Network Working Group
Request for Comments: 1714
Category: Informational

S. Williamson
M. Kosters
Network Solutions Inc./
InterNIC

November 1994

Referral Whois Protocol (RWhois)

Status of this Memo

This memo provides information for the Internet community. This memo does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

Abstract

This memo describes version 1.0 of the client/server interaction of RWhois. RWhois provides a distributed system for the display of hierarchical information. This system is hierarchical by design, allowing for the reduction of a query, and the referral of the user closer to the maintainer of the information.

Table of Contents

1.0 Introduction

Early in ARPANET development, the SRI-NIC established a centralized whois database that provided host and network information about the systems connected to the network and the E-mail addresses of the users on those systems. The ARPANET experiment has evolved into a global network with countless people and hundreds of thousands of end systems. Given the sheer size and effort needed to maintain a centralized database, an alternate, decentralized approach to store and display this information is desired.

The Internet portions of the DDN NIC have been transitioned to what is now known as InterNIC Registration Services (RS). The charter for InterNIC RS has been reduced to maintain information only for IP networks, top-level domains, Autonomous System Numbers, and the points of contact for each of these particular entities. In addition, the InterNIC, in its role as an Internet Registry (IR), has delegated IP block assignment authority to Regional Registries such as the RIPE NCC for Europe and the APNIC for the Asian Pacific region, while retaining authority for North America and all non-delegated regions. This has led to a fragmentation of whois service to the Internet user.

Several different solutions have been proposed and developed by the various regional IR's. Two solutions have been worked on extensively: the Shared Whois Project (SWIP) and X.500.

The SWIP project has a common exchange format that can be parsed by the various IR's for input and output. Thus, one can synchronize their databases with information obtained from the other IR's. This project is showing promise and is now operational. However, this approach still requires a centralized database for store and display.

The InterNIC has also been involved in the use of X.500 for display of registration information. Among other things, this included defining schemas and Directory information tree structures for the purpose of distributing information amongst the various IR X.500 Directory Service Agents (DSA). Unfortunately, X.500's complexity, resource utilization, and lack of Internet support has made a search for an alternative solution necessary.

The information that the various IR's maintain is inherently hierarchical in nature. (Examples: hammer.nic.ddn.mil is under the nic.ddn.mil domain which is under the ddn.mil domain which is under the .mil domain. 198.41.0.21 is part of network 198.41.0.0/24 which is part of the block 198.41.0.0/16 which is part of the block 198.0.0.0/8) The InterNIC may not have the information, but will at least be able to reduce the query and point or "refer" the users closer to their goal. This has led to the development of a referral whois, and the corresponding RWhois protocol.

The underlying premise for this project has been to retain the basic functionality of the whois server and client, making all of the extensions optional. The server must respond to the original whois client, currently included with many operating systems. The RWhois client must also interact with RFC954 [RFC-954] whois servers.

RWhois has been designed as an extensible protocol to ensure that many uses can be accommodated. Public extensions to the protocol should be documented as RFCs. Private extensions can be used with agreement left up to the client and server.

If extensions are not implemented at the server in question, an appropriate error message must be sent. The use of extended error message is outlined in Section 5 - *Error Codes*.

Throughout this document the following notations will be used to describe the RWhois server/client interaction:

`<SP>`	space
`[arg]`	optional argument
`<arg>`	required argument
`(<arg>)`	conditional required argument
`([arg])`	conditional optional argument
`{format}`	format of item
`\`	continued on next line

The words “should” and “must” are significant in this document. If “should” is used, the implementor has the option to follow the advice of this document. If “must” is used then it is a required part of the protocol. Implementations without this functionality may not interact correctly with other RWhois servers.

The format descriptions throughout this document use macro definitions described in Section 6.1. Refer to that section for clarification.

The RWhois protocol specified in this document can be extended to accommodate such applications as NetHelp and ZoneGen (DNS zone generator).

2.0 RWhois Client Model

The RWhois design requires compatibility with the current Whois and Whois++ servers. Therefore, the RWhois client must wait or have knowledge of server type to determine if the server contacted is an RWhois server. The user should have control over the time the client waits, since this will vary based on network congestion and capacity. If after the wait the server does not respond with the *“%RWhois”* response, the client must not send any RWhois extended directives. In this case, the client should only send the query. We realize that the server identification feature may mean that the identity of an RWhois server may be missed. However, it will allow the RWhois system to utilize the current Whois and Whois++ infrastructure. Referrals from RWhois can be directed toward a Whois or Whois++ server. These non-RWhois servers must be placed as a leaf on the hierarchical tree. These servers represent a mesh structure from the RWhois perspective. This restriction should not discourage the use of these servers in building the RWhois structure.

The RWhois server must remain connected until a query is received. If the client wishes to make multiple queries it must send the *"-holdconnect"* directive. In this mode, once the client has sent the last query and received either an answer or the error code indicating that no records were found, it must issue the *"-quit"* directive. If the client only wishes to issue directives, then upon completion the *"-quit"* directive must be sent. If it is not sent, the server will wait until it receives non-directive input from the client.

Considering the requirement for compatibility with the original whois, the RWhois client in default mode must operate exactly like the current Whois client. However, in the enhanced mode, the RWhois client can do much more based on information received from the RWhois server.

2.1 Directives: Client to Server Interaction

The RWhois client sends directives to the RWhois server. These directives are prefaced with the '-' character always at the start of a new line. However, for compatibility with older Whois clients, the query is not prefaced with the '-' character. Only after the client is certain that the server is an RWhois server should these directives be sent. Compatibility with RFC954 [RFC-954] whois servers is required. All directives must be terminated by <LF><CR>.

2.2 Required Directives

The following are *required* RWhois client directives.

2.2.1 <query>

The query is generally the final directive sent to the server. It is the only directive that does not start with a '-'. The query is the question that the client wants the server to answer. The qualifiers that may proceed the query are addressed in Section 3.1 - *Output Display and Restriction Keywords*.

Format for use:

```
[display format]<SP>[query restriction]<SP><query>
```

`[Display format]{%s}`	This optional pre-query directive allows the requester to select the format of the returned data. Details of the allowable values can be found in Section 3.1.

`[Query restriction]{%s}` This optional pre-query directive allows the requester to limit the area in which the servers search for a specific object.

Example of use:

```
dump domain netsol.com
```

2.2.2 RWhois

The *"-RWhois"* directive identifies the client as an RWhois client allowing the server to operate using the RWhois protocol exclusively.

