ASID Working Group
Internet-Draft
Expires: September 11, 1998
draft-ietf-asid-whoispp-02.txt
Replaces: RFC-1835

Patrik Faltstrom
Tele2
Leslie L. Daigle
Bunyip Information Systems Inc.
Sima Newell
Bunyip Information Systems Inc.
March 1998

Architecture of the Whois++ service

Status of this Memo

Abstract

This document describes Whois++, an extension to the trivial WHOIS service described in RFC 954 to permit WHOIS-like servers to make available more structured information to the Internet. We describe an extension to the simple WHOIS data model and query protocol and a companion extensible, distributed indexing service. A number of options have also been added such as the use of multiple languages and character sets, more advanced search expressions, structured data and a number of other useful features. An optional authentication mechanism for protecting all or part of the associated Whois++ information database from unauthorized access is also described.

Table of Contents

1. Part I - Whois++ Overview

1.1. Purpose and Motivation

The current NIC WHOIS service [HARR85] is used to provide a very limited directory service, serving information about a small number of Internet users registered with the DDN NIC. Over time the basic service has been expanded to serve additional information and similar services have also been set up on other hosts. Unfortunately, these additions and extensions have been done in an ad hoc and uncoordinated manner.

The basic WHOIS information model represents each individual record as a Rolodex-like collection of text. Each record has a unique identifier (or handle), but otherwise is assumed to have little structure. The current service allows users to issue searches for individual strings within individual records, as well as searches for individual record handles using a very simple query-response protocol.

Despite its utility, the current NIC WHOIS service cannot function as a general White Pages service for the entire Internet. Given the inability of a single server to offer guaranteed response or reliability, the huge volume of traffic that a full scale directory service will generate and the potentially huge number of users of such a service, such a trivial architecture is obviously unsuitable for the current Internet's needs for information services.

This document describes the architecture and protocol for Whois++, a simple, distributed and extensible information lookup service based upon a small set of extensions to the original WHOIS information model. These extensions allow the new service to address the community's needs for a simple directory service, yet the extensible architecture is expected to also allow it to find applications in a number of other information service areas.

Added features include an extension to the trivial WHOIS data model and query protocol and a companion extensible, distributed indexing service. A number of other options have also been added, like boolean operators, more powerful search constraints and search methods. In addition, the data has been structured to make both the client and server elements of the dialogue more stringent and easily parsed. An optional authentication mechanism for protecting all or parts of the associated Whois++ information database from unauthorized access is also briefly described.

The architecture of Whois++ allows distributed maintenance of the directory contents and the use of the Whois++ indexing service for locating additional Whois++ servers. Although a general overview of this service is included for completeness, the indexing extensions are described described separately in [WINDX].

It should be noted that Whois++ is not backward compatible with WHOIS.

1.2. The Whois++ Information Model

The Whois++ service is based on the use of information templates, which consist of ordered sets of data elements (or attribute-value pairs). It underlying recommendation is to use standardized templates where available.

It is intended that adding structured template types to a server and subsequently searching through information stored in templates of a specified type should be simple tasks. The creation and use of customized templates should also be possible with little effort, although their use is discouraged where appropriate standardized templates exist.

Registration and schema definitions are done on an attribute by attribute basis, so a client that receives a record parses the record structure one attribute at a time. Because of this system, the client does not need to know the structure of the whole record, only individual attributes. If the client sees an unknown attribute, it will skip that one and continue parsing the subsequent attributes. A server that defines schemas can therefore add its own unregistered attributes to a well-defined template type.

We also offer methods to allow the user to constrain searches to desired attributes or template types, in addition to the existing commands for specifying handles or simple strings.

It is expected that the minimalist approach we have taken will find applications where the high cost of configuring and operating traditional White Pages services can not currently be justified.

Note also that the architecture makes no assumptions about the search and retrieval mechanisms used within individual servers. Operators are free to use dedicated database formats, fast indexing software or even provide gateways to other directory services to store and retrieve information. The Whois++ server simply functions as a known front end, offering a simple data model and communicating through a well known port and query protocol. The format of both queries and replies has been structured to allow the use of client software for generating searches and displaying the results. At the same time, some effort has been made to keep responses legible (to some degree) by human users, both to ensure low entry cost and to ease debugging.

The actual implemention details of an individual Whois++ search engine are left to the imagination of the implementor. It is hoped that the simple, extensible approach taken will encourage experimentation and the development of improved search engines.

1.2.1. Changes to the current WHOIS Model

The current WHOIS service is based upon an extremely simple data model. The NIC WHOIS database consists of a series of individual records, each of which is identified by a single unique identifer (the "handle"). Each record contains one or more lines of information. Currently, there is no structure or implicit ordering of this information, although each record is implicitly concerned with information about a single user or service.

We have implemented two basic changes to this model. First, we have structured the information within the database as collections of data elements that are simple attribute/value pairs. Each individual record contains a specified ordered set of these data elements.

Second, we have introduced the classing of database records into template types. In effect, each record is based upon one template of a specified set; each template contains a finite and specified number of data elements. This classing allows users to limit searches to specific collections of information, such as information about users, services, abstracts of papers, or descriptions of software.

Since the data typing is done at the attribute level, not the template level, it is also possible to add non-standard attributes to a well-known template type.

As an addition to the model, we require that each individual Whois++ database on the Internet be assigned a unique handle, analogous to the handle associated with each database record.

The Whois++ database structure is shown in Fig. 1.

```
 ______________________________________________________________________
|                                                                      |
|   +  Single unique Whois++ server   handle                           |
|                                                                      |
|              _______                 _______                _______  |
|    handle3  |..  .. |      handle6  |..  .. |     handle9  |..  .. | |
|            _______  |              _______  |             _______  | |
|  handle2  |..  .. |      handle5  |..  .. |     handle8  |..  .. |   |
|           _______ |               _______ |              _______ |   |
| handle1  |..  .. |      handle4  |..  .. |     handle7  |..  .. |    |
|          |..  .. |               |..  .. |              |..  .. |    |
|           -------                 -------                -------     |
|      Template                   Template               Template      |
|       Type 1                     Type 2                 Type 3       |
|                                                                      |
|                                                                      |
|                                                                      |
|                                                                      |
|               Fig.1 - Structure of a Whois++ database.          |
|                                                                  |
| Notes: - Entire database is identified by a single unique Whois++ |
|          serverhandle.                                           |
|        - Each record has a single unique handle.                 |
|        - Each record has a specific set of attributes, which is  |
|          determined by the Template Type used.                   |
|        - Each value associated with an attribute is a text string |
|          of an arbitrary length.                                 |
|__________________________________________________________________|
```

1.2.2. Registering Whois++ servers

We propose that individual database handles be registered through the Internet Assigned Numbers Authority (the IANA), ensuring their uniqueness. This will allow us to specify each Whois++ entry on the Internet as a unique pair consisting of a server handle and a record handle.

A unique registered handle is preferable to using the host's IP address, since it is conceivable that the Whois++ server for a particular domain may move over time. If we preserve the unique Whois++ handle in such cases we have the option of using it for resource discovery and networked information retrieval (see [IIIR] for a discussion of resource and discovery and support issues).

