Architecture of the WHOIS++ service

STATUS OF THIS MEMO

This document is an Internet-Draft. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

To learn the current status of any Internet-Draft, please check the "1id-abstracts.txt" listing contained in the Internet-Drafts Shadow Directories on ds.internic.net (US East Coast), nic.nordu.net (Europe), ftp.isi.edu (US West Coast), or munnari.oz.au (Pacific Rim).

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

Part I - WHOIS++ Overview

1.1. Purpose and Motivation............................................. 2
1.2. Basic Information Model........................................... 3
1.2.1. Changes to the current WHOIS Model.......................... 3
1.2.2. Registering WHOIS++ servers................................ 4
1.2.3. The WHOIS++ Search Selection Mechanism..................... 5
1.2.4. The WHOIS++ Architecture..................................... 6
1.3. Indexing in WHOIS++............................................... 6
1.4. Getting Help...................................................... 7
1.4.1. Minimum HELP Required....................................... 8
1.5. Options and Constraints.......................................... 8
1.6. Formatting Responses.............................................. 8
1.7. Reporting Warnings and Errors................................... 9
1.8. Privacy and Security Issues...................................... 9

Part II - WHOIS++ Implementation

2.1. Introduction........................................................ 11
2.1.1. The WHOIS++ interaction model................................ 11
2.2. The WHOIS++ Command set.......................................... 12
2.2.1. System Commands................................................. 12
2.2.1.1. The COMMANDS command.................................... 13
2.2.1.2. The CONSTRAINTS command.................................. 13
2.2.1.3. The DESCRIBE command..................................... 14
2.2.1.4. The HELP command.......................................... 14
2.2.1.5. The LIST command........................................ 14
2.2.1.6. The POLLED-BY command.................................... 14
2.2.1.7. The POLLED-FOR command.................................. 15
2.2.1.8. The SHOW command......................................... 15
2.2.1.9. The VERSION command...................................... 15
2.2.2. The Search Command............................................ 15
2.2.2.1. Format of a Search Term.................................. 16
2.2.2.2. Format of a Search String................................ 17
2.3. WHOIS++ Constraints............................................... 18
2.3.1. Required Constraints.......................................... 18
2.3.2. Optional Constraints.......................................... 19
2.3.2.1. The SEARCH Constraint.................................... 20
2.3.2.2. The FORMAT Constraint.................................... 21
2.3.2.3. The MAXFULL Constraint................................... 21
2.3.2.4. The MAXHITS Constraint................................... 21
2.3.2.5. The CASE Constraint....................................... 21
2.3.2.6. The AUTHENTICATE Constraint.............................. 21
2.3.2.7. The NAME Constraint...................................... 22
2.3.2.8. The PASSWORD Constraint.................................. 22
2.3.2.9. The LANGUAGE Constraint.................................. 22
2.3.2.10. The INCHARSET Constraint................................ 22
2.3.2.11. The IGNORE Constraint................................... 23
2.3.2.12. The INCLUDE Constraint................................. 23
2.4. Server Response Modes............................................... 23
2.4.1. Default Responses.................................................... 24
2.4.2. Format of Responses................................................... 24
2.4.3. Syntax of a Formatted Response..................................... 24
2.4.3.1. A FULL format response............................................ 25
2.4.3.2. ABRIDGED Format Response.......................................... 26
2.4.3.3. HANDLE Format Response............................................. 26
2.4.3.4. SUMMARY Format Response........................................... 27
2.4.3.5. SERVERS-TO-ASK Response......................................... 27
2.4.4. System Generated Messages......................................... 28
2.5. Compatibility with Older WHOIS Servers............................. 28
3. Miscellaneous.............................................................. 28
3.1. Acknowledgements........................................................ 28
3.2. Contact information..................................................... 29

Appendix A - Some Sample Queries........................................ 30
Appendix B - Some sample responses........................................ 31
Appendix C - Sample responses to system commands........................ 33
Appendix D - Sample whois++ session..................................... 35
Appendix E - System messages................................................. 37
Appendix F - The WHOIS++ BNF Grammar.................................... 39
Appendix G - Description of Regular expressions.......................... 41
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 application 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 and most specifically structured the data to make both the client and the server part of the dialogue more stringent and parseable. An optional authentication mechanism for protecting all or parts of the associated WHOIS++ information database from unauthorized access is also briefly described.

