. It, along with Class-Name, definitively defines the type of the object. It is of type TEXT and is required.
ID	This attribute is a universal identifier for the object. It is formed by choosing a string that is unique within an authority area and appending the authority area to it, separating the local string from the authority area name with a period. The only restrictions on the local string are that it must be unique within the authority area and must not contain the period character. This attribute is

	hierarchical in nature. It is always generated by the server (for example, during a register operation). It is of type TEXT and is required.
Updated	This attribute is a time/date stamp that indicates the time of last modification of the object. It is both informational and a form of record locking. It prevents two clients from modifying the same object at the same time. It is of type TEXT and is required.
Guardian	This attribute is a link to a guardian object (described below). Its value is the ID of a guardian object. It is of type ID and is optional. It is repeatable, since an object may have multiple guardians.
Private	This attribute is a true or false flag that indicates whether or not an object is private (that is, not publicly viewable). It defaults to false. If it is true, only the clients that satisfy the authentication/encryption requirements of one of the object's guardians are able to view the object. If the object is publicly viewable, then the Private attribute property of each of its attributes still applies.
TTL	This attribute is the "time-to-live " of a given object. It is included only if an object has a different time-to-live than the default given in the Start of Authority information. Its value is specified in seconds. It is of type TEXT and is optional.

4.2 Defining a New Class

One of the more useful but underutilized aspects of X.500 is the encoding of a particular attributes=92 definition within the actual protocol. The problem was that its existence was not discoverable.

In this version of RWhois, this feature is specifically made to be part of the protocol. Thus, two objects are specified: "Attribute" and "Class". "Attribute" is used to define objects that describe the characteristics of an attribute. The "Class" class is used to define a class of objects by specifying the "Attribute" objects that make up the class.

4.2.1 Attribute Class

Attributes have a number of properties, some mandated by the RWhois protocol and some that are implementation dependent. These properties are usually a reflection of the database system used by the server. The following is a list of the protocol-mandated properties and their descriptions.

Attribute	This is the name of the attribute.
Description	This is a natural-language description of the attribute.
	This is a parameter that broadly indicates the use of the attribute to the protocol. There are three standard types: TEXT, ID, and SEE-ALSO. The default is TEXT, which indicates that the value is a text string. ID indicates that the

Type	attribute contains the ID of another RWhois object; this type of attribute is used for database normalization. SEE-ALSO indicates that the attribute contains a pointer (a Uniform Resource Identifier (URI)) to some other kind of external data; for example, a World Wide Web page or FTP site.
Format	This is an interpretable string that describes the acceptance format of the value. The server (and optionally the client) should match the value to the format string to determine if the value is acceptable. The format of this property is a keyword indicating the syntax of the format string, followed by a colon, followed by the format string itself. Currently, the only keyword recognized is "re" for POSIX.2 extended regular expressions.
Indexed	This is a true or false flag indicating that this attribute should be indexed (and therefore able to be searched).
Required	This is a true or false flag indicating that this attribute must have a value in an instance of the class.
Repeatable	This is a true or false flag indicating that there may be multiple instances of this attribute in a class and each instance is to be interpreted as a separate instance (in contrast to Multi-Line). This flag is mutually exclusive with Multi-Line: if Multi-Line is true, then Repeatable must be false and vice versa.
Primary	This is a true or false flag that indicates that this attribute is a primary key. If more than one attribute in a class is marked as primary, then these attributes together form a single primary key. The primary key is intended to be used to force uniqueness among class instances. Therefore, there can be only one instance of a primary key in a database. The Primary flag implies that the attribute is also required.
Hierarchical	This is either a regular expression or a URI. If it is a regular expression, any value should be matched with the regexp to find the character(s) that form the separator between each sub-part. If it is a URI, the URI is used to uniquely identify a well known rule set that is used to define how hierarchy is expressed in a given value.
Private	This is a true or false flag that indicates whether or not this attribute is private (that is, not publicly viewable). It defaults to false. If it is true, only the clients that satisfy the authentication/encryption requirements of a guardian (described below) are able to view the attribute-value pair.

4.2.2 Class Class

A class is not as difficult to specify as an attribute, since the class is defined by its attributes. Thus, the only required attribute is the ID of the attribute objects contained within the class. Additionally, a

description of the class should be included for human discovery and searching. The following attributes make up the "Class" class.

Description	This is the free-text description of the class that is intended for the consumption of the user.
Attribute-Name	This is the link-style attribute that points to the Attribute object(s) that make up this class by using the ID of the object. This attribute is repeatable.
Constructor	This is a URI that identifies an object that acts as the Constructor for objects that belong to the defined class.