Format for use:

```
-RWhois<SP>V-<spec version #><SP>[imp identifier]
```

`<Spec version #>{%2d.%2d}` This required argument identifies the specification version that the client is built to conform with. Clients that are built in accordance with this document are V-1.0. This argument will be used by the server to determine if features introduced in subsequent releases of the protocol document may be used.

`[Imp identifier]{%s}` This optional argument identifies client implementation information. It is recommended that the implementor maintain a version number separate from the specification version.

Example of use:

```
-RWhois V-1.0 [InterNIC B.0.9.7]
```

2.3 Optional Directives

The following are OPTIONAL RWhois server directives.

2.3.1 load

The *"-load"* directive allows the client to make a quick decision about presenting the query to the current server. If the client determines that another server can better serve the query, then control may be transferred to the server with the lower load and better connection. This directive has no arguments.

2.3.2 limit

The *"-limit"* directive will allow the client to request the server allocate enough space to collect more responses than would currently be collected by the server.

Format for use:

```
-limit<SP><value>
```

`<Value>{%d}`	This required argument is the new limit requested by the client. If the limit exceeds the limit set by the server administrator, the client must receive an error message. It is recommended that if the client receives an error for exceeding the servers upper limit, it should cut the request in half and resend the request until an acceptable level has been negotiated.

Example of use:

```
-limit 2000
```

2.3.3 schema

One of the shortcomings of X.500 was the requirement to know the schema of an object before making a query. RWhois allows the client to request the schema for an object without knowledge of the object by using the *"-schema"* directive.

Format for use:

```
-schema<SP>[object]
```

`[object]{%s}`	This optional argument identifies the objects for which the schema is being requested. If this argument is not sent, the schemas for all objects contained in the server will be sent.

Example of use:

```
-schema domain
```

2.3.4 xfer

The *"-xfer"* directive is used to transfer all data from a server. This method of transfer has no limit on the number of records that can be transferred to the client application. This directive is primarily used to transfer data contained in an authority area for caching at a secondary server.

Format for use:

```
-xfer<SP>[object]<SP>[authority area]<SP>[SOA]
```

`[Object]{All|%s}`	This required argument identifies the object to transfer. If the keyword "all" is sent, all objects contained in the server will be transferred. Otherwise, only the object specified will be sent.
`[Authority area]{%s}`	This optional argument contains the authority area of the object to send further limiting the data transfer.
[SOA]{%d}	This optional argument notifies the server to send everything that has been updated since this SOA number.

Example of use:

```
-xfer domain netsol.com
-xfer domain netsol.com 19940818141259
```

2.3.5 quit

The *"-quit"* directive will inform the server that the client is finished. The server and client should close the connection. This directive has no arguments.

2.3.6 status

The *"-status"* directive is used to poll the server for its status. There are seven required responses to this directive. Additional attributes may be sent in the response. The client should ignore all unknown attributes. This directive has no arguments.

2.3.7 cache

The RWhois server can hold data that it has no authority over. If the server sends this data to a requester, it is considered a non-authoritative response. The holding of this data is called caching. The physical data for these objects is not contained on the system hosting the server. The *"-cache"* directive allows the client to instruct the server whether or not to send cached data. The RWhois client should start with the cache turned off. The server must start with the cache turned on in order to function like the RFC954 [RFC-954] whois server. Because of the server's default, the client should send the *"-cache off"* directive during initial session setup if cached data should not be sent. Details on expiration of cache data can be found in section 3.4.3, *"%soa"* response.

Format for use:

```
-cache<SP><mode>
```

```
<mode>{on|off}
on:  Turns caching on.
off: Turns caching off.
```

Example of use:

```
-cache on
```

2.3.8 holdconnect

The RWhois server must close the connection after the response to a query has been received. The query is the final exchange between the client and server. However, this characteristic can be modified with the *"-holdconnect"* directive. If this directive is issued to the RWhois server, it will remain connected until the *"-quit"* directive is received. Once the *"-quit"* directive is received, both the server and the client must close their connection.

Format for use:

```
-holdconnect<SP><mode>
```

```
<mode>{on|off}
On:  Turns holdconnect on.
Off: Turns holdconnect off.
```

Example of use:

```
-holdconnect on
```

2.3.9 forward

During normal sever operation the server will send *"%referral"* or *"%see-also"* responses to the client, expecting the client to redirect the query to the server identified in the response. If the client is located behind a firewall or is poorly connected, having a server make the query may improve query performance or allow a query to be satisfied. The *"-forward"* directive will instruct the server to operate as a forwarding server. Whether or not this directive should be allowed should be a configuration parameter of the server.

Format for use:

```
-forward<SP><mode>
```

```
<mode>{on|off}
On:  Turns forwarding on.
Off: Turns forwarding off.
```

Example of use:

```
-forward on
```

2.3.10 soa

The identification of authority area is an important part of the RWhois design. The *"-soa"* directive is used to question the server's authority for a specific area. A positive response will include the administrative parameters for the authority area as detailed in section 3.4.3. If the server does not contain an SOA for the authority area requested, it must send an error message to the client.

Format for use:

```
-soa<SP>[authority area]
```

`[Authority area]{%s}` This optional argument identifies the authority area being requested. If this argument is not sent, information about all authority areas contained in the server must be sent.

Example of use:

```
-soa netsol.com
```

2.3.11 notify

The *"-notify"* directive is used to notify a server of a bad or recursive referral or a change in a primary server's data.

Format for use:

```
-notify<SP><action><SP><information>
```

```
<action>{badref|recurref|update|inssec|delsec}
```

`badref` When a client receives a *"%referral"* response that does not work, it must report the bad referral to the server that issued the referral. The referral is bad only if the referred server does not contain the SOA record for the authority area in question. It is not considered a bad referral if the server does not have an answer to the query, but responds positively to the *"-soa area"* directive. This merely means that there is not an answer to the query. When a *"-badref"* is sent to the referring server; it should log the bad referral so the administrator of that server can remove the reference if it is no longer correct. This action should only be taken after receiving a negative response to the query and the SOA request.

`recurref` When a client receives a referral that results in a recursive action, the referring server must be informed. The "-recurref" directive must be sent identifying the recursive loop. This directive should only be sent to the server one level back, even if multiple server were involved in the referral.

`update` An RWhois primary server must be aware of its secondary servers. If the data in the primary server changes, the primary server may choose to notify the secondary servers. This allows the secondary servers to quickly reflect changes in the primary server's data.