The basic 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 in a separate paper.
1.2. Basic Information Model

Our extensions to the existing WHOIS service are centered upon a recommendation to structure user information around a series of standardized information templates, such as to those described by [IAFA1]. Such templates consist of ordered sets of data elements (or attribute-value pairs) and a number of groups at the IETF are now working on standardizing their format and content [IAFA], [NIR].

It is intended that adding such structured templates to a server and subsequently identifying and searching them be simple tasks. The creation and use of customized templates should also be possible with little effort, although their use should be discouraged where appropriate standardized templates exist.

We also offer a set of extensions to the trivial protocol described in RFC954 [HARR85] 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 application where the high cost of configuring and operating traditional White Pages services can not currently be justified.

Also note that the new 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, if desired.

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 at least to some degree readable by humans, to ensure low entry cost and to ease debugging.

The actual implementation details of of an individual WHOIS search engine are left to the imagination of the implementor and 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 identifier (the "handle"). Each record contains one or more lines of information. Currently, there is no structure or implicit ordering of this information, although by implication each record is 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, or simple attribute/value pairs. Each individual record contains a specified ordered set of these data elements.

Secondly, we have introduced typing of the database records. In effect, each record is based upon one of a specified set of templates, each containing a finite and specified number of data elements. This allows users to easily limit searches to specific collections of information, such as information about users, services, abstracts of papers, descriptions of software, and so on.

As a final extension, 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.

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 server handle and a unique record handle pair. This pair is called the 'composed 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). There are many ways of guaranteeing uniqueness of server handles; we will discuss them in a separate paper.

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 better able to specify their searches, leading to a more useful service.
1.2.3. The WHOIS++ Search Selection Mechanism

The WHOIS++ search mechanism is intended to be extremely simple. A search command consists of one or more search terms, with an optional `nvset` of global constraints (specifiers that modify or control a search).

Search terms allow the user to specify template type, attribute, value or handle that any record returns must satisfy. Each search term can have an optional set of local constraints that apply to only that term.

A WHOIS++ database may be seen as a single rolodex-like collection of typed records. Each 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. Boolean searches are possible by the use of AND, OR, NOT and parenthesis.
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 a indexing server. 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.

Each base level server and index server which wishes 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: if a whois++ server contained exactly three records, as follows:

Record 1
Template: Person
First-Name: John
Last-Name: Smith
Favourite-Drink: Labatt Beer

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

Domain: 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 wishes. 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.

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 therefore 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>

may give information on any topic.

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. These would 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 summarized in section 2.2. WHOIS++ constraints are described in section 2.4. Optional commands and constraints are described in section 2.5.

1.6. Formatting Responses

The output returned by a WHOIS++ server is structured to allow machine parsing and automated handling. Of particular interest in the ability to return summary information about a search (without 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 SERVERS-TO-ASK. Note that a conforming server is only required to support the first four formats.

When available, SERVERS-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 2.5.

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 2.5, and details of WHOIS++ warnings and error conditions are given in section appendix E.

All system messages are numerical, but can be tagged with text. It is the clients 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, an optional mechanism is provided for authenticating each WHOIS++ transaction.

The current identified authentication mechanism is PASSWORD, which uses simple password authentication. Any other scheme name used must begin with the characters "X-" and should thus be regarded as experimental and non-standard.

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 mechanisms to use. This mechanism is extensible and individual implementors are free to add additional mechanisms.

This document includes a very simple authentication scheme where a combination of username and password is sent together with the search string so the server can verify that the user have access to the information. Note that this is NOT by any means a method recommended to secure the data itself because both password and information are transferred unencrypted over the network.

Given the unauthenticated nature that default services like white pages services are, it is easy to either forget the implications of this and just show all data to the public Internet, or think that Internet is so dangerous that information is hidden from the Internet so the whole idea of a global whitepages service is lost. Therefore the type of authentication scheme selected and the public nature of the Internet environment must still be taken into consideration when assessing the security and authentication of the information served.
A more detailed exposition on security is outside the scope of this document.
2. Part II - WHOIS++ Implementation

2.1. Introduction

The WHOIS++ protocol specifies the interactions between a WHOIS client and a WHOIS server supporting the WHOIS++ extensions. These extensions are designed to be backwards compatible with existing servers, in the sense that a new server receiving any of the older commands specified in RFC 954 [HARR85] will behave in the same manner as the original NIC WHOIS server.