Uniqueness of server handles can be guaranteed by registering them with IANA.

We believe that organizing information around a series of such templates will make it easier for administrators to gather and maintain this information and thus encourage them to make such information available. At the same time, as users become more familiar with the data elements available within specific templates they will be able to specify their searches better, and the service will become more useful.

1.2.3. The Whois++ Search Selection Mechanism

The WHOIS++ search mechanism is intended to be extremely simple. A search command comprises one required element and one optional element. The first (required) element is a set of one or more search terms. The second (optional) element is a colon followed by set of one or more global constraints, which modify or control the search.

Within each search term, the user may specify the template type, attribute, value or handle that any record returned must satisfy. Each search term can have an optional set of local constraints that apply only to that term.

A Whois++ database may be seen as a single collection of typed records. Each search term specifies a further constraint that the selected set of output records must satisfy. Each term may thus be thought of as performing a subtractive selection, in the sense that any record that does not fulfill the term is discarded from the result set. Result sets can be further specified by supplying multiple search terms, related by logical connectives (AND, OR, NOT).

1.2.4. The Whois++ Architecture

The Whois++ directory service has an architecture which is separated into two components: the base level server, which is described in this paper, and an indexing server (described in [WINDX]). A single physical server can act as both a base level server and an indexing server.

A base level server is one which contains only filled templates. An indexing server is one which contains forward knowledge (q.v.) and pointers to other indexing servers or base level servers.

1.3. Indexing in Whois++

Indexing in Whois++ is used to tie together many base level servers and index servers into a unified directory service. For more detailed information on this subject, see [WINDX].

Each base level server and index server that is to participate in the unified directory service must generate forward knowledge for the entries it contains. One type of forward knowledge is the "centroid".

An example of a centroid is as follows. Consider a Whois++ server that contains exactly three records:

```
     Record 1                        Record 2
     Template: Person                Template: Person
     First-Name: John                First-Name: Joe
     Last-Name: Smith                Last-Name: Smith
     Favourite-Drink: Labatt Beer    Favourite-Drink: Molson Beer

     Record 3
     Template: Domain
     Domain-Name: foo.edu
     Contact-Name: Mike Foobar
```

the centroid for this server would be

```
     Template:       Person
     First-Name:     Joe
                     John
```

```
        Last-Name:      Smith
        Favourite-Drink:Beer
                        Labatt
                        Molson
        Template:       Domain
        Domain-Name:    foo.edu
        Contact-Name:   Mike
                        Foobar
```

An index server would then collect this centroid for this server as forward knowledge.

Index servers can collect forward knowledge for any servers it polls. In effect, all of the servers that the index server knows about can be searched with a single query to the index server; the index server holds the forward knowledge along with pointers to the servers it indexes, and can refer the query to servers which might hold information which satisfies the query.

Implementors of this protocol are strongly encouraged to incorporate centroid generation abilities into their servers.

Whois++ uses the Common Indexing Protocol, which was originally described in [WINDX] as a centroid-like object to provide index information (forward knowledge) about server contents. This work is being extended in the IETF's FIND Working-Group.

```
--------------------------------------------------------------------

                              ----             ----
top level                    |    |           |    |
whois index                  |    |           |    |
servers                       ----             ----
                             /    \_________    /
                            /              \   /
                        ____                ____
first level            |    |              |    |
whois index            |    |              |    |
servers                 ----                ----
                       /                   /    \
                      /                   /      \
                    ____                ____      ____
individual         |    |              |    |    |    |
whois servers      |    |              |    |    |    |
                    ----                ----      ----


                 Fig. 2 - Indexing system architecture.

--------------------------------------------------------------------
```

1.4.   **Getting Help**

Another extension to the basic WHOIS service is the requirement that all servers support at least a minimal set of help commands, allowing users to find out information about both the individual server and the entire Whois++ service itself. This is done in the context of the new extended information model by defining two specific template formats and requiring each server to offer at least one example of each record using these formats. The operator of each Whois++ service is therefor expected to have, as a minimum, a single example of SERVICES and HELP records, which can be accessed through appropriate commands.

1.4.1. Minimum HELP Required

Executing the command:

```
        DESCRIBE
```

gives a brief information about the Whois++ server.

Executing the command:

```
        HELP
```

gives a brief description of the Whois++ service itself.

The text of both required helped records should contain pointers to additional help subjects that are available.

Executing the command:

```
        HELP <searchstring>
```

gives information on <searchstring>.

1.5. Options and Constraints

The Whois++ service is based upon a minimal core set of commands and controlling constraints. A small set of additional optional commands and constraints can be supported by a server. These allow users to perform such tasks as provide security options, modify the information contents of a server or add multilingual support. The required set of Whois++ commands are listed in section 2.2. Whois++ constraints are described in section 2.3. Optional constraints are described in section 2.3.2.

1.6. Formatting Responses

The output returned by a Whois++ server is structured to allow machine parsing and automated handling. Of particular interest is the ability to return summary information about a search instead of having to return the entire results.

All output of searches will be returned in one of five output formats, which will be one of FULL, ABRIDGED, HANDLE, SUMMARY or SERVER-TO-ASK. Note that a conforming server is only required to support the FULL format.

When available, SERVER-TO-ASK format is used to indicate that a search cannot be completed but that one or more alternative Whois++ servers may be able to perform the search.

Details of each output format are specified in section 2.4.

1.7. Reporting Warnings and Errors

The formatted response of Whois++ commands allows the encoding of warning or error messages to simplify parsing and machine handling. The syntax of output formats are described in detail in section 2.4, and details of Whois++ warnings and error conditions are given in Appendix E.

All system messages are numerical, but can be tagged with text. It is the client's decision if the text is presented to the user.

1.8. Privacy and Security Issues

The basic Whois++ service was conceived as a simple, unauthenticated information lookup service, but there are occasions when authentication mechanisms are required. To handle such cases, one optional mechanism is provided for authenticating each Whois++ transaction. This is the ability to name a (mutually-recognized) authentication scheme in the optional AUTHENTICATE global constraint.

Note that the Whois++ authentication mechanism does not dictate the actual authentication scheme used, it merely provides a framework for indicating that a particular transaction is to be authenticated, and the appropriate scheme to use. This mechanism is extensible and individual implementors are free to add additional schemes.

Sophisticated security and authentication schemes may be proposed to address specific needs. For example, the Simple Authentication and Security Layer (SASL) work proposed by John Myers (particularly for POP and IMAP) may be applicable here.

2. Part II - Whois++ Implementation

2.1. The Whois++ interaction model

The Whois++ service has an assigned port number -- number 63. However, there is nothing inherent the Whois++ protocol or interaction model that prevents it from being used on any TCP connection on any port -- the specification of the connection is outside the scope of this protocol spec. Once a connection is established, the server issues a banner message, and listens for

input. The command specified in this input is processed and the
results returned including an ending system message. If the client
does not specify the optional HOLD constraint, the connection is
then terminated.

If the server supports the optional HOLD constraint, and this constraint is specified as part of any command, the server continues to listen on the connection for another (single) line of input. This cycle continues as long as the sender continues to append the required HOLD constraint to each subsequent command.