`inssec` This action will inform the authority server that the server indicated in the argument will be a secondary for its authority area. The server receiving this directive must determine if the secondary is acceptable. If it is, the server should be added to the update list so that it will be informed if data in the authority area changes.

`delsec` This action will inform the server that the server indicated in the subsequent arguments will no longer be a secondary. The server receiving this action must determine if the server is a secondary and if so, remove it from the update list.

```
<information>{action=badref|recurref <<server>:<query>>
             action=inssec|delsec|update
             <<server>:<object>:<authority>>}
```

`<server>{%Mserver}` This required argument identifies the server that contained the recursive or bad referral, or has data that changed.

`<query>{%s}` This required argument identifies the query that was sent to the server that gave a recursive or bad referral.

`<object>{%s}` This required argument identifies the object that changed.

`<authority>{%s}` This required argument identifies the authority area where the object that changed currently resides.

```
Example of use:

-notify recurref netman1.netsol.com:4343:scottw@netsol.com
-notify badref nic.ddn.mil:43:abc.af.mil
-notify update netman1.netsol.com:4343:domain:netsol.com
-notify inssec dmeister.internic.net:4343:domain:netsol.com
-notify delsec dmeister.internic.net:4343:domain:netsol.com
```

2.3.12 register

This directive allows the client to add, modify, or delete information that exists or should exist in the server's database. During the exchange, all attributes of an object must be sent. The client must wait to send the registration data until the *"%ok"* response is received from the server.

Format for use:

```
-register<SP><mode><SP>(on:<action><SP><e-mail contact>
       <SP><authority info>)
```

`<mode>{on|off}`

`on:` This required argument starts the registration process.

`off:` This required argument ends the registration process.

The following arguments are only required if the mode argument is sent with the value "on."

`(<action>){add|mod|del}`

`add:` This conditionally required argument indicates that the object being sent should be added to the server's database.

`mod:` This conditionally required argument indicates that the object being sent should be modified and should already exist in the server's database.

`del:` This conditionally required argument indicates that the object being sent should be deleted from the server's database.

`(<e-mail contact>){%Memail}` This conditionally required argument identifies the sender of the registration information.

`(<authority info>){%s}` This required argument contains information used to authenticate the person sending the registration information. The method used must be identified using the *"-private"* directive. Work must be done to identify usable authentication methods for unsupervised delegation. This is beyond the scope of this document. However, the authors have made an effort to allow flexibility in the implementation of an authentication system.

Example of use:

```
-register on add scottw@netsol.com
Object-type:referral
Referral:netman1.netsol.com:4343
Domain-Name:netsol.com
IP-Network:192.153.247.0
IP-Network:198.41.0.0
-register off
```

2.3.13 object

RWhois data is a collection of objects with defined attributes. The attributes for an object can be acquired by issuing the *"-schema"* directive. Each object must at a minimum define the attribute "object-type." This attribute identifies the name of the object that will be displayed in response to the *"-object"* directive. This directive can be used by a client to verify that a server contains the desired object. Another possible use may be to gather all of the objects contained on a server and display them to the user in the form of a menu for selection.

Format for use:

```
-object<SP>[object]
```

[object]{%s} This optional argument identifies the object requested. If no argument is sent, all objects contained in the server will be returned.

Example of use:

```
-object domain
```

2.3.14 define

Format strings describing the format of an object's attribute may include format macros. More information about definitions of format macros can be found in Section 6. The *"-define"* directive allows the client to request the definition of a format macro.

Format for use:

```
-define<SP>[macro name]
```

[macro name]{%s} This optional argument identifies the name of the macro to display. If no arguments are sent, the server must return the definition of all macros contained in the server.

Example of use:

```
-define server
```

2.3.15 private

The *"-private"* directive allows the client to identify the authentication method to be used. More research needs to be done with respect to client authentication. This directive will allow more experimentation.

Format for use:

```
-private<SP><action><SP><method><SP>[data]
```

<action>{auth|encr} This required argument identifies the action the directive is taking. Currently the value for this argument can be "auth" for authentication or "encr" for encryption.

<method>{%s} — This required argument contains the name of the method to be used. The value must be recognized by the server or an error will be sent. It is beyond the scope of this document to identify the possible method to be used.

[data]{%s} — This optional argument must be supplied if required by the method identified in the previous argument.

Example of use:

```
-private auth pass1 xxjdk_998uu
```

The above example is a simple password exchange. It is beyond the scope of this document to determine the authentication technique that would best suit this protocol. Development is underway to determine the authentication needs and to experiment with potential solutions.

2.3.16 X-

This directive is the preface to extended directives, mutually agreed to between the client and server. The client and server must have knowledge of the extended directives to use. Extension can accommodate other uses such as NetHelp, white pages, and many others. If the extensions are public, they should be documented in an RFC and available through the *"-directive"* directive.

Format for use:

```
-X-<directive name><SP>[directive arguments]
```

<directive name>{%s} — This required argument identifies the name of the directive being issued.

[directive arguments]{?} — This optional argument is dependent upon the required or optional arguments of the extended directive. There may be multiple directive arguments.

Example of use:

```
-X-date
```

2.3.17 directive

Directives allowed by a server may vary. The client can issue the *"-directive"* directive to determine if the server allows a specific directive or to obtain a list of all acceptable directives for that server.

Format for use:

`-directive<SP>[directive]`

`[directive][%s]` This optional argument identifies the directive being requested. If no arguments are sent, all of the directives accepted by the server must be sent.

Example of use:

```
-directive X-date
```

2.3.18 display

The *"-display"* directive is used to set the display mode of the server or to identify display modes the client is capable of. If this directive is sent without arguments, the server will return all available display methods.

Format for use:

`-display<SP>[action]<SP>[method]`

`[action]{activate|capable}` The 'activate' setting enables a certain display mode, while a 'capable' setting sends the display mode the client is capable of.

`[method]{%s}` This optional argument indicates the display method desired by the client.

Example of use:

```
-display swip
-display mime
```

2.3.19 language

The *"-language"* directive is used to set the language mode of the server or to identify language modes the client is capable of. If this directive is sent without arguments, the server will return all available languages.

Format for use:

`-language<SP>[language]`

`[language]{%s}` This optional argument indicates the language desired by the client.