Obviously, it is not possible to ensure desired behaviour if one of the extended commands is sent to an older WHOIS server, since the requested functionality is simply not there. Still, it would be possible to query whether the WHOIS++ command set is supported as an attribute for each WHOIS server in an appropriate services registry (which itself could be set up using a WHOIS++ server). Thus, in practice this should not be a problem. In addition, any such command sent to an older WHOIS server would simply be treated as a search term, and thus no harm should result.

The small number of older servers, and the probability that at least some of the older servers will be converted to WHOIS++ as implementations become available, means that backwards compatibility is not expected to be a problem in practice.

2.1.1. The WHOIS++ interaction model

A WHOIS++ server will normally listen for a TCP connections on the allocated WHOIS port (port 43) (although a WHOIS++ server can be accessed over any TCP connection). Once a connection is established, the server issues a banner message, then listens for input. The command specified in this input is processed and the results returned including an ending system message. If the optional HOLD constraint has not been specified 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 line of input. This cycle continues as long as the sender continues to append the required HOLD constraint to each subsequent command.

At the same time, each server is permitted to set an optional timeout value (which should be indicated in the response to the CONSTRAINTS command). If set, the server is free to terminate an idle connection at any time after this delay has passed with no input from the client. If the server terminates the connection due to timeout, it will be indicated by the system message. The timeout value is not changeable by the client.
2.2. The WHOIS++ Command set

There are two types of WHOIS++ commands - system commands and the WHOIS++ search command.

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. Details of the allowable search terms for the search command are included in Table II.

Each WHOIS++ command also allows the use of one or more controlling constraints, which select can be used to override defaults or otherwise modify server behavior. There is a core set of constraints that must be supported by all conforming servers. These include 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 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, indicate the need and type of authentication to perform on a transaction, and permit 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.

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 (which is intended, among other things, to support the automated registration of the service by yellow pages directory services). All of these commands are required from a conforming WHOIS++ server.

<table>
<thead>
<tr>
<th>Short</th>
<th>Long Form</th>
<th>Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>-----</td>
<td>-----------------------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>COMMANDS</td>
<td>[ ':' HOLD ]</td>
<td>list valid WHOIS++ commands</td>
</tr>
</tbody>
</table>
Supported by this server

CONSTRAINTS  [ ‘:’ HOLD ]  List valid constraints supported by this server

DESCRIBE  [ ‘:’ HOLD ]  Describe this server, formatting the response using the standard IAFA "Services" template

'? '  HELP [<string>] [‘:’ <cnstrnts>]  System help, using standard IAFA "Help" template

LIST  [‘:’ <cnstrnts>]  List templates supported by this system

POLLED-BY  [ ‘:’ HOLD ]  List indexing servers that are known to track this server

POLLED-FOR  [ ‘:’ HOLD ]  List information about what this server is tracking for

SHOW <string> [‘:’ <cnstrnts>]  Show contents of templates specified

VERSION  [ ‘:’ HOLD ]  return current version of the protocol supported.

Table I - Required WHOIS++ SYSTEM commands.

2.2.1.1. The COMMANDS command

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

2.2.1.2. The CONSTRAINTS command

The CONSTRAINTS command returns a list of constraints and the values of those 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 fields:

- "Constraint", which contains the attribute name described - "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" field, 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.

2.2.1.3. The DESCRIBE command

This is equivalent to issuing the search command on the local server with only the terms "template=services" and "subject=describe" and will result in the display of the corresponding SERVICES template with an attribute of "subject" and value of "describe", except that the DESCRIBE command only searches local information and may not return pointers to other servers.

2.2.1.4. The HELP command

The HELP command takes an optional argument as subject to get help for. This is equivalent to issuing the search command on the local server only with the terms "template=help and subject=<subject>" (or "subject=help" if no argument specified) and will result in the display of the corresponding HELP template with subject "help". The HELP command differs from the above search command in that the HELP command only searches local information and may not return pointers to other servers.