2.2. The Whois++ Command set

The Whois++ command set consists of a core set of required systems commands, a single required search command and an set of optional system commands which support features that are not required by all servers. The set of required Whois++ system commands are listed in Table I. Valid search terms for the search command are described in Table II.

Each Whois++ command also allows the use of one or more controlling constraints, which, when selected, are used to override defaults or otherwise modify the server's behavior. There is a core set of constraints that must be supported by all conforming servers: SEARCH (which controls the type of search performed), FORMAT (which determines the output format used) and MAXHITS (which determines the maximum number of matches that a search can return). These required constraints are summarized in Table III.

An additional set of optional constraints are used to provide support for different character sets, provide data for the authentication scheme, and requesting multiple transactions during a single communications session. These optional constraints are listed in Table IV.

It is possible, using the required COMMANDS and CONSTRAINTS system commands, to query any Whois++ server for its list of supported commands and constraints.

Please note that the line terminator is defined as a carriage return and line feed (CR/LF) pair. Also, none of the commands or constraints supported by Whois++ are case sensitive. For example, the following are equivalent: HELP, Help, help, hElp. Capitalization of all letters (e.g. HELP) is used only to improve the legibility of this document. Finally, "attribute value" is defined as "the value associated with an attribute".

2.2.1. System Commands

System commands are commands to the server for information or to control its operation. These include commands to list the template types available from individual servers, to obtain a single blank template of any available type, and commands to obtain the list of valid commands and constraints supported on a server.

There are also commands to obtain the current version of the Whois++ protocol supported, to access a simple help subsystem, to obtain a brief description of the service provided by the Whois++

server. The DESCRIBE command is intended, among other things, to support the automated registration of the service in yellow pages directory services. The required commands are listed in Table I.

Short	Long Form		Functionality
	COMMANDS	[':' HOLD]	List Whois++ commands supported by this server
	CONSTRAINTS	[':' HOLD]	List valid constraints supported by this server
	DESCRIBE	[':' HOLD]	Describe this server, formating the response using a standard SERVICES template
'?'	HELP [<string>	[':' (<othercnstrnts> / HOLD) 0*(';' (<otherconstraints> / HOLD))]]	Provide help specific to this Whois++ server, using a "Help" template
	LIST	[':' (<othercnstrnts> / HOLD) 0*(';' (<otherconstraints> / HOLD))]	List templates supported by this server
	POLLED-BY	[':' HOLD]	List indexing servers that are known to poll this server
	POLLED-FOR	[':' HOLD]	List information about servers this server polls
	SHOW <string>	[':' <cnstrnts>]	Show contents of template specified in <string>
	VERSION	[':' HOLD]	Show the version of the protocol supported by this server

Table I - Required Whois++ SYSTEM commands.

Descriptions of each command follow. Examples of responses to each command are provided in Appendix C.

2.2.1.1. The COMMANDS command

The COMMANDS command returns a list of commands that the server supports. The response is formatted as a FULL response.

2.2.1.2. The CONSTRAINTS command

The CONSTRAINTS command returns a list of both the constraints and their values that the server supports. The response is formatted as a FULL response, where every constraint is represented as a separate record. The template name for these records is CONSTRAINT. No attention is paid to handles. Each record has, as a minimum, the following two attributes:

- "Constraint", whose value is the constraint name
- "Default", which shows the default value for this constraint.

If the client is permitted to change the value of the constraint, there is also:

- "Range", which contains a list of values that this server supports, as a comma separated list, or, if the range is numerical, as a pair of numbers separated with a hyphen.

Note that, irrespective of whether a session is continued (with the HOLD constraint) or not, constraints are set to the default value unless explicitly changed with a constraint in each query.

2.2.1.3. The DESCRIBE command

The DESCRIBE command gives a brief description about the server in a "Services" template. The result is formatted as a FULL response with as a minimum one attribute:

- "Text", which describes the service in a form legible by human users.

2.2.1.4. The HELP command

The HELP command takes an optional argument which is the subject on which to get help. The answer is formatted as a FULL format response.

2.2.1.5. The LIST command

The LIST command returns the name of the templates available on the server. The answer is formatted as a FULL format response.

2.2.1.6. The POLLED-BY command

The POLLED-BY command returns a list of servers and the templates and attribute names that those servers polled as centroids from this server. The format is in FULL format with two attributes, "Template" and "Field", whose values are lists of the names of the polled templates and fields, respectively. An empty result means either that the server is not polled by anyone, or that it doesn't support indexing.

2.2.1.7. The POLLED-FOR command

The POLLED-FOR command returns a list of servers that this server has polled, and the template and attribute names for each of those. The answer is in FULL format with two attributes, Template and Field. An empty result means either that the server is not polling anyone, or that it doesn't support indexing.

2.2.1.8. The SHOW command

The SHOW command takes a template name as argument and returns information about that template, formatted as a FULL response. The answer is formatted as a blank template with the requested name.

2.2.1.9. The VERSION command

The output format is a FULL response containg a record with template name VERSION. The record must have attribute name "Version", whose value is "2.0" for this version of the protocol. The record may also have the additional fields "Program-Name" and "Program-Version" which gives information about the server implementation if the server so desires.

If the server also supports the earlier version of the protocol, "1.0", two records are given back as a response to the VERSION command, one for each version supported.

2.2.2. The SEARCH Command

A SEARCH command comprises one required element and one optional element. The first (required) element is a set of one or more search terms. The second (optional) element is a set of global constraints, which modify or control the search.

Each attribute value in the Whois++ database is divided into one or more words separated by whitespace:

```
whitespace = 1*( %d32 / %d09 / %d10 / %d13 / %d64 )
              ;  space  tab     LF     CR      @
```

Each search term operates on every word in the attribute value. Two or more search terms have to be combined with boolean operators AND, OR or NOT. The operator AND has higher precedence than the operator OR, but this can be changed by the use of parentheses.

Boolean operators function as follows for two search terms, A and B. Let A1 be the result set from the first search term and B1 be the result set from the second search. The operation A AND B returns the hits in the intersection of sets A1 and B1. The operation A OR B returns the hits in the union of the sets A1 and B1. The operation NOT A returns all possible results that are not in set A1. The behaviour of the boolean operators can be generalized to N search terms where N > 2. Note that NOT has a higher precedence than AND or OR, so NOT A AND B returns the hits in B that are not in A.

Search constraints that apply to all search terms are specified as global constraints. The search terms and the global constraints are separated with a colon (':'). Each additional global constraint is appended to the end of the search command, and a semicolon ';' is used as the delimiter between global constraints.

If any of the search constraints can not be fulfilled, or if several of the specified constraints are mutually exclusive, the server ignores the constraints that can not be fulfilled and those that are mutually exclusive. The server performs the search using only the remaining constraints and returns the corresponding set of records.

The set of required constraints are listed in Table III. The set of optional constraints are listed in Table IV.

As an option, the server may accept specifications for attributes to be included or excluded from a reply. Thus, users could specify -only- those attributes to return, or specific attributes to filter out, thus creating custom views.