Example of use:

```
-language german
```

2.4 RWhois Client Model

```
Server <-------> Client

START:
<------ Connection (record time to connect)
        If no server type...Wait up to specified
         time for------> "%RWhois" response
          (recommend wait of at least 5 seconds)

if  "%RWhois" is not received from server, assume that it is
 not an RWhois server
    goto QUERY:

else if "%RWhois" is received from server
    <------- send "-RWhois -VX.X"
    --------> receive "%ok"
    DIRECTIVE: if directive for server
               <------- send directive
               -------> receive server response
               if "%ok" received
                 goto DIRECTIVE:
               if "%error" received
                 process error then goto DIRECTIVE:
             else if no more commands for server
                goto QUERY:
QUERY:
    <-------- send query
    --------> Receive and display response
    PROCESS: if "%referral" received
                if first referral
                   restart server list
                else
                   add to server list
            if "%see-also" received
                insert server into server list
            if in holdconnection mode
                goto DIRECTIVE:
            if no directive (%)
                goto END:
            goto PROCESS:
END:
server will disconnect
if more servers on Queue and multi or referral mode active
    goto START:
```

Every time the RWhois client receives a *"%referral"* or *"%see-also"* response from the RWhois server it must compare the host:port:query with those already executed. If the client discovers that it is being directed to repeat the same query to a server that it has already visited, it must not repeat that query. As an example, the prototype RWhois client maintains a server trail and compares each new directive with the entire list. If a recursive act is about to occur, the client will notify the user and exit.

The original Whois client opens a TCP connection, sends the query, and displays the response. The RWhois client must be more robust in order to handle multiple server queries, servers that do not exist, and recursive referrals. The client must also remain connected while sending directives and receiving responses. All of these features have been incorporated into the experimental RWhois client.

3.0 RWhois Server Model

This section describes the functionality of the RWhois server.

3.1 Output Display and Restriction Keywords

The RWhois server will behave similarly to the original whois server in terms of display formats and restrictions. The following are required in the RWhois server.

Display Format Keywords

`EXPand (*)`	Expand
`~`	no sub displays
`SUBdisplay (%)`	sub displays
`SUMmary ($)`	Give a short summary for the query on one to many hits (defaults on multiple hits).
`Full (=)`	Give the full record output on one to many hits (defaults on one hit only).

The following was added to whois post RFC954 [RFC-954] and is part of the RWhois requirements:

`dump (#)`	Display the record in a parsable format.

In addition to the above, the RWhois server must accept additional pre-query directives such as Boolean queries and attribute=value query combinations. The capability to perform partial matches are requested by post fixing a '*' or '.' at the end of the search item for unknown characters. This capability is required for an RWhois server.

Example: `last-name=williamson and first-name=scott`

Data Restriction Format Keywords

The following restriction keywords are found in the RFC954 [RFC-954] whois server:

```
!(handle) Query on Handle only
mailbox   Query on all records for person
person    Query on User records only
host      Query on Host records only
domain    Query on Domain records only
network   Query on Network Records only
asn       Query on Autonomous System Numbers only
```

The RWhois server must allow restriction of search to any object contained on that server. With the exception of the '!' restriction format keyword, the above listed restriction keywords represent defined objects. In the prototype software, each of these objects are defined in configuration files, not hard-coded into the server. New objects, and therefore restriction keywords, should be easily designed with no code change necessary to the server.

3.2 Responses: Server to Client Interaction

Responses are sets of data that servers send in response to a client directive. Responses from an RWhois server must be prefaced with the '%' character at the start of a line. Responses are divided into two groups: those that are required to provide minimal RWhois interaction and those that are used to achieve the desired characteristics of a fully functional distributed system. A server must respond with an error message indicating that a directive is not available on the server and therefore does not have the required responses.

3.3 Required Responses

The following sections describe the *required* RWhois server responses.

3.3.1 RWhois

The *"%RWhois"* response is used to identify a server as an RWhois server. Clients that treat RWhois servers differently will need this response to enable the RWhois capabilities.

Format for use:

```
%RWhois<SP>V-<Spec version #><SP><server name><SP>[imp name and
version #]
```

`<Spec version #>[V-%2.2f]`	This required response indicates the version number of the RWhois protocol specification that the software is capable of handling. The version described in this document is V-1.0.
`<server name>[%s]`	This required response is the host name of the computer hosting the RWhois server.
`[imp name and version #][%s]`	This optional argument 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 rs.internic.net (Network Solutions V-1.6)
```

3.3.2 referral

The *"%referral"* response instructs the client to query another server (which could be a whois, RWhois, or whois++ server). Referrals are cumulative. 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.

In the non-Uniform Resource Location (URL) response format below, the authority area equals the reduced query. There are three types of referral. The type can be determined by the client evaluating the authority area which is part of the *"%referral"* response.

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

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.

Format for use:

```
%referral<SP><server>[:type]<SP>[authority area]
%referral<SP>url:<url>
```

<server>{%Mserver}	This required argument identifies the server that the client should re-connect with.
[type]{%Mstype}	This optional argument identifies the server type. This could save wait time for the client trying to identify a server which is non-RWhois.
<authority area>{%s}	This optional argument identifies the authority area that caused the referral for the query in question. Using this value as the argument for the *"-soa"* directive to the referral server should result in a positive response. If this is not the case, the referral is considered bad.
<url>{%Murl}	This required argument defines the Uniform Resource Location (URL) string that points to the resource containing the information desired.

Example of use:

```
%referral nic.ddn.mil:43 .mil
%referral url:http://www.netsol.com/
```

3.3.3 ok

The *"%ok"* response must be sent by the RWhois server at the completion of every task or to positively acknowledge a directive.

Format for use:

```
%ok
```

3.3.4 error

The *"%error"* response is used to indicate an error condition to the client. Refer to Section 5 for details on the error reporting scheme. It is important to note that only the error number will be used to determine the client's action. The text message will only be used to make the error readable by humans connected using telnet or an old whois client. The only exception to this rule is the error message used to indicate problems with registration transactions. The format for these message can be found in Section 5.

Format for use:

```
%error<SP><error number><SP>[error text]
```

`<error number>{%d}`	This required argument is the error number identified in Section 5. The client can use this number to categorize errors.
`[error text]{%s}`	This optional argument is the text that describes the error message. This message must be consistent for each error. Variables should not be added to this message. This message is only to make the error message human readable. Message sent following an error code associated with the registration process will contain the line number of the attribute that is incorrect.

Example of use:

```
%error<SP>400<SP>Invalid Server Directive
```

3.4 Optional Responses

The following are *optional* RWhois server responses.

3.4.1 see-also

The *"%see-also"* response instructs the client to contact another server for additional information about the current query. See-also servers should be inserted into the server list just after the current server. If multiple see-alsos are received from a single query, each subsequent see-also should be inserted after any other see-alsos previously received. See-alsos should be additional information related to the current query.

One example use for the see-also response is to display autonomous system information relating to an IP network number or router interface information relating to an IP host number.

Format for use:

```
%see-also<SP><server>[:type]:<query>
%see-also<SP>url:<url>
```

`<server>{%Mserver}`	This required argument identifies the server the client should reconnect with.
`[type]{%Mstype]`	This optional argument identifies the server type. This could save wait time for the client trying to identify a server which is non-RWhois.

`<query>{%s}`	This required argument sets the query that must be sent to the referred server. The query may be different from the original query sent to the referring server.
`<url>{%Murl}`	This required argument defines the Uniform Resource Location (URL) string that points to the resource containing the information desired.

Example of use:

```
%see-also prmd.merit.edu:43:handle=xxx
%see-also url:http://www.netsol.com/
```

3.4.2 load

The *"%load"* response returns the current and average load of the computer hosting the RWhois server. We realize that the measurement may be different depending on the implication of the system's load mechanism. This directive/response was implemented to allow experiments with sorting preferred servers to deliver better results to the user.

Format for use:

```
%load <SP><current><SP><average>
```

`<current>{%2.2f}`	This required argument delivers the current load on the system hosting the RWhois server.
`<average>{%2.2f}`	This required argument delivers the average load on the system hosting the RWhois server.

Example of use:

```
%load 5.68 1.32
```

3.4.3 soa

The *"%soa"* response delivers information about the authority area in question. If the server does not contain the authority for the area in question, it must respond with the appropriate error message. The SOA data must never be cached. SOA records must originate on the server giving the answer. The increment and refresh attributes are used to provide for incremental updates of the secondary server. Deleted data will remain in the secondary server's cache until the refresh time has been reached. This will reduce the amount of data transferred and not require the primary server to retain deleted data. The following are the minimum attributes required for the soa object:

```
object-type
authority
ttl
serial
refresh
increment
retry
tech-contact
admin-contact
hostmaster
primary
```

Format for use:

```
%soa<SP>authority:<SP><authority area>
%soa<SP>ttl:<SP><ttl>
%soa<SP>serial:<SP><serial>
%soa<SP>refresh:<SP><refresh>
%soa<SP>increment:<SP><incremental>
%soa<SP>retry:<SP><retry>
%soa<SP>tech-contact:<SP><tech-contact>
%soa<SP>admin-contact:<SP><admin-contact>
%soa<SP>hostmaster:<SP><hostmaster>
%soa<SP>primary:<SP><primary>
```

`<authority area>{%s}` The authority name of the SOA. (Example: internic.net or 198.41.0.0/24)

`<ttl>{%d}` The time to live for data within this authority area that another server may cache. The server caching the data should consider the data expired after storage for the number of seconds identified by this attribute.

`<serial>{%Mserial}` Serial number of the data contained in the authority area. The serial number must be incremented every time data in the authority area has changed. It must be numeric.

`<refresh>{%d}` The time to completely remove cached data and transfer all data from the primary server.

`<increment>{%d}` The time to wait before checking for incremental updates from a primary server.

`<retry>{%d}` The time to wait before retrying connection to a server that appears to be out-of-service.

`<tech-contact>{%Memail}` E-mail address of the person or role account responsible for the operation of the server.

`<admin-contact>{%Memail}`	E-mail address of the person or role account responsible for the data contained on the server.
`<hostmaster>{%Memail}`	E-mail address to which changes of the server's data should be sent. A data edit tool can automatically send changes to update the data on the server via e-mail.
`<primary>{%Mperm}`	Primary server for authority area.

Example of use:

```
%soa authority: netsol.com
%soa ttl: 7200
%soa serial: 19940606203030
%soa refresh: 7200
%soa increment: 60
%soa retry: 1200
%soa tech-contact: markk@netsol.com
%soa admin-contact: stanb@netsol.com
%soa hostmaster: hostmaster@netsol.com
%soa primary: netman1.netsol.com:4343
%ok
```

3.4.4 status

The *"%status"* response returns the status of several important flags or values. The response must contain the following elements.

`Limit:`	Current limit set on the server. This value may be changed using the *"-limit"* directive. This is not the maximum limit of the server. This value is not disclosed to prevent clients from automatically setting the highest limit possible, causing degradation in performance of the server.
`Load:`	This is the current load of the host system.
`Cache:`	Current status of the cache flag. (on or off)
`Holdconnect:`	Current status of the holdconnect flag. (on or off)
`Forward:`	Current status of the forward flag. (on or off)
`Authority records:`	Number of authority records in server's database.
`Cached records:`	Number of records in the server's cache database.
`Display:`	Indicates the types of display modes the server is using.

Format for use:

```
%status<SP>limit:<SP><current limit>
%status<SP>load:<SP><current load>
%status<SP>cache:<SP><cache>
%status<SP>holdconnect:<SP><holdconnect>
%status<SP>forward:<SP><forward>
%status<SP>Authority:<SP><SOA number>
%status<SP>Cached:<SP><cached number>
%status<SP>Display<SP><mode>:<SP><type>
```

See above for the description of these values.

```
<current limit>{%d}
<current load>{%2.2f}
<cache>{on|off}
<holdconnect>{on|off}
<forward>{on|off}
<SOA number>{%d}
<cached number>{%d}
<mode>{single|multi}
<type>{%s}
```

Example of use:

```
%status limit: 1500
%status load: 1.23
%status cache: off
%status holdconnect: on
%status forward: off
%status Authority:25
%status Cached:200
%status display multi: summary
```

3.4.5 xfer

The *"%xfer"* response will send all instances of an object. This is in response to the *"-xfer"* directive. The transfer may be limited by the arguments to the directive. If there are no arguments, the server must send all of the objects in the database. Cached data must not be transferred using this method unless caching is turned on.

Each object instance is sent with a blank %xfer response between instances.

Format for use:

```
%xfer<SP>[<object>:<attribute>:<value>]
```

These arguments are not required if the current response is an object instance separator.

`<object>{%s}`	This required argument represents the name of the object being transferred.
`<attribute>{%s}`	This required argument identifies the attribute being sent.
`<value>{%s}`	This required argument contains the value of the attribute. If blank, the attribute value is blank.

Example of use:

```
%xfer user:last-name:Kosters
%xfer user:first-name:Mark
%xfer user:organization-phone:703-555-1212
%xfer
%xfer user:last-name:Williamson
%xfer user:first-name:Scott
%xfer user:organization-phone:703-555-1212
%xfer
```

3.4.6 schema

The *"%schema"* response is used to describe the attributes of an object. This is in response to the *"-schema"* directive.

Each attribute is sent with a blank *"%schema"* as a separator.

Format for use:

```
%schema<SP><object>:attribute:<attribute name>
%schema<SP><object>:format:<format string>
%schema<SP><object>:description:<descriptive string>
%schema<SP><object>:indexed:<indexed>
%schema<SP><object>:required:<required>
%schema<SP><object>:multi-line:<multi-line>
%schema<SP><object>:type:<type>
%schema<SP><object>:primary:<primary>
%schema
```

These arguments are not required if the current response is an attribute separator.

`<attribute name>{%s}`	This required argument identifies the name of the attribute being described.
`<format string>{%s}`	This required argument describes the allowed format for the attribute.
`<descriptive string>{%s}`	This required argument describes the attribute's use.
`<indexed>{on|off}`	This required argument identifies attributes that are indexed.

`<required>{on|off}`	This required argument identifies attributes that are required.
`<multi-line>{on|off}`	This required argument indicates whether the attribute can span multiple lines.
`<type>{text|MIME|see-also|PostScript}`	This required argument identifies the type of the attribute.
`<unique-key>{on|off}`	This required argument indicates whether the attribute is a unique key.

Example of use:

```
%schema user:attribute:Object-Type
%schema user:description:Name of the object
%schema
%schema user:attribute:Email
%schema user:format:[%Memail]
%schema user:description:RFC-822 compliant Email address
%schema
%schema user:attribute:Organization-Phone
%schema user:format:[%3d[0-999]-%3d[0-999]-%4d[0-9999]]
%schema user:description:Work phone number
%schema
```

3.4.7 define

The *"%define"* response describes format macros to the client. All format macros used in the schema format definition string must be available to the client through the *"-define"* directive. Format macros may be nested. It is the client's responsibility to request all format strings that are unrecognized from a server. If the format strings change on a server, the serial number of the schemas that use the format must change.

Format for use:

```
%define<SP><macro name>:<[format string]>
```

[NOTE: The brackets around the format string are required to ensure that spaces contained in the format string are interpreted correctly by the client.]

Example of use:

```
%format server:[%s:%16Bd]
%format email:[%s@%s]
```

3.4.8 object

All visible objects on an RWhois server must be identified in response to a *"-object"* directive. The *"%object"* response either confirms the existence of an object or returns a complete list of all objects available to the currently connected user.

A blank "*%object*" line serves as an object separator.

Format for use:

```
%object
%object<SP><object name>:description:<object description>
%object<SP><object name>:restrict:<restriction words>
```

`<object name>{%s}`	This required argument is the name of the object.
`<object description>{%s}`	This required argument is a description of the object identified.
`<restriction words>{%s}`	This required argument is a list of words used to restrict a search to this object.

Example of use:

```
%object user:description:user records for entity POC
%object user:restrict:user
%object user:restrict:person
%object user:restrict:mailbox
```

3.4.9 directive

The *"%directive*" response is used to display directives allowed on the connected server. The directive name, description and syntax attributes must be sent for each directive. If information about a single directive is requested then only information about that directive must be returned.

A "*%directive*" response with no arguments must be sent between directives.

Format for use:

```
%directive<SP>directive:<directive>
%directive<SP>description:<description>
%directive<SP>syntax:<format>
%directive
```

The arguments below are required except when separating directives.

`<directive>{%s}`	This required argument indicates the name of the directive.

`<description>{%s}` This required argument describes the directive.

`<format>{%s}` This required argument defines the format of the directive.

Example of use:

```
%directive directive:schema
%directive description:displays schema attributes
%directive syntax:schema<SP>[%s]
%directive
%directive directive:xfer
%directive description:transfer all object[authority area]
%directive syntax:xfer<SP>[%s]<SP>[%s]
```

3.4.10 info

The *"%info"* response is used to give the user of the client a message. This response is not initiated by any directive. The information between the *"%info on"* and the *"%info off"* should be presented to the user of the client. An ideal use of this response is to present a Message of The Day (MOTD) to the user.

Format for use:

```
%info<SP><mode>
```

`<mode>{on|off}`
- `on:` Turns the passthru mode on.
- `off:` Turns the passthru mode off.

Example of use:

```
%info on
As of 3/24/1994 at 9:00 EST this server will no longer be in
service. If you have this server in your configuration file we
recommend that you change it to rs.internic.net:4343. You will
automatically be redirected there following this message.
%info off
```

3.4.11 display

The *"%display"* response is used to inform the client that the data following this response is using the indicated method. The method selected will continue to be active until a "*%display*" response is sent without any arguments. The server must send an error message to clients that have been identified as non-RWhois clients. This response allows the use of display methods such as MIME [RFC 1521] or other special character sets such as those used in the Japanese language.

Format for use:

```
%display<SP>extended:<extended>

%display<SP>name:<name>
%display<SP>length:<length>
%display<SP>description:<description>
%display<SP>command-line-option:<mode>
```

`<extended>` This optional argument identifies if the display method is extended, i.e., RWhois specific.

Example of use:

```
%display extended:mime
MIME-Version:1.0
Content-type: image/gif
Content-Transfer-Encoding: base64
...data...
%display
```

3.4.12 X-

The *"%X-"* response represents extended responses. The client must have prior knowledge of this response.

Format for use:

```
%X-<response><SP>[arguments]
```

`<response>{%s}` This required argument contains the response name.

Example of use:

```
%X-extstatus numusers:500
%X-extstatus avalslots:200
```

[NOTE: The above examples are not implemented in the current RWhois prototype software. They are only examples of the *"%X-"* response to a *"-X-"* directive. "X6X" error codes are used when problems are encountered in relation to the "-X-" directives contained on the server. Details can be found in Section 5.]

3.4.13 language

The *"%language"* response is used to inform the client that the data following this response will be sent in the indicated language. The language selected will continue to be active until a "*%language*" response is sent without any arguments, at which time the server will revert back to

English, the default. The server must send an error message to clients that have been identified as non-RWhois clients.

Format for use:

```
%language:<SP><language>
```

Example of use:

```
%language: german
RWhois Deutsche Version: 1.0
%language
```

3.5 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. The search algorithm of the server is the following:

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 goto step 3

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

```
 1) query whois for ietf.cnri.reston.va.us
 2) search rs.internic.net for information (no hits).
 3) search referrals for ietf.cnri.reston.va.us (no hits)
 4) search referrals for cnri.reston.va.us (no hits)
 5) search referrals for reston.va.us (no hits)
 6) search referrals for va.us (no hits)
 7) search referrals for us (referral found) - referral to
       isi.edu.
[currently on rs.internic.net:4343 for proof of concept]
```

3.6 Determining Authority

Authority areas are a major part of the RWhois protocol. If an authoritative response is required, turning the cache off is the first step. The client can also determine if the server connected has authority over the name/number space of interest by sending the *“-soa <authority area>”* directive. If the server has authority for the area requested, it must return important information about the authority area. The exchange below is a client determining if the server is an authority for abc.net or no.net.