2.2.1.5. The LIST command

The LIST command returns the name of the templates available on the server. The answer is in ABRIDGED format with the template name as the first word on each line.

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 server polled as centroids from this server. The format is in FULL format with two attributes, Template and Field. Each of these is a list of names of the templates or fields polled.
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.

2.2.1.8. The SHOW command

The SHOW command takes a template name as argument and returns information about a specific 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

This is equivalent to issuing the search command on the local server only with the terms "template=version" and will result in the display of the VERSION template, except that the VERSION command only searches local information and may not return pointers to other servers.

The output format is a FULL response containing a record with template name VERSION. The handle for this record is unspecified. The record must have attribute name "Version", which value is "1.0" 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.

2.2.2. The Search Command

A search command consists of one or more search terms, which might each have local constraints, followed by an optional colon with a set of global search constraints.

Each attribute value in the WHOIS++ database is divided into one or more words separated by whitespace. Each search term operates on every word in the attribute value.

Two or more search terms may be combined with boolean operators AND, OR or NOT (other than the implied AND between terms). The operator AND has higher precedence than the operator OR, but this can be changed by the use of parentheses.

Search constraints that apply to every search term are specified as global constraints. Local constraints override global constraints for the search term they are bound to. The search terms and the
global constraints are separated with a colon (’:’). Additional
global constraints are appended to the end of the search command
delimited with a semicolon ’;’.

If different search constraints can not be fulfilled, or the combi-
nation of different search constraints is uncombinable, the server
may choose to ignore some constraints, but still do the search and
return some records.

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

As an option, the server may accept specifications for attributes
for either inclusion or exclusion 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, followed by an optional comma and set of
   comma-separated local constraints.

2) A search term specifier (as listed in Table II), followed by ‘=’,
   followed by a search string, an optional comma and a set of
   comma-separate local constraints.

3) An abbreviated search term specifier, followed by a search
   string, followed by an optional comma and set of comma-separate
   local constraints.

4) A combination of attribute name, followed by ‘=’, followed by a
   search string, followed by an optional comma and set of
   comma-separate local constraints.

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

If a HANDLE specifier is used then the search term can specify
either a composed handle or a record handle. The server is respon-
sible for resolving the composed handle to a server handle and a
record handle.

If a SEARCH-ALL specifier is used then the search will be applied
to all template names, handles, attribute names and attribute
values.

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 2.5.

Valid specifiers:

<table>
<thead>
<tr>
<th>Name</th>
<th>Functionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATTRIBUTE-VALUE [ ',', &lt;constrnt&gt;]</td>
<td>allows combining attribute and value specifiers in one term.</td>
</tr>
<tr>
<td>HANDLE [ ',', &lt;constrnt&gt;]</td>
<td>Confine search to handles.</td>
</tr>
<tr>
<td>SEARCH-ALL [ ',', &lt;constrnt&gt;]</td>
<td>Search everything.</td>
</tr>
<tr>
<td>TEMPLATE [ ',', &lt;constrnt&gt;]</td>
<td>Confine search to template names.</td>
</tr>
<tr>
<td>VALUE [ ',', &lt;constrnt&gt;]</td>
<td>Confine search to attribute values. This is the default.</td>
</tr>
</tbody>
</table>

(Note: The name HANDLE can be replaced with the shortname ']').

Acceptable forms of a search specifier:

1) <searchstring> [’,’ <constraint>]*

2) <specifier> = <searchstring> [’,’ <constraint>]*

3) <shortspecifier> <searchstring> [’,’ <constraint>]*

4) <attribute_name> = <searchstring> [’,’ <constraint>]*

(Note: A <constraint> is a name of a valid local constraint.)

Table II - Valid search command term specifiers.

2.2.2.2. Format of a Search String

Special characters that need to be quoted are preceded 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. It might also specify "LANGUAGE=Fr", which implies that the server should use French in fuzzy matches. It might also be able to issue system messages in French.

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

All constraints can be used as a global constraint, but only a few can be used as local. See tables IV and V for information of which constraints can be local.

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

If a server cannot satisfy the specified constraint there will be a mechanism for informing the user in the reply, using system messages. In such cases, the search is still performed, with the server ignoring unsupported constraints.