2.2.2.1. Format of a Search Term

Each search term consists of one of the following:

1) A search string

   <value>

2) A search term specifier (as listed in Table II), followed by a '=', followed by a search string. This is noted as:

   <specifier> = <value>

3) An attribute name, followed by '=', followed by a search string:

   <attribute_name> = <value>

If no search term specifier is provided, then the search will be applied to attribute values only. This corresponds to an identifier of VALUE.

When the user specifies the search term using the form:

```
          "<attribute_name> = <value>"
```

this is considered to be an ATTRIBUTE-VALUE search.

For discussion of the system reply format, and selecting the
appropriate reply format, see section 2.4.

Valid specifiers:

Name	Functionality
HANDLE	Confine search to handles.
VALUE	Confine search to attribute values.

Table II - Valid search command term specifiers.

2.2.2.2. Format of a Search String

Special characters that need to be quoted are preceeded by a backslash, ''.

Special characters are space ' ', tab, equal sign '=', comma ',', colon ':', backslash '', semicolon ';', asterisk '*', period '.', parenthesis '()', square brackets '[]', dollar sign '$' and circumflex '^'.

If the search term is given in some other character set than ISO-8859-1, it must be specified by the constraint INCHARSET.

2.3. Whois++ Constraints

Constraints are intended to be hints or recommendations to the server about how to process a command. They may also be used to override default behaviour, such as requesting that a server not drop the connection after performing a command.

Thus, a user might specify a search constraint as "SEARCH=exact", which means that the search engine is to perform an exact match search. The user might also specify "LANGUAGE=Fr", which means that the server should (if possible) display the French versions of the attribute values, and if possible use French in fuzzy matches. The server should also issue system messages in French.

In general, constraints take the form "<constraintname>=<value>", where <value> is one of a specified set of valid values. The notable exception is "HOLD", which takes no argument.

The CONSTRAINTS system command is used to list the search constraints supported by an individual server.

If a server cannot satisfy the specified constraint, the server should indicate this to the user through the use of system messages. In such cases, the search is still performed, with the the server ignoring unsupported constraints.

2.3.1. Required Constraints

The following CONSTRAINTS must be supported in all conforming Whois++ servers.

```
  -----------------------------------------------------------------

   Format
   ------

  SEARCH=   exact / lstring

  FORMAT=   full / abridged / handle / summary

  MAXHITS=  1-<max-allowed>

  Table III - Required Whois++ constraints.

  -----------------------------------------------------------------
```

2.3.2. Optional CONSTRAINTS

The following CONSTRAINTS and constraint values are not required of a conforming Whois++ server, but may be supported. If supported, their names and supported values must be returned in the response to the CONSTRAINTS command.

```
---------------------------------------------------------------------

 Format
 ------

SEARCH=       regex / fuzzy / substring

CASE=         ignore / consider

FORMAT=       server-to-ask

MAXFULL=      1-<max-allowed>

AUTHENTICATE= data

INCHARSET=    us-ascii / iso-8859-* /
                UNICODE-1-1-UTF-8 / UNICODE-2-0-UTF-8 / UTF-8

OUTCHARSET=   us-ascii / iso-8859-* /
                UNICODE-1-1-UTF-8 / UNICODE-2-0-UTF-8 / UTF-8

LANGUAGE=     <As defined in RFC 1766 [ALVE95]>
```

```
   HOLD

   IGNORE=       <attributelist>
```

INCLUDE=	<attributelist>

N.B.: "UTF-8" is as defined in [RFC2279]. This is the character set label that should be used for UTF encoded information; the labels "UNICODE-2-0-UTF-8" and "UNICODE-1-1-UTF-8" are retained primarily for compatibility with older Whois++ servers, and as outlined in [RFC2279].

Table IV - Optional Whois++ constraints.

2.3.2.1. The SEARCH Constraint

The SEARCH constraint is used for specifying the method that is to be used for the search. The default method is "exact". Following is a definition of each search method.

exact	The search will succeed for a word that exactly matches the search string.
substring	The search will succeed for a word that matches a part of a word.
regex	The search will succeed for a word when a regular expression matches the searched data. Regular expression is built up by using constructions of '*', '.', '^', '$', and '[]'. For use of regular expressions see Appendix H.
fuzzy	The search will succeed for words that matches the search string by using an algorithm designed to catch closely related names with different spelling, e.g. names with the same pronunciation. The server chooses which algorithm to use, but it may vary depending on template name, attribute name and language used (see Constraint Language above).
lstring	The search will succeed for words that begins with the search string.

2.3.2.2. The FORMAT Constraint

The FORMAT constraint describes what format the result will be in. Default format is FULL. For a description of each format, see Server Response Modes below.

2.3.2.3. The MAXFULL Constraint

The MAXFULL constraint sets the limit of the number of matching records the server allows before it enforces SUMMARY responses. The client may attempt to override this value by specifying another value to that constraint. Example: If, for privacy reasons, the server is to return the response in SUMMARY format if the number of hits exceeds 2, the MAXFULL constraint is set to 2 by the server.

Regardless of what format the client asked for, the server will change the response format to SUMMARY when the number of matching records equals or exceeds this value.

2.3.2.4. The MAXHITS Constraint

The MAXHITS constraint sets the maximum number of records returned to the client in response to a query.

2.3.2.5. The CASE Constraint

The CASE constraint defines if the search should be case sensitive or not. Default value is to have case ignored.

2.3.2.6. The AUTHENTICATE Constraint

The AUTHENTICATE constraint describes which authentication scheme to use when executing the search. Depending on the authentication scheme used, some other constraints may have to be specified. The authentication scheme definition identifies which constraints it requires.

2.3.2.7. The LANGUAGE Constraint

The LANGUAGE constraint specifies the language in which the client wishes to receive responses. It therefore specifies which attribute values should be presented to the user (i.e., only those in the specified language, or for which no language information is available). It can also be used as an extra information to the fuzzy matching search method, and it might also be used to tell the server to give the system responses in another language. This should preferably be handled by the client. The language codes defined in RFC 1766 [ALVE95] should be used as a value for the language constraint. In these, the case of the letters are insignificant.

If a record has attribute values in different languages, and no LANGUAGE search constraint was given in the query, the switch between the different languages should be given in the response by the use of system messages 601 which has one argument only, the name of the language or one of the predefined strings "ANY" or "DEF". A block of alternative attribute values starts with a language definition like "% 601 SE". After the first language specification, zero or more language specifications can be given, each switching into the desired language. When all specific languages have been tagged, the specification "% 601 DEF" can be used for specifying default attribute values. A block of alternative attributes must end with "% 601 ANY".

The following is an example of a response using the language messages:

```
# FULL USER LOCAL USER-DOE
% 601 FR
 Name: Monsieur John Doe
% 601 SV
 Name: Herr John Doe
% 601 DEF
 Name: Mister John Doe
% 601 ANY
 Email: jdoe@doe.pp.se
# END
```

The language specifications may be suppressed by the server (using the % 601 messages) if the client has explicitly, by using the global constraint LANGUAGE, asked for a specific language.

2.3.2.8. The INCHARSET Constraint

The INCHARSET constraint tells the server in which character set the search string itself is given. The default character set is ISO-8859-1.

