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 1
2.0 RWhois Client Model 2
2.1 Directives: Client to Server Interaction 3
2.2 Required Directives 3
2.2.1 <query> 3
2.2.2 RWhois 4
2.3 Optional Directives 4
2.3.1 load 4
2.3.2 limit 5
2.3.3 schema 5
2.3.4 xfer 5
2.3.5 quit 6
2.3.6 status 6
2.3.7 cache 6
2.3.8 holdconnect 7
2.3.9 forward 7
2.3.10 soa 8
2.3.11 notify 8
2.3.12 register 9
2.3.13 object 10
2.3.14 define 11
2.3.15 private 11

2.3.16 X- 12
2.3.17 directive 12
2.3.18 display 13
2.3.19 language 13
2.4 RWhois Client Model 14
3.0 RWhois Server Model 15
3.1 Output Display and Restriction Keywords 15
3.2 Responses: Server to Client Interaction 16
3.3 Required Responses 16
3.3.1 RWhois 16
3.3.2 referral 17
3.3.3 ok 18
3.3.4 error 18
3.4 Optional Responses 19
3.4.1 see-also 19
3.4.2 load 20
3.4.3 soa 20
3.4.4 status 22
3.4.5 xfer 23
3.4.6 schema 24
3.4.7 define 25
3.4.8 object 26
3.4.9 directive 26
3.4.10 info 27
3.4.11 display 27
3.4.12 X- 28
3.4.13 language 28
3.5 Query Reduction 29
3.6 Determining Authority 29
3.7 Secondary Server Interaction 30
3.8 Registration Process 31
3.9 Out-of-Service 31
4.0 Interaction: Client Directives and Acceptable Server Responses 32
4.1 General 32
4.2 On Connection 32
4.3 <QUERY> 32
4.4 -RWhois 32
4.5 -load 33
4.6 -limit<SP>< value > 33
4.7 -schema<SP>[object] 33
4.8 -xfer<SP>[object] 33
4.9 -quit 33
4.10 -cache<SP><on/off> 33
4.11 -status 33
4.12 -forward 33

4.13 -soa 33
4.14 -notify 34
4.15 -register 34
4.16 -holdconnect 34
4.17 -object 34
4.18 -define 34
4.19 -X- 34
4.20 -display 34
4.21 -language 34
5.0 Error Codes 35
5.1 Error Code List 35
6.0 Attribute Format 36
6.1 Format Specification Macros 37
7.0 Quick Query (RWhois using UDP) 38
8.0 References 38
9.0 Security Considerations 39
10.0 Authors' Addresses 39

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.