```
S wait for connection
C connect to rs.internic.net port 4343
S %RWhois V-1.0 rs.internic.net (Network Solutions, Inc. V-1.0)
C -RWhois V-1.0 (Network Solutions, Inc. V-1.0)
S %ok
C -cache off
S %ok
C -soa abc.net
S %error 333 Not SOA for requested authority area
S %ok
C -soa no.net
S %soa authority: no.net
S %soa ttl: 7500
S %soa serial: 45
S %soa refresh: 3600
S %soa retry: 3600
S %soa tech-contact: markk@no.net
S %soa admin-contact: stanb@no.net
S %soa hostmaster: hostmaster@no.net
S %ok
```

3.7 Secondary Server Interaction

A server that operates as a secondary will report an authoritative SOA for the authority area of the data it contains. Below is the interaction between the primary and secondary server. In reality the secondary operation would be performed using a client specifically designed for this purpose.

```
S wait for connection
C connect to slam.internic.net port 4343
S %RWhois V-1.0 slam.internic.net (Network Solutions Inc. V-1.0)
C -RWhois V-1.0 (Network Solutions Inc. V-1.0)
S %ok
C -soa internic.net
S %soa authority: internic.net
S %soa ttl: 7500
S %soa serial: 45
S %soa refresh: 3600
S %soa retry: 3600
S %soa tech-contact: markk@internic.net
S %soa admin-contact: stanb@internic.net
S %soa hostmaster: hostmaster@rs.internic.net
S %ok
C -xfer domain internic.net
S ... all data for domain object in the internic.net authority
area transferred
S%ok
C -notify inssec netman1.netsol.com:4343:domain:internic.net
S %ok
C -quit
S close connection
C close connection
```

3.8 Registration Process

The following is the interaction that occurs when a server accepts a registration from a client.

```
S wait for connection
C connect to slam.internic.net port 4343
S %RWhois V-1.0 slam.internic.net (Network Solutions Inc. V-1.0)
C -RWhois V-1.0 (Network Solutions Inc. V-1.0)
S %ok
C -soa internic.net
S %soa authority: internic.net
S %soa ttl: 7500
S %soa serial: 45
S %soa refresh: 3600
S %soa retry: 3600
S %soa tech-contact: markk@internic.net
S %soa admin-contact: stanb@internic.net
S %soa hostmaster: hostmaster@rs.internic.net
S %ok
C -private auth password _98uuuts_
S %ok
C -register on add scottw@netsol.com _98uuuts_
S %ok
C ... send all attributes for object to register
S %error 120 Registration not processed... will process hours:24
C %quit
```

3.9 Out-of-Service

Servers that are being taken out of service should automatically refer the client back into the tree. Of course, this is not possible if the system which hosts the server is out of service. In this case, the client's robustness must be relied upon to return to the referrer and notify that server that the referral was bad. If the system will still be available on the Internet, the following exchange is recommended:

```
S wait for connection
C connect to slam.internic.net port 3636
S %RWhois V-1.0 slam.internic.net (Network Solutions Inc. V-1.0)
C -RWhois V-1.0 (Network Solutions Inc. V-1.0)
S %info on
S This server will no longer be in service. You should
S change your configuration file to reflect the new root
S server at rs.internic.net:4343. You will automatically be
S referred to the new root.
S %error 200 Service not available referral to follow
S %referral rs.internic.net:4343
S close connection
C close connection
```

4.0 Interaction: Client Directives and Acceptable Server Responses

This section describes the responses to the various client directives.

4.1 General

The responses below are general responses that can occur as a result of any directive. Therefore, they will not be repeated under each directive.

```
%ok
%error<SP>400<SP>Invalid Server Directive
%error<SP>100<SP>Get Peer Name query failed
%error<SP>500<SP>Memory Allocation Problem
%error<SP>401<SP>Not authorized for directive
%error<SP>402<SP>Unidentified error... continue
%error<SP>502<SP>Unrecoverable error... goodbye
%error<SP>503<SP>Idle time exceeded... goodbye
```

4.2 On Connection

These responses will only occur following successful connection to the server's host and start-up of the application:

```
%RWhois
%error<SP>501<SP>Service not available
%referral
%error<SP>503<SP>Idle time exceeded... goodbye
```

4.3 <QUERY>

These responses may occur following a query:

```
<answer>
%referral
%see-also
%error<SP>334<SP>Pre-query directive not implemented
%error<SP>230<SP>No Records Found
%error<SP>130<SP>Not authority for answer... TTL good
%error<SP>231<SP>Not authority for answer... TTL expired
```

4.4 -RWhois

```
%error<SP>300<SP>Not compatible with that version number
```

4.5 -load

```
%load
%error<SP>335<SP>System's load not available
```

4.6 -limit<SP>< value >

```
%limit
%error<SP>330<SP>Exceeded Max Records Limit
%error<SP>331<SP>Invalid Max Records Size
```

4.7 -schema<SP>[object]

```
%schema
%error<SP>337<SP>Object's schema not found
```

4.8 -xfer<SP>[object]

```
%xfer
%error<SP>332<SP>Nothing to transfer
%error<SP>337<SP>Object's schema not found
```

4.9 -quit

```
%ok
```

4.10 -cache<SP><on/off>

```
%error<SP>232<SP>Cache disabled
```

4.11 -status

```
%status
```

4.12 -forward

```
%error<SP>431<SP>Not authorized to forward
%error<SP>433<SP>Bad reference on forward
```

4.13 -soa