2.3.2.9. The OUTCHARSET Constraint

The OUTCHARSET constraint tells the server in which character set the search result data (not attributenames or system information) is supposed to be given in. The default character set is ISO-8859-1, but the server may choose something else.

2.3.2.10. The IGNORE Constraint

The IGNORE constraint specifies which attributes NOT to include in the result. All other attributes will be included (as if named explicitly by the "include" constraint).

If an attribute is named both with the "include" and "ignore" constraint, the attribute is to be included in the result, but the system message "% 112 Requested constraint not fulfilled" must be sent.

2.3.2.11. The INCLUDE Constraint

The INCLUDE constraint specifies which attributes to include in the result. All other attributes will be excluded (as if named explicitly by the "ignore" constraint).

If an attribute is named both with the "include" and "ignore" constraint, the attribute is to be included in the result, but the system message must be "% 112 Requested constraint not fulfilled".

2.3.2.12. The HOLD Constraint

The HOLD constraint requests that the server hold open the connection after sending the response to the query. The server waits for another user input string.

2.4. Server Response Modes

The grammar for Whois++ responses is given in Appendix G, and described below.

There are currently a total of five different response modes possible for Whois++ servers. These are FULL, ABRIDGED, HANDLE, SUMMARY and SERVER-TO-ASK. The syntax of each output format is specified in more detail in Appendix G.

1) A FULL format response provides the complete contents of a template matching the specified query, including the template type, the server handle and an optional record handle.

2) An ABRIDGED format response provides a brief summary, including (as a minimum) the server handle, the corresponding record handle and relevant information for that template.

3) A HANDLE format response returns a line with information about the server handle and record handle for a record that matched the specified query.

4) A SUMMARY response provides only a brief summary of information the number of matches and the list of template types in which the matches occurred.

5) A SERVER-TO-ASK response only returns pointers to other index servers which might possibly be able to answer the specified query.

The server may optionally respond with an empty result set and may also respond with an empty response together with a system message to indicate that the query was too complex for it to fulfill.

2.4.1. Default Responses

By default, a Whois++ server will provide FULL responses. This may be changed by the client with the use of the global constraint "format".

The server will not respond with more matches than the value specified with the global constraint "maxhits" in any response format. If the number of matches exceeds this value, the server will issues the system message 110 (maxhits value exceeded), but will still show the responses, up to the number of the "maxhits" constraint value. This mechanism will allow the server to hide the number of possible matches to a search command.

2.4.2. Format of Responses

Each response consists of a numerical system generated message, which can be tagged with text, followed by an optional formatted response message, followed by a second system generated message. The formatted response itself can include system messages, for example for switches in language.

That is:

```
     '%' <system messages> <CR/LF>

     [ <formatted response> ]

     '%' <system messages> <CR/LF>
```

If there are no matches to a query, the system is not required to generate any output as a formatted response, although it must still generate system messages.

For information about the standard text for system messages, see Appendix E.

2.4.3. Syntax of a Formatted Response

All formatted responses except for the HANDLE response, consist of a response-specific START line, followed by an optional response-specific data section, followed by a TERMINATION line. The HANDLE response is different in that it only consists of a START line. It is permissible to insert any number of lines consisting solely of CR/LF pairs within a formatted response to improve readability.

Each line shall be limited to no more than 81 characters, including the terminating CR/LF pair. If a line (including the required leading single space) would exceed 81 characters, it must be broken into lines of no more than 81 characters, with each continuation line beginning with a "+" character in the first column instead of the leading character.

If an attribute value in a data section includes a line break, the line break must be replaced by a CR/LF pair and the following line begin with a "-" character in the first column, instead of the leading character. The attribute name is not repeated on consecutive lines.

A TERMINATION line consists of a line with a '#' in the first column, followed by one space (ASCII 32) character, followed by the keyword END, followed by zero or more characters, followed by a CR/LF pair.

A response-specific section will be one of the following:

1) FULL Format Response
2) ABRIDGED Format Response
3) HANDLE Format Response
4) SUMMARY Format Response

5) SERVER-TO-ASK Format Response

2.4.3.1. A FULL format response

A FULL format response consists of a series of responses, each consisting of a START line, followed by the complete template information for the matching record and a TERMINATION line.

Each START line consists of a '#' in the first column, followed by one space character, the word "FULL", a space character, the name of the corresponding template type, one space character, the server handle, a space character, (optionally) the handle for the record, and a terminating CR/LF pair.

The template information for the record will be returned as a series of lines consisting of a single space, followed by the corresponding line of the record.

The line of the record shall consist of a single space and the attribute name followed by a ':', a single space, the value of that attribute, and a CR/LF pair.

2.4.3.2. ABRIDGED Format Response

Each ABRIDGED format response consists of a START line, a single line excerpt of the template information from each matching record and a TERMINATION line. The excerpt information shall include information that is relevant to the template type.

The START line consists of a '#' in the first column, followed by one space character, the word "ABRIDGED", a space character, the name of the corresponding template type, a space character, the server handle, a space character, the handle for the record, and a terminating CR/LF pair.

The abridged template information will be returned as a line, consisting of a single space, followed by the abridged line of the record and a CR/LF pair.

2.4.3.3. HANDLE Format Response

A HANDLE response consists of a single START line, which shall start with a '#' in the first column, followed by one space character, the word "HANDLE", a space character, the name of the corresponding template, a space character, the handle for the server, a space character, the handle for that record, and a terminating CR/LF pair.

2.4.3.4. SUMMARY Format Response

A SUMMARY format response consists of a single response, consisting of a line listing the number of matches to the specified query, optionally a count of referrals, followed by a list of all template types which satisfied the query at least once.

The START line shall begin with a '#' in the first column, be followed by one space character (decimal 32), the word "SUMMARY", a single space character, the handle for the server, and a terminating CR/LF pair.

The format of the attributes in the SUMMARY format follows the rules for the FULL template, with the attributes "matches", "referrals" and "templates". "matches" and "templates" are mandatory, "referrals" optional.

The first line must begin with the string "matches:", be followed by a space and the number of responses to the query and terminated by a CR/LF pair.

The following line shall either begin with the string "templates: " or the string "referrals: ". The string "templates: " are followed by a CR/LF separated list of the name of the template types which matched the query. Each line following the first which include the text "templates:" must begin with a '-' instead of a space. The string "referrals: " is followed by the number of referrals included in the number of hits.

2.4.3.5. SERVER-TO-ASK Response

A SERVER-TO-ASK response consists of information to the client about a server to contact next to resolve a query. If the server has pointers to more than one server, it will present additional SERVER-TO-ASK responses.

The SERVER-TO-ASK response will consist of a START line and a number of lines with attribute-value pairs, separated by CRLF. Each line is indented with one space. The end of a SERVER-TO-ASK response is indicated with a TERMINATION line.

Each START line consists of a '#' in the first column, followed by one space character, the word "SERVER-TO-ASK", a space character, the handle of the server and a terminating CR/LF pair.

1. "Server-Handle" - The server handle of the server pointed at. (req.)
2. "Host-Name" - Hostname for the server pointed at.
3. "Host-Port" - Portnumber for the server pointed at.
4. "Protocol" - The protocol to use when contacting this server. (opt.)