2.3.1. Required Constraints

The following CONSTRAINTS must be supported in all conforming WHOIS++ servers.
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.
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 G.
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 will 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 did or did not ask 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 the client can get in a search response.

2.3.2.5. The CASE Constraint

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

2.3.2.6. The AUTHENTICATE Constraint

The AUTHENTICATE constraint describes which authentication method
to use when executing the search. By using a specific authentica-
tion method, some other constraints might be needed which is speci-
ified by the authentication method.

The only authentication method described in this document is "pass-
word", if used, also the two other constraints "name" and "pass-
word" need to be set.

2.3.2.7. The NAME Constraint

The NAME constraint is only used together with some authentication
method named by the constraint "authenticate". The only use
described in this document is by sending a username as a string of
characters which together with the string given as an argument to
the "password" constraint is sent to the server. The server can use
that pair of strings to do a simple authentication check, like the
UNIX login program do.

2.3.2.8. The PASSWORD Constraint

The PASSWORD constraint is only used together with some authentica-
tion method named by the constraint "authenticate". The only use
described in this document is by sending a password as a string of
characters which together with the string given as an argument to
the "name" constraint is sent to the server. The server can use
that pair of strings to do a simple authentication check, like the
UNIX login program do.

2.3.2.9. The LANGUAGE Constraint

The LANGUAGE constraints can 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, although
this ability should be handled by the client. The two-letter
language code defined in ISO 639:1988 can be used as a value for
the language constraint. In these, the case of the letters are
insignificant. Other language codes shall be interpreted according
to the Internet standard for language codes in RFC 822/MIME mes-
sages, if and when such a standard is adopted.

2.3.2.10. The INCHARSET Constraint

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

The IGNORE constraint specifies which attributes to NOT 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 must be "% 205 Requested constraint not fulfilled".

2.3.2.12. 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 "% 205 Requested constraint not fulfilled".

2.4. Server Response Modes

There are currently a total of five different response modes possible for WHOIS++ servers. These are FULL, ABRIDGED, HANDLE, SUMMARY and SERVERS-TO-ASK. The syntax of each output format is specified in more detail in the following section.

1) A FULL format response provides the complete contents of each template matching the specified query, including the template type and the composed handle for each record.

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

3) A HANDLE format response returns only a list of handles 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 SERVERS-TO-ASK response returns only a pointer to another WHOIS++ server, which might possibly be able to answer the specified query.

The server may respond with a null answer and may also respond with a null answer together with a correct system message to indicate
that the query was too complex.

2.4.1. Default Responses

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

The server is allowed to provide response in SUMMARY format if the number of hits exceeds the value of the global constraint "max-full".

The server will not respond with more matches than the value specified with the global constraint "maxhits"; Not in any response format. If the number of matches exceeds this value, the server will issue 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.

The server response modes are summarized in Table V.

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 messages.

That is:

```
'%' <system messages> <nl>

[ <formatted response> ]

'%' <system messages> <nl>
```

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 format for system messages, see Appendix E.

2.4.3. Syntax of a Formatted Response

All formatted responses consist of a START line, followed by a
response-specific section, followed by a TERMINATION line. It is permissible to insert any number of lines consisting solely of newlines within a formatted response to improve readability.

A START line consists of a line beginning with a ‘#’ in the first column, followed by one white space character (SPACE or TAB), followed by one of the following keywords FULL, ABRIDGED, HANDLE, SUMMARY or SERVERS-TO-ASK.

A START line must contain no more than 81 characters, including the terminating newline character.

A TERMINATION line consists of a line beginning with a ‘#’ in the first column, followed by one white space character (SPACE or TAB), followed by the keyword END, followed by zero or more characters, followed by a newline.

A TERMINATION line must contain no more than 81 characters, including the terminating newline character.

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) SERVERS-TO-ASK Format Response

The details of each are specified in the following sections:

2.4.3.1. A FULL format response

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

Each FORMAT specifier line consists of a ‘#’ in the first column, followed by one white space character, the name of the corresponding template type, one white space character, the serverhandle, a colon, the handle for that record, and a terminating newline.

The template information for each 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 newline.
Each such line shall be limited to no more than 81 characters, including the terminating newline. If a line (including the required leading single space) would exceed 81 characters, it is to 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 space.