```
%soa
%error<SP>333<SP>Not SOA for requested authority area
```

4.14 -notify

```
   %error<SP>434<SP>Referral does not exist on this server
   %error<SP>530<SP>Not authorized as secondary
```

4.15 -register

```
   %error<SP>120<SP>Registration not processed... will process
hours:<hours>
   %error<SP>320<SP>Invalid attribute line:<line number>
   %error<SP>321<SP>Invalid format line:<line number>
   %error<SP>322<SP>Required attribute missing name:<attribute
name>
   %error<SP>323<SP>Required related object missing name:<object
name>
   %error<SP>324<SP>Primary key not unique
   %error<SP>420<SP>Registration not authorized
   %error<SP>421<SP>Not authorized to change object:<object
name><SP>key:<key>
```

4.16 -holdconnect

4.17 -object

```
   %object
   %error<SP>336<SP>Object not defined
```

4.18 -define

```
   %define
   %error<SP>435<SP>Macro not defined
```

4.19 -X-

```
   %X-
   %error<SP>460<SP>Extended directive not recognized
   %error<SP>461<SP>Extended directive not authorized
```

4.20 -display

```
   %display
   %display<SP>436<SP>Display mode not allowed
```

4.21 -language

```
   %language<SP>437<SP>Language not supported
```

5.0 Error Codes

The error code immediately follows the *"%error"* response from the RWhois server. The definitions of the error codes are below. The error codes are descriptive so that the client can group the error messages in order 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 *"%error 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 command error, retry that command 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

5.1 Error Code List

Below is an ordered list of RWhois error codes. These codes may be extended with implementation specific codes. These extended codes will have a '6' in the second position of the code.

```
100 Get Peer Name query failed
120 Registration not processed... will process hours:<hours>
130 Not authority for answer... TTL good

200 Service not available... Referral to follow
230 No Records Found
231 Not authority for answer... TTL expired
232 Cache disabled
```

```
300 Not compatible with that version number
320 Invalid attribute line:<line number>
321 Invalid format line:<line number>
322 Required attribute missing name:<attribute name>
323 Required related object missing name:<object name>
324 Primary key not unique
330 Exceeded Max Records Limit
331 Invalid Max Records Size
332 Nothing to transfer
333 Not SOA for requested authority area
334 Pre-query directive not implemented
335 System's load not available
336 Object not defined
337 Object's schema not found

400 Invalid Server Directive
401 Not authorized for directive
402 Unidentified error... continue
420 Registration not authorized
421 Not authorized to change object:<object name><SP>key:<key>
431 Not authorized to forward
432 Not authorized to transfer
433 Bad reference on forward
434 Referral does not exist on this server
435 Macro not defined
436 Display mode not allowed
437 Language not supported
460 Extended directive not recognized
461 Extended directive not authorized

500 Memory Allocation Problem
501 Service not available
502 Unrecoverable error... goodbye
503 Idle time exceeded... goodbye
530 Not authorized as secondary
```

6.0 Attribute Format

The format for all attributes for objects in the RWhois server must be specified using a format specifier. This definition will allow the client software to interpret the received data correctly. The RWhois format specifier closely follows the ‘C’ language scanf syntax with macro extensions.

Format specifiers must follow this pattern:

```
%[alignment][length restriction]<type>[range restriction]
```

`[alignment]` `'-'` = left justified
`'.'` = right justified

[length restriction] <value> = number of bytes allowed
<value>B = number of bits allowed

<type> This is the only required part of the format specifier. Below are the allowed format type values. The length of these values are not specified. These restrictions will be on the left of the [length restriction].

```
%c Character
%s String
%d Integer
%x Hex Integer
%o Octal Integer
%f Float
%e Scientific
%M defined macro
```

[range restriction] The range restriction will limit the allowed values. This may specify a number, character, or string range.

Examples: %-3s["ON","OFF"] = Defines a string with 3 characters left aligned and limited to the strings ON or OFF.
%16Bd[0-50] = 16 bit integer between 0 and 50

%4.2f[0-2500.50] = Defines a floating point number limited to 4 digits before and 2 after the decimal with a value between 0 and 2500.50.

6.1 Format Specification Macros

Format specifications may be presented as macros. Format specification macros may be defined using the following format.

```
%M<macro name>=<format string or earlier macro>
```

The following macros are pre-defined in this RWhois specification:

Month/Day/Year formats:
```
%MM=[%-2d[0-12]]
%MD=[%-2d[0-31]]
%MY2=[%-2d[0-99]]
%MY4=[%-4d[0-9999]]
%MMs=[%-3s\
["JAN","FEB","MAR","APR","MAY","JUN","JUL","AUG","SEP",\
"OCT","NOV","DEC"]]
```

Date formats:
```
%Mdate1=[%MM/%MD/%MY2]
%Mdate2=[%MM/%MD/%MY4]
%Mdate3=[%MD-%MMs-%MY2]
%Mdate4=[%MD-%MMs-%MY4]
%Mdate5=[%MY4%MM%MD]
```

Hour/Minute/Second formats:
```
%MTH=[%-2d[0-24]]
%MTM=[%-2d[0-59]]
%MTS=[%-2d[0-59]]
```

Time formats:
```
%Mtime1=[%MTH:%MTM:%MTS]
%Mtime2=[%MTH%MTM%MTS]
```

Miscellaneous formats:
```
%Mserver=[%s:%16Bd]
%Mipnumber=[%8Bd.%8Bd.%8Bd.%8Bd]
%Memail=[%s@%s]
%Mserial=[%Mdate5%Mtime2]
%Mstype=[RWHOIS/WHOIS/WHOIS++/OTHER]
%Murl=[%s://%s]
```

Macro definitions may be obtained by sending the *"-define"* directive to the server. For client efficiency, definitions can be remembered. If the definition of a macro changes, the serial number of all schemas using that macro must change, allowing the client to reacquire the schema and format specifier macros.

7.0 Quick Query (RWhois using 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 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.

8.0 References

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

[RFC-1521] Borenstein, N., and N. Freed, "MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the

Format of Internet Message Bodies", RFC 1521, Bellcore, Innosoft, September 1993.

9.0 Security Considerations

Security issues are not discussed in this memo.

10.0 Authors' Addresses

Scott Williamson
505 Huntmar Park Dr.
Herndon, VA 22070

Phone: (703) 742-4820
email: scottw@internic.net

Mark Kosters
505 Huntmar Park Dr.
Herndon, VA 22070

Phone: (703) 742-4795
email: markk@internic.net