Other attributes may be present, depending on the index server.

The default protocol to use is Whois++.

2.4.4. System Generated Messages

All system generated messages must have a '%' as the first character, a space as the second one, followed by a three digit number, a space and an optional text message. The total length of the line must be no more than 81 characters long, including the terminating CR/LF pair. There is no limit to the number of system messages that may be generated.

The format for multiline replies requires that every line, except the last, begin with "%", followed by space, the reply code, a hyphen, and an optional text. The last line will begin with "%", followed by space, the reply code, a space and some optional text.

System generated messages displayed before or after the formatted response section are expected to refer to operation of the system or refer to the entire query. System generated messages within the output of an individual record during a FULL response are expected to refer to that record only, and could (for example) be used to indicate problems with that record of the response. See Appendix E for a description of system messages.

2.5. Compatibility with Older WHOIS Servers

Note that this format, although potentially more verbose, is still in a human readable form. Responses from older systems that do not follow this format are still conformant, since their responses would be interpreted as being equivalent to optional text messages, without a formatted response. Clients written to this specification would display the responses as a advisory text message, where it would still be readable by the user.

3. Miscellaneous

3.1. Acknowledgements

This document has been through many iterations of refinement, with contributions of different natures along the way. These acknowledgements accrue.

The Whois++ effort began as an intensive brainstorming session at the 24th IETF, in Boston Massachusetts. Present at the birth, and contributing ideas through this early phase, were (alphabetically) Peter Deutsch, Alan Emtage, Jim Fullton, Joan Gargano, Brad Passwaters, Simon Spero, and Chris Weider. Others who have since helped shape this document with feedback and suggestions include Roxana Bradescu, Patrik Faltstrom, Kevin Gamiel, Dan Kegel, Michael Mealling, Mark Prior and Rickard Schoultz.

Version 2 of the protocol spec is based on input during the lifetime of version 1. Special mention goes to Jeff Allen, Leslie Daigle, and Philippe Boucher. During the polishing of the RFC for version 2, important input was given by Len Charest, Clarke Anderson and others in the ASID working group of the IETF.

This work was supported in part by grant 12/39/01 from the UK Electronic Libraries Programme (eLib), an initiative of the

Joint Information Systems Committee (JISC). This grant has provided the opportunity to test the protocol specification by developing a test suite. The challenge was not only to provide AN implementation that satisfied the document, but to build tools that would be able to respond to all POSSIBLE responses that could be implemented from the spec. This lead to the contribution of some textual clarifications. Specific thanks go to Bill Heelan and Philippe Boucher.

3.2 References

[ALVE95] Alvestrand H., "Tags for the Identification of Languages", RFC 1766, UNINETT, March 1995.

[RFC2234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997.

[HARR85] Harrenstein K., Stahl M., and E. Feinler, "NICNAME/WHOIS", RFC 954, SRI, October 1985.

[POST82] Postel J., "Simple Mail Transfer Protocol", STD 10, RFC 821, USC/Information Sciences Institute, August 1982.

[IIIR] Weider C., and P. Deutsch, "A Vision of an Integrated Internet Information Service", RFC 1727 Bunyip Information Systems, Inc., December 1994.

[WINDX] Weider, C., J. Fullton, and S. Spero, "Architecture of the Whois++ Index Service", RFC 1913, February 1996.

[RFC2279] F. Yergeau, " UTF-8, a transformation format of ISO 10646", RFC 2279, January 1998.

3.3. Authors Addresses

Patrik Faltstrom
Tele2
Borgarfjordsgatan 16
BOX 62
194 64 Kista
SWEDEN

Email: paf@swip.net

Leslie L. Daigle
Bunyip Information Systems Inc.

310 Ste. Catherine St. W
Suite 300
Montreal, Quebec, CANADA
H2X 2A1

Email: leslie@bunyip.com

Sima Newell
Bunyip Information Systems Inc.
310 Ste. Catherine St. W
Suite 300
Montreal, Quebec, CANADA
H2X 2A1

Email: sima@bunyip.com

Appendix A - Some Sample Queries

```
    author=leslie and template=user
```

The result will consist of all records where attribute "author" matches "leslie" with case ignored. Only USER templates will be searched. An example of a matching attribute is "Author=Leslie L. Daigle".

This is the typical case of searching.

```
    author=leslie and template=user:language=fr
```

The result will consist of the same records as above, but if attributes are available in alternative languages, only the ones in French will be displayed. These are either the ones which have explicitly been tagged as having French values, or ones that are tagged as being in the "DEF" (default) language.

```
    schoultz and rick;search=lstring
```

The result will consist of all records which have one attribute value matching "schoultz" exactly (because the default search type is exact) and one attribute with "rick" as leading substring, both with case ignored. One example is "Name=Rickard Schoultz".

```
    value=phone;search=substring
```

The result will consist of all records which have attribute values matching *phone*, for example the record "Name=Acme telephone inc.", but will not match the attribute name "phone". (Since term specifier is "value" by default, the search term could just as well have been simply "phone").

```
   ucdavis;search=substring and (gargano or joan):include=name,email
```

This search command will find records which have records containing the words "gargano" or "joan" somewhere in the record, and has the word "ucdavis" somewhere in a word. The result will only show the "name" and "email" fields.

Appendix B - Some sample responses

1) FULL format responses:

```
# FULL USER SERVERHANDLE1 PD45
 Name: Peter Deutsch
 email: peterd@bunyip.com
# END
# FULL USER SERVERHANDLE1 AE1
 Name: Alan Emtage
 email: bajan@bunyip.com
# END
# FULL USER SERVERHANDLE1 NW1
 Name: Nick West
 Favourite-Bicycle-Forward-Wheel-Brand: New Bicy
+cles Acme Inc.
 email: nick@bicycle.acme.com
 My-favourite-song: Happy birthday to you!
-Happy birthday to you!
-Happy birthday dear Nick!
-Happy birthday to you.
# END
# FULL SERVICES SERVERHANDLE1 WWW1
 Type: World Wide Web
 Location: the world
# END
```

----

2) An ABRIDGED format response:

```
# ABRIDGED USER SERVERHANDLE1 PD45
 Peter Deutsch             peterd@bunyip.com
# END
# ABRIDGED USER SERVERHANDLE1 AE1
 Alan Emtage               bajan@bunyip.com
# END
# ABRIDGED USER SERVERHANDLE1 WWW1
 World Wide Web            the world
# END
```

----

3) HANDLE format responses:

```
# HANDLE USER SERVERHANDLE1 PD45
# HANDLE USER SERVERHANDLE1 AE1
# HANDLE SERVICES SERVERHANDLE1 WWW1
```

----

4) A SUMMARY format response:

```
# SUMMARY SERVERHANDLE1
 Matches: 35
 Referrals: 2
 Templates: User
-Services
-Abstracts
# END
```

Appendix C - Sample responses to system commands

C.1 Response to the LIST command

```
# FULL LIST SERVERHANDLE1
 Templates: USER
-SERVICES
-HELP
# END
```

C.2 Response to the SHOW command

This example shows the result after issuing "show user":

```
# FULL USER SERVERHANDLE1
  Name:
  Email:
  Work-Phone:
  Organization-Name:
  City:
  Country:
# END
```