If the attribute value 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 space character. The attribute name is not repeated on consecutive lines.

2.4.3.2. ABRIDGED Format Response

An ABRIDGED format response consists of a single set of responses, consisting of a single line excerpt of the template information from each matching record. The excerpt information shall include, as a minimum, the composed handle of the record, as well as other information that is relevant to the template type.

The abridged template information for each record will be returned as a series of lines, each of which must consist of a single space, followed by the abridged line of the record, one or more space characters, the composed handle, and a CR/LF pair.

Each line shall be limited to no more than 81 characters, including the terminating newline. If a line (including the required single space, would exceed 81 characters, it is to be broken into lines of no more than 81 characters, with the remainder following on the subsequent line, with the space replaced by a "+" character in the first column.

If the attribute value 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 space character. The attribute name is not repeated on consecutive lines.

2.4.3.3. HANDLE Format Response

A HANDLE format response consists of a single set of responses, consisting of a single line listing the composed handle and template type for each matching record.

Each line shall start with one space, followed by the server handle, a colon, the record handle, one or more whitespace characters, the template type and terminated by a newline.

Each such line must contain no more than 81 characters, including the terminating newline character. If a line (including the required first space) would exceed 81 characters, it shall be split
into multiple lines, with each continuation line beginning with a ‘+’ instead of a space.

2.4.3.4. SUMMARY Format Response

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

The first line shall begin with the string "matches: ", be followed by a space and the number of responses to the query and terminated by a newline. The second line shall begin with the string "templates: ", be followed by a newline 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.

If the line is longer than 81 characters including the terminating CR/LF pair, it shall be split into multiple lines with each continuation line beginning with a ‘+’ instead of a space.

2.4.3.5. SERVERS-TO-ASK Response

A SERVERS-TO-ASK response consists of information to the client about which servers to contact next to resolve a query.

The servers-to-ask response will consist of a number of attribute-value pairs, separated by CRLF. Each line is indented with one space.

Required attributes are "Version-number", which will be "1.0", and "Next-Servers".

The "Next-Servers" field will be returned as a series of lines, each holding pointer information to one server. Consecutive lines shall have a hyphen "-" in the first column. Each line will be separated into five fields, separated with a semicolon. Those fields are:

1. The server handle of the server pointed at. (required)
2. A cached host named for the server pointed at. (optional)
3. A cached port number for the server pointed at. (optional)

Each such line shall be limited to no more than 81 characters, including the terminating newline. If a line (including the required leading single space) would exceed 81 characters, it is to be broken into lines of no more than 81 characters, with each continuation line beginning with a "+" character in the first column.
2.4.4. System Generated Messages

All system generated messages must begin with 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

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.
3.2. Contact information

Peter Deutsch,
BUNYIP INFORMATION SYSTEMS, inc
310 St-Catherine St West,
suite 202,
Montreal, Quebec H2X 2A1
CANADA
<peterd@bunyip.com>

Rickard Schoultz,
KTHNOC, SUNET/NORDUnet/Ebone Operations Centre
100 44 STOCKHOLM
SWEDEN
<schoultz@sunet.se>

Patrik Faltstrom
BUNYIP INFORMATION SYSTEMS, inc
310 St-Catherine St West,
suite 202,
Montreal, Quebec H2X 2A1
CANADA
<paf@bunyip.com>

Chris Weider
BUNYIP INFORMATION SYSTEMS, inc
2001 S. Huron Parkway, #12
Ann Arbor, MI 48104
USA
<clw@bunyip.com>
Appendix A - Some Sample Queries

author=chris and template=user

The result will consist of all records where attribute "author" matches "chris" with case ignored. Only USER templates will be searched. An example of a matching record is "Author=Chris Weider". This is the typical case of search.

schoultz and rick; search=lstring

The result will consist of all records which have one attribute value matching "schoultz" exactly and one having "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 "value" term specifier is the default, the search term could be "phone" as well as "value=phone").

search-all=Peter ; search=substring; case=consider

The result will consist of all records which have attribute names, template names or attribute values matching "Peter" with respect to case. One example is "Friend-Of-Peter: Yes".