The constructor is used to create new objects since all of the semantics associated with creating an object would be nearly impossible to express in the Attribute definition. For example, an Attribute object might define a given attribute to be an IP address. A client would know what to do with an IP address. But the semantics for how to assign an IP address are simply too complicated to express without some programming language. This is an area for additional standardization.

4.3 RWhois Standard Schemas

The following RWhois standard schema definitions are required for any RWhois installation that wishes to participate in the RWhois tree. The RWhois tree is a mapping of referrals from one RWhois server to another.

Every schema defined below inherits the base result class defined above. The attributes defined in Base Schema as REQUIRED will still be REQUIRED in RWhois standard schemas, while the OPTIONAL attributes will still be the OPTIONAL.

4.3.1 Referral Class

The referral class is defined to hold referral information (typically for link referrals). It consists of attributes defined as part of the base class, the protocol-specific attributes described below, and any installation-specific attributes.

Referral	This attribute contains the referral itself; it is an RWhois URL. It is of type TEXT and is required. It is repeatable, since more than one server can host a Referred-Auth-Area. In RWhois 1.5, the referred authority area was in a separate field. In this version, the authority area MUST be included within the URL contained in this field.

4.3.2 SOA Class

Each authority area has some administrative variables, defined at the master server, to control data replication. These variables are called the SOA variables. They are listed below.

Serial-Number	This is the serial number of the data in an authority area. The master server should update this variable whenever the data in the authority area is changed. Its value is a time/date stamp.
Refresh-Interval	This is the time interval before a slave server checks for complete replication. Its value is specified in seconds.
Increment-Interval	This is the time interval before a slave server checks for incremental replication. Its value is specified in seconds.
Retry-Interval	This is the time interval before a slave server tries again to connect to a master server that appears to be out-of-service. Its value is specified in seconds.
Time-To-Live	This is the default time-to-live for the data in an authority area at a slave server. The slave server should not answer authoritatively to queries for such stale data. Its value is specified in seconds.
Time-To-Die	This is the minimum time interval for which a server will keep information about the deletion of a given object. If a slave server asks for an update after this time has expired, it may not receive information that a given object was deleted.
Admin-Contact	This is the email address of an individual or a role account responsible for the data integrity in an authority area at the master server.
Tech-Contact	This is the email address of an individual or a role account responsible for the operation of the master server for an authority area.
Hostmaster	This is the email address of an individual or a role account to whom email messages to update the data in an authority area at the master server are sent.
Primary-Server	This is the location of the master server for an authority area. Its value must contain both the host name (or IP address) and port number of the master server.

4.3.3 Guardian Class

The guardian class is defined to hold security information. The fundamental concept behind the guardian class is that an object (or another structure) is "guarded" by containing a pointer to a guardian object [Guardian]. To modify, delete, or possibly view the guarded object, the authentication (or encryption, or both) scheme must be satisfied. Guardians are intended to not have rank: if an object is guarded by more than one guardian object, satisfying any one of those guardians is sufficient. A guardian object that does not have any Guardian attribute linking it to other guardians guards itself. That is, the authentication scheme in the guardian object itself must be satisfied to modify, delete, or possibly view it.

Guardian objects are typically linked to actual database objects with the Guardian attribute found in the base class. However, a guardian may also be

linked to an entire authority area, in which case the guardian becomes implicitly linked to all of the objects contained within the authority area.

The guardian class consists of the base class, the protocol-specific attributes described below, and any installation-specific attributes.

Guard-Scheme	This attribute contains a keyword indicating the authentication methodology. Its value must be understood by both the client and server, and its value dictates the contents of the Guard-Info attribute. It is of type TEXT and is required.
Guard-Info	This attribute contains that data that is used by the Guard-Scheme to verify the authentication. Its actual format is dictated by the Guard-Scheme, for example, it could contain a password or PGP public key id [RFC 1991]. For security reasons, it should not be displayed, and its Private attribute property should be set to true. It is of type TEXT and is required.

4.3.4 Status Class

This class is used to define objects that are used to store the current status of the server. As with any object text/directory object, locally defined attributes can be added to the object that are not defined by that object=92s class definition. This class will be subject to such treatment, as certain server implementations will want to return additional implementation specific information.

It is important to note that querying this class is usually meaningless, since the client should not know the values in advance. Objects of this type are usually created as they are returned. See the 'status' directive for how to ask for an object of this class.

Limit	This is the current hit limit.
Forward	This is the status of the Forward flag.
Objects	This is the current number of objects stored by the server.
Display	This is the current display type.
Contact	This is the email address of the server maintainer.

4.3.5 Directive Class

This class is used to define objects that describe the directives currently supported by the server in question. See the 'directive' directive for more information on this class and how to query it.

Directive-Name	This is the name of the directive as it should be entered to invoke it.
Description	A natural-language description of what the directive does, that is intended for human consumption.

4.3.6 TOC Class

This class is used to =91point=92 to other objects in a multipart/related result. It is not normally queried and doing so may simply not make sense. There is only one attribute/value pair which will usually be repeated several times.

Object The value for this attribute contains information about each object contained within the multipart/related object.

The format of the value follows the following ABNF:

value =3D cid [directory-profile] [rwhois-id] [user-string]

directory-profile =3D 1*any-char

user-string =3D 1*any-char

directory-profile is the value of the profile attribute of the application/directory object who=92s content-id is contained in cid.

user-string is descriptive text formed from the object itself and is meant as a summary of the object so that the TOC object can be displayed somewhat like a menu. How this descriptive string is formed is completely implementation specific.

5.0 Search Model

The search model is made up of four distinct aspects. There is the query itself, which is specified by the client. The server then takes that query and looks in its local database. If it finds an appropriate object, it returns the results. If it cannot find an appropriate object, the server uses query reduction to find a referral to a server that may know more about the object in question. Returned objects can also have a see-also, which provides a pointer to another object that contains additional information that may be of use to the client.

5.1 Query

The query is generally the final directive sent to the server. The query is the question that the client wants the server to answer. It has two primary parts: the query terms themselves and global constraints that apply to the entire query and the display of its output. There is a similarity between this query syntax and the whois++ query syntax, but there are differences.

Format for Use:

query <query terms>:<global constraints>;<global constraints>;...

A query term is defined as one or more terms followed by an optional semicolon and one or more local constraints. Local constraints are bound only to the term they follow. All local constraints can be global

constraints, but only some global constraints make sense as local constraints.

Terms can be separated by the logical operators AND, OR, or NOT. The lack of an operator between two terms MUST be interpreted as an implicit AND. A term itself can have several formats.

1. A value string by itself is considered a search on all attributes.
2. A term followed by a semicolon and a semicolon-delimited list of local constraints.
3. An attribute specifier or an abbreviated attribute specifier followed by an "=3D" and a single value string followed by an optional semicolon and a semicolon delimited list of local constraints.

Special characters that need to be quoted are preceded by a backslash (\). Special characters are space (' '), tab, equal sign (=3D), comma (,), colon (:), backslash (\), semicolon (;), asterisk (*), period (.), parenthesis [()], square brackets ([]), dollar sign ($), and circumflex (^).

NOTE: Unlike the whois++ syntax, values may be enclosed within double quotes. In this case the only values that need to be escaped are '"'.

Example query:

```
query Name=3D"Scott Williamson";Language=3Den;Charset=3DISO-8859-1 AND \
 Country=3Dus:output=3DMIME
```

NOTE: The query should be all on one line. In the above example, the query is continued over two lines for readability only.

5.2 Constraints

Local constraints are generally used to override some global constraint or default. Any local constraint can be a global constraint.

There are two groups of Global constraints: those that make sense only in the query and those that are also directives in their own right. For example, specifying a global query match type of regular expression makes no sense for other types of client/server interaction. Thus, it only appears in the query. Setting a language and/or character set, on the other hand, has value for other functions such as info directives and schema definitions.

Experimental constraints can be constructed by using the "X-" convention used by many protocols.

Constraint	Allowed Values	Local or Global	Equivalent Directive
SEARCH	{ exact-string | regex | fuzzy | substring | }	LOCAL/ GLOBAL	search
LIMIT	{ 1-<max-allowed> }	GLOBAL	limit

CASE	{ ignore | consider }	LOCAL/ GLOBAL	case
INCHARSET	{ us-ascii | iso-8859-* }	LOCAL/ GLOBAL	rwhois
LANGUAGE	<As defined in ISO 639:1988>	LOCAL/ GLOBAL	language
IGNORE	{attributelist}	GLOBAL	ignore
INCLUDE	{attributelist}	GLOBAL	include
CLASS	{NAME of a valid object type}	LOCAL/ GLOBAL	
AUTH_AREA	{NAME of a valid authority area}	LOCAL/ GLOBAL	

NOTE: IGNORE is used to specify attributes that should specifically be ignored when searching and displaying a given class of objects. Conversely, INCLUDE is used to specifically ignore all cases except those specified in the INCLUDE's values.

5.3 Query Reduction

The critical component of the RWhois server is the ability to reduce the query to find a server that is closer to the data. One of the fundamental changes between Version 1.5 of RWhois and this version is that the reduction rules for a given schema are much more codified. For any given schema definition, a given A/V pair is denoted as being hierarchical in nature. Whenever this is noted, a rule must be given in the schema definition that specifically defines what the separation token for each reducible part is. The rule is given as either a Posix Extended Regular Expression or URI. The URI is used both to uniquely identify the rule being used and also to retrieve that rule if some method for doing so is used in the future.

5.3.1 Search Algorithm

1. Accept a query.
2. Find any local matches; display them.
3. Find any referrals; display them.
4. If no local or referral hits, reduce the query and go to Step 3.

Here is an example of the query ietf.cnri.reston.va.us:

Query whois for ietf.cnri.reston.va.us.

1. Search rs.internic.net for information (no hits).
2. Search referrals for ietf.cnri.reston.va.us (no hits).
3. Search referrals for cnri.reston.va.us (no hits).
4. Search referrals for reston.va.us (no hits).
5. Search referrals for va.us (no hits).
6. Search referrals for us (referral found)=97referral to isi.edu.

[currently on rs.internic.net:4343 for proof of concept]

5.3.2 Reduction Rules

As specified in the schema definition section, each object type must specify which of its A/V pairs are hierarchical (e.g., reducible). With that specification comes a regular expression that states what the token is for understanding what the separator is between each of the reducible parts. The default will be "/\./", which matches a period. This default is used for the majority of the searches, as it is used for domain names and IP addresses. A good example would be HTTP-based URLs. Its rule would match on "/" and "." thus giving a way to resolve URLs based on other criteria than simply hostname and path. Other examples are ISBN numbers and OIDs.

5.4 Referral

The result profile referral instructs the client to query another server, which could be a Whois, RWhois, LDAP, or Whois++ server. Since the primary referral field is in the form of a URI, the referral can happen to any server via any protocol. If this is the case, many of the assertions made here with respect to up or down referrals are considered meaningless.

Referrals are cumulative, and the first referral received during a session must replace the default server list. Any subsequent referrals received must be appended to the end of the server list. The type of a referral can be determined by the client evaluating the authority area, which is part of the referral RWhois URI response.

If the authority area received from a referral response is equal to the original query, then it is a link referral. If the authority area is not equal to the query, then it is a reduction referral. If no authority area is sent, then it is a punt referral. Punt means the server is not a root and cannot answer the query, and is therefore referring the client to a level up the tree or to a server that can better answer the query. NOTE: The punt referral may be used to direct a client into a CIP mesh.

The client may receive multiple referrals from a single query. If the SOA for each of these referrals is the same, then the first referral is the primary server and all subsequent servers are secondary. Each of the servers will report the SOA for the authority area in question.

5.5 Other Elements Affecting Query Completion

Earlier versions of RWhois contain the concept of a 'see-also'. This response was used to direct a client to additional information outside an RWhois tree. In this version of the protocol, this functionality is handled by the text/directory MIME type through its use of 'value=3Durl' attribute parameters. For example, a given record has the attribute Photo that contains the image of the contact. This image is not returned with the record. Instead, the client receives a record that has the following attribute:

Photo;value=3Durl: http://www.foo.com/bla.gif

This means that the actual value for the Photo attribute is located at the address specified in the URI.

In RWhois 1.5, this concept was also extended to the specific type of see-also that dealt with objects within the RWhois tree. This type of value was used to normalize databases and to have records share values. In this version the text/directory MIME type also handles this simply by using the RWhois URL as the URL in a value=3Durl attribute parameter. For example:

Admin-contact;value=3Durl: rwhois://rwhois.internic.net/BN123.NET

6.0 Security Model

There are several areas within the protocol where security is needed. There is the need for the privacy of the stream, authentication of the reading and writing of objects, and the execution of certain directives. Security in RWhois is an area that needs further development. The issues discussed below are a guide for what needs to be done, not how to do it.

6.1 Stream Privacy

There is the need to secure the flow of data between the client and server to thwart eavesdroppers looking for secure information. There are two possible solutions: encrypt the protocol stream at the protocol level or at the stream level. The former would require RWhois to provide some handshaking. The latter would use existing solutions such as Transport Layer Security (TLS). The primary issue to be aware of is the desire to not require the server to buffer an entire answer to encrypt it. In the case of an Xfer this would be an impossibly large task.

6.2 Authentication of Reading/Writing of Objects

There is a need to authenticate clients who need to create, update, or delete objects. Currently, the Guardian model handles this to a certain extent. That model does need to be extended due to its inability to deal with attribute control lists so that certain values are visible depending on the viewer.

7.0 Tree Management

Data replication and tree management for RWhois 2.0 currently follows the same procedures and models as RWhois 1.5. See Section 3.6 of that RFC for further information.

8.0 References

[RFC-954] Harrenstien, K., Stahl, M., Feinler, E., NICNAME/WHOIS, RFC 954, SRI, October 1985.

[RFC-1521] Borenstein, N., Freed, N., MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies, RFC 1521, Bellcore, Innosoft, September 1993.

[RFC2167] Williamson, S., M. Kosters, D. Blacka, J. Singh, K. Zeilstra. Referral Whois (RWhois) Protocol V1.5. RFC 2167, Network Solutions, June 1997.

[RFC1714] Williamson, S., M. Kosters. Referral Whois Protocol (RWhois). RFC 1714, InterNIC, November 1994.

9.0 Authors' Addresses

The following employees of Network Solutions were involved in the discussions used to formulate the ideas expressed in this document. While specific credit is difficult to express, Scott Williamson and Mark Kosters are recognized for their original work with the first version of RWhois.

David Blacka davidb@rwhois.net
Mark Kosters markk@rwhois.net
Ming Lu cming@rwhois.net
Leslie Meador lesliem@rwhois.net
Michael Mealling michaelm@rwhois.net
Greg Pierce gregp@rwhois.net
Amar Rao amarr@rwhois.net
Jasdip Singh jasdips@rwhois.net
Scott Williamson scottw@rwhois.net
Koert Zeilstra kzeil@rwhois.net

Appendix A: Error Codes

The definitions of the error codes are provided below. The error codes are descriptive so that the client can group the error messages to determine group action that must be taken before taking error specific action. Error codes should remain consistent without variable extensions except for messages associated with the registration process. If a client receives a '6' in the second position of the error code and the client does not support the extended code received, the client must act on the first position code. (Example: If a client received 561 and the client did not support the extended error codes for the server currently connected, the client would exit based on the '5' in the first position of the error code.)

X00

* 1 - Information only, no action required
* 2 - Information, action required
* 3 - Specific directive error, retry that directive or try another directive
* 4 - Serious for current directive, may correct with another directive
* 5 - Fatal, must disconnect

0X0

   * 0(1) - System wide, no specific directive
   * 2 - Registration error
   * 3(4,5) - Specific directive
   * 6 - Extended message (version specific)

00X

Sequential order

Appendix B: Capabilities ID

The server and client must send their capabilities to each other during the initial handshaking after connection. The server indicates which optional directives that it can accept. The client responds with the server responses that it can understand. Below is the capabilities list.

Directives: (Sent by the server)

Directive	ID
load	1h
limit	2h
schema	4h
xfer	8h
quit	10h
status	20h
cache	40h
holdconnect	80h
forward	100h
soa	200h
notify	400h
register	800h
object	1000h
define	2000h
private	4000h
X-	8000h
directive	10000h
display	20000h
language	40000h

Appendix B: General ABNF definitions

These definitions are used to define several widely used tokens.

alpha =3D "a".."z" / "A".."Z"
digit =3D "0".."9"
hex-digit =3D digit / "a".."f" / "A".. "F"
id-char =3D alpha / digit / "_" / "-"
any-char =3D <ASCII 1..255, except LF (linefeed) and CR (carriage return)>
dns-char =3D alpha / digit / "-"
email-char =3D <see [RFC 822]>
space =3D " "

```
tab =3D <ASCII TAB (tab)>
lf =3D <ASCII LF (linefeed)>
cr =3D <ASCII CR (carriage return)>
crlf =3D cr lf
```

Grammar

```
year =3D 4digit
month =3D 2digit
day =3D 2digit
hour =3D 2digit
minute =3D 2digit
second =3D 2digit
milli-second =3D 3digit
host-name =3D dns-char *(dns-char / ".")
ip-address =3D 1*3digit "." 1*3digit "." 1*3digit "." 1*3digit
authority-area =3D 1*id-char
rwhois-id =3D 1*id-char "." authority-area
email =3D 1*email-char "@" host-name
host-port =3D (host-name / ip-address) ":" 1*5digit
class-name =3D 1*id-char
attribute-name =3D 1*id-char
attribute-value =3D 1*any-char
time-stamp =3D year month day hour minute second millisecond
on-off =3D "on" / "off"
```

Note that the time-stamp must be in the Greenwich Mean Time (GMT) time zone.