C.3 Response to the POLLED-BY command

```
# FULL POLLED-BY SERVERHANDLE1
 Server-handle: serverhandle2
 Cached-Host-Name: sunic.sunet.se
 Cached-Host-Port: 7070
 Template: USER
 Field: ALL
```

```
      # END
      # FULL POLLED-BY SERVERHANDLE1
       Server-handle: serverhandle3
       Cached-Host-Name: kth.se
       Cached-Host-Port: 7070
```

```
    Template: ALL
    Field: Name,Email
   # END
```

C.4 Response to the POLLED-FOR command

```
   # FULL POLLED-FOR SERVERHANDLE1
    Server-Handle: serverhandle5
    Template: ALL
    Field: Name,Address,Job-Title,Organization-Name,
   +Organization-Address,Organization-Name
   # END
   # FULL POLLED-FOR SERVERHANDLE1
    Server-Handle: serverhandle4
    Template: USER
    Field: ALL
   # END
```

C.5 Response to the VERSION command

```
   # FULL VERSION BUNYIP.COM
    Version: 2.0
    Program-Name: Digger
    Program-Version: 3.0b1
    Program-Author: Bunyip Information Systems Inc.
    Program-Author-Email: digger-info@bunyip.com
    Bug-Report-Email: digger-bugs@bunyip.com
   # END
```

C.6 Response to the CONSTRAINTS command

```
   # FULL CONSTRAINTS SERVERHANDLE1
    CONSTRAINT: maxhits
    DEFAULT: 100
    RANGE: 0-100
   # END
   # FULL CONSTRAINTS SERVERHANDLE1
    CONSTRAINT: case
    DEFAULT: ignore
    RANGE: ignore, consider
   # END
   # FULL CONSTRAINTS SERVERHANDLE1
    CONSTRAINT: search
    DEFAULT: exact
    RANGE: exact, lstring, substring, fuzzy
   # END
```

```
      # FULL CONSTRAINTS SERVERHANDLE1
       CONSTRAINT: language
       DEFAULT: DEF
       RANGE: FR, EN, SV, ANY, DEF
      # END
```

```
   # FULL CONSTRAINTS SERVERHANDLE1
    CONSTRAINT: incharset
    DEFAULT: ISO-8859-1
    RANGE: ISO-8859-1, UTF-8
   # END
   # FULL CONSTRAINTS SERVERHANDLE1
    CONSTRAINT: outcharset
    DEFAULT: ISO-8859-1
    RANGE: ISO-8859-1, UTF-8, HTML
   # END
```

C.7 Response to the COMMANDS command

```
   # FULL COMMANDS SERVERHANDLE1
    Commands: commands
   -constraints
   -describe
   -help
   -list
   -polled-by
   -polled-for
   -show
   -version
   # END
```

Appendix D - Sample Whois++ session

Below is an example of a session between a client and a server. The angle brackets to the left is not part of the communication, but is just put there to denote the direction of the communication between the server or the client. Text appended to '>' means messages from the server and '<' from the client.

Client connects to the server

```
  >% 220-Welcome to
  >% 220-the Whois++ server
  >% 220 at ACME inc.
  <name=Nick:hold
  >% 200 Command okay
  >
  ># FULL USER ACME.COM NW1
  > name: Nick West
  > email: nick@acme.com
  ># END
```

```
   ># SERVER-TO-ASK ACME.COM
   > Server-Handle: SUNETSE01
   > Host-Name: whois.sunet.se
   > Host-Port: 7070
   ># END
```

```
># SERVER-TO-ASK ACME.COM
> Server-Handle: KTHSE01
># END
>% 226 Transfer complete
<version
>% 200 Command okay
># FULL VERSION ACME.COM
> Version: 2.0
># END
>% 226 Transfer complete
>% 203 Bye
Server closes the connection
```

In the example above, the client connected to a Whois++ server and queried for all records where the attribute "name" equals "Nick", and asked the server not to close the connection after the response by using the global constraint "HOLD".

The server responds with one record and a pointer to two other servers that either holds records or pointers to other servers.

The client continues with asking for the servers version number without using the HOLD constraint. After responding with protocol version, the server closes the connection.

Note that each response from the server begins system message 200 (Command OK), and ends with system message 226 (Transfer Complete).

Appendix E - System messages

A system message begins with a '%', followed by a space and a three digit number, a space, and an optional text message. The line message must be no more than 81 characters long, including the terminating CR LF pair. There is no limit to the number of system messages that may be generated.

A multiline system message have a hyphen instead of a space in column 6, immediately after the numeric response code in all lines, except the last one, where the space is used.

Example 1

```
% 200 Command okay
```

Example 2

```
% 220-Welcome to
% 220-the Whois++ server
% 220 at ACME inc.
```

The client is not expected to parse the text part of the response
message except when receiving reply 600 or 601, in which case the
text part is in the former case the name of a character set that
will be used by the server in the rest of the response, and in the

latter case when it specifies what language the attribute value is in. The valid values for characters sets is specified in the "characterset" list in the grammar in Appendix F.

The theory of reply codes is described in Appendix E in STD 10, RFC 821 [POST82].

List of system response codes
----

110 **Too many hits**	The number of matches exceeded the value specified by the maxhits constraint. Server will still reply with as many records as "maxhits" allows.
111 **Requested constraint not supported**	One or more constraints in query is not implemented, but the search is still done.
112 **Requested constraint not fulfilled**	One or more constraints in query has unacceptable value and was therefore not used, but the search is still done.
200 **Command Ok**	Command accepted (i.e., syntax okay, will be executed). The client must wait for a transaction end system message.
201 **Command Completed successfully**	Command accepted and executed.
203 **Bye**	Server is closing connection
220 **Service Ready**	Greeting message. Server is accepting commands.
226 **Transaction complete**	End of data. All responses to query are sent.
430 **Authentication needed**	Client requested information that needs authentication.
500 **Syntax error**	
502 **Search expression too complicated**	This message is sent when the

```
                                      server is not able to resolve
                                      a query (i.e. when a client
                                      sent a regular expression that
                                      is too deeply nested).
```

530 Authentication failed	The authentication phase failed.
600 <token>	Subsequent attribute values are encoded in the character set specified by <token>.
601 <token>	Subsequent attribute values are in the language specified by <token>.
601 DEF	Subsequent attribute values are default values, i.e. they should be used for all languages not specified by "601 <token>" since last "601 ANY" message.
601 ANY	Subsequent attribute values are for all languages.

Table V - System response codes

Appendix F - The Whois++ Input Grammar

The following grammar, which uses BNF-like notation as defined in [RFC2234] defines the set of acceptable input to a Whois++ server.