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) A FULL format response:

# FULL

# USER SERVHANDLE1:PD45
Name: Peter Deutsch
email: peterd@bunyip.com

# USER SERVHANDLE1:AE1
Name: Alan Emtage
email: bajan@bunyip.com

# USER SERVHANDLE1:NW1
Name: Nick West
Favourite-Bicycle-Forward-Wheel-Brand: New Bicycles 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.

# SERVICES SERVHANDLE1:WWW1
Type: World Wide Web
Location: the world

# END

---------------------

2) An ABRIDGED format response:

# ABRIDGED

Peter Deutsch        peterd@bunyip.com        PD45
Alan Emtage          bajan@bunyip.com         AE1
World Wide Web       the world                WWW1

# END

---------------------

3) A HANDLE format response:

# HANDLE

SERVHANDLE1:PD45        User
SERVHANDLE1:AE1        User
4) A SUMMARY HANDLE format response:

# SUMMARY

Matches: 175
Templates: User
- Services
- Abstracts
# END
Appendix C - Sample responses to system commands

C.1 Response to the LIST command

# ABRIDGED
USER
SERVICES
HELP
# END

C.2 Response to the SHOW command

This example show the result after issuing "show help":

# FULL
# TEMPLATE serverhandle1:user-template
  Template-Name: USER
  Attribute-Names: Name,Organization-Name,Organization-Type,Work-Phone,
  +Work-Fax,Work-Postal,Job-Title,Department,Email,Handle,Home-Phone,
  +Home-Postal,Home-Fax
# TEMPLATE serverhandle1:help
  Template-Name: HELP
  Attribute-Names: Subject,Description,Handle
# END

C.3 Response to the POLLED-BY command

# FULL
# POLLED-BY xyz:serverhandle1
  Server-handle: serverhandle1
  Cached-Host-Name: sunic.sunet.se
  Cached-Host-Port: 7070
  Template: user
  Field: ALL
# POLLED-BY xyz:serverhandle2
  Server-handle: serverhandle2
  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 xyz:serverhandle3
  Server-Handle: serverhandle3
  Template: ALL
Field: Name, Address, Job-Title, Organization-Name, Organization-Address, +Organization-Name

# POLLED-FOR xyz:serverhandle4
Server-Handle: serverhandle4
Template: User
Field: ALL
# END

C.5 Response to the VERSION command

# FULL
# VERSION serverhandle:version
Version: 1.0
Program-Name: kth-whoisd
Program-Version: 2.0
# END

C.6 Response to the CONSTRAINTS command

# FULL
# CONSTRAINT asdjkq
Constraint: format
Default: full
Range: full, abridged, summary, handle
# CONSTRAINT ljkqwer
Constraint: maxhits
Default: 200
Range: 1-1000
# CONSTRAINT slkjewer
Constraint: search
Default: exact
Range: exact, substring, lstring
# CONSTRAINT qwewerq
Constraint: maxfull
Default: 20
# 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

```plaintext
>% 220-Welcome to
>% 220-the whois++ server
>% 220 at ACME inc.
<name=Nick:hold
>% 200 Command okay
>
>% # FULL
>
>% # USER serverhandle:nw1
> name: Nick West
> email: nick@acme.com
>% # END
>% # SERVERS-TO-ASK
> Version-number: 1.0
> Next-Servers: sunetse01;whois.sunet.se;7070
>-              kthse01
>-              anotherserverhandle01;whois.acme.com;7070
>% # END
>% 226 Tranfer complete
<version
>% 200 Command okay
>% # FULL
>% # VERSION ignoredserverhandle:version
> Version: 1.0
>% # END
>% 226 Tranfer 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 three other servers that either holds records or pointers to other servers. The first server was polled for "USER" templates and "name" and "email" fields.

The client continues with asking for 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, in which case the text part is the name of a character set that will be used by the server in the rest of the response. The valid values for characters sets is specified in the "characterset" list in the BNF listing in appendix F.

The theory of reply codes is described in appendix E in 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 fullfilled
One or more constraints in query has unacceptable value and was therefore not used, but the search is still done.
<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>Command Ok</td>
</tr>
<tr>
<td></td>
<td>Command accepted and executed. The client must wait for a transaction end system message.</td>
</tr>
<tr>
<td>201</td>
<td>Command Completed successfully</td>
</tr>
<tr>
<td></td>
<td>Command accepted and executed.</td