N.B.: As outlined in the ABNF definition, rule names and string literals are in the US-ASCII character set, and are case-insensitive.

```
   whois-command   =   ( system-command [":" "hold"]
                       / terms [":" globalcnstrnts] ) nl

   system-command  =   "constraints"
                       / "describe"
                       / "commands"
                       / "polled-by"
                       / "polled-for"
                       / "version"
                       / "list"
                       / "show" [1*sp bytestring]
                       / "help" [1*sp bytestring]
                       / "?" [bytestring]

   terms           =   and-expr *("or" and-expr)
   and-expr        =   not-expr *("and" not-expr)

   not-expr        =   ["not"] (term / ( "(" terms ")" ))

   term            =   generalterm / specificterm
```

```
                    / combinedterm

generalterm     =   bytestring

specificterm    =   specificname "=" bytestring

specificname    =   "handle" / "value"

combinedterm    =   attributename "=" bytestring

globalcnstrnts  =   globalcnstrnt *(";" globalcnstrnt)

globalcnstrnt   =   "format" "=" format
                    / "maxfull" "=" 1*digit
                    / "maxhits" "=" 1*digit
                    / opt-globalcnst

opt-globalcnst  =   "hold"
                    / "authenticate" "=" auth-method
                    / "language" "=" language
                    / "incharset" "=" characterset
                    / "ignore" "=" bytestring
                    / "include" "=" bytestring

format          =   "full" / "abridged" / "handle" / "summary"
                    / "server-to-ask"

auth-method     =  bytestring

language        = <The language code defined in RFC1766 [ALVE95]>

characterset    =   "us-ascii" / "iso-8859-1" / "iso-8859-2" /
                    "iso-8859-3" / "iso-8859-4" / "iso-8859-5" /
                    "iso-8859-6" / "iso-8859-7" / "iso-8859-8" /
                    "iso-8859-9" / "iso-8859-10" /
                    "UNICODE-1-1-UTF-8" / "UNICODE-2-0-UTF-8"
                    "UTF-8"

                      ;"UTF-8" is as defined in [RFC2279].  This is
                      ;the character set label that should be used
                      ;for UTF encoded information; the labels
                      ;"UNICODE-2-0-UTF-8" and "UNICODE-1-1-UTF-8"
                      ;are retained primarily for compatibility with
                      ;older Whois++ servers (and as outlined in
                      ;[RFC2279]).


searchvalue     =   "exact" / "substring" / "regex" / "fuzzy"
                    / "lstring"
```

```
   casevalue       =   "ignore" / "consider"

   bytestring          =   0*charbyte
   attributename   =   1*attrbyte

   charbyte        =   "

   normalbyte      =   <%d33-255, except specialbyte>

   attrbyte        =   <%d33-127 except specialbyte> /
                       "

   specialbyte     =   " " / tab / "=" / "," / ":" / ";" / "
                       "*" / "." / "(" / ")" / "[" / "]" / "^" /
                       "$" / "!" / "?"

   tab             =  %d09
   sp              =  %d32        ; space

   digit           =   "0" / "1" / "2" / "3" / "4" /
                       "5" / "6" / "7" / "8" / "9"

   nl              =   %d13 %d10   ; CR LF
```

NOTE: Blanks that are significant to a query must be escaped. The following characters, when significant to the query, may be preceded and/or followed by a single blank:

```
      : ; , ( ) = !
```

Appendix G - The Whois++ Response Grammar

The following grammar, which uses ABNF-like notation as defined in [RFC2234], defines the set of responses expected from a Whois++ server upon receipt of a valid Whois++ query.

N.B.: As outlined in the ABNF definition, rule names and string literals are in the US-ASCII character set, and are case-insensitive.

```
   server          =   goodmessage mnl output mnl endmessage nl
                     / badmessage nl endmessage nl

   output          =   full / abridged / summary / handle

   full            =   0*(full-record / server-to-ask)

   abridged        =   0*(abridged-record / server-to-ask)

   summary         =   summary-record
   handle          =   0*(handle-record / server-to-ask)

   full-record     =   "# FULL " template serverhandle localhandle
```

```
                              system-nl
                       1*(fulldata system-nl)
                       "# END" system-nl

abridged-record =   "# ABRIDGED " template serverhandle localhandle
                        system-nl
                    abridgeddata
                    "# END" system-nl

summary-record   =  "# SUMMARY " serverhandle system-nl
                    summarydata
                    "# END" system-nl

handle-record    =  "# HANDLE " template serverhandle localhandle
                        system-nl

server-to-ask   =   "# SERVER-TO-ASK " serverhandle system-nl
                    server-to-askdata
                    "# END" system-nl

fulldata        =   " " attributename ": " attributevalue

abridgeddata    =   " " 0*( attributevalue / tab )

summarydata     =   " Matches: " number system-nl
                    [" Referrals: " number system-nl]
                    " Templates: " template 0*( system-nl "-"
                                               template)

server-to-ask-data = " Server-Handle:" serverhandle system-nl
                     " Host-Name: " hostname system-nl
                     " Host-Port: " number system-nl
                     [" Protocol: " prot system-nl]
                     0*(" " labelstring ": " labelstring system-nl)


attributename   =   1*attrbyte

attrbyte        =   <%d33-127 except specialbyte>

attributevalue  =   longstring

template        =   labelstring

serverhandle    =   labelstring

localhandle     =   labelstring

hostname        =   labelstring
```

```
   prot            =   labelstring

   longstring      =   bytestring 0*( nl ( "+" / "-" ) bytestring )
```

```
bytestring      =   0*charbyte

labelstring     =   0*restrictedbyte

restrictedbyte  =   <%d32-%d255 except specialbyte>

charbyte        =   <%d32-%d255 except nl>

specialbyte     =   ":" / " " / tab / nl

tab             =   %d09

mnl             =   1*system-nl

system-nl       =   nl [ 1*(message nl) ]

nl              =   %d13 %d10

message         =   [1*( messagestart "-" bytestring nl)]
                    messagestart " " bytestring nl

messagestart    =   "% " digit digit digit

goodmessage     =   [1*( goodmessagestart "-" bytestring nl)]
                    goodmessagestart " " bytestring nl

goodmessagestart=   "% 200"

messagestart    =   "% " digit digit digit

badmessage      =   [1*( badmessagestart "-" bytestring nl)]
                    badmessagestart " " bytestring nl

badmessagestart =   "% 5" digit digit

endmessage      =   endmessageclose / endmessagecont

endmessageclose =   [endmessagestart " " bytestring nl]
                    byemessage

endmessagecont  =   endmessagestart " " bytestring nl

endmessagestart =   "% 226"

byemessage      =   byemessagestart " " bytestring nl

endmessagestart =   "% 203"

number          =   1*( digit )
```

```
   digit           =   "0" / "1" / "2" / "3" / "4" / "5" /
                       "6" / "7" / "8" / "9"
```

Appendix H - Description of Regular expressions

The regular expressions described in this section are the same as used in many other applications and operating systems. However, it is very simple and does not include logical operators AND and OR.

Searches using regular expressions always use substring matching except when the regular expression contains the characters '^' or '$'.

Character	Function
<any except those listed in this table>	Matches itself
.	Matches any character
a*	Matches zero or more 'a'
[ab]	Matches 'a' or 'b'
[a-c]	Matches 'a', 'b' or 'c'
^	Matches beginning of a token
$	Matches end of a token

Examples

String	Matches	Doesn't match
hello	xhelloy	heello
h.llo	hello	helio
h.*o	hello	helloa
h[a-f]llo	hello	hgllo
^he.*	hello	ehello
.*lo$	hello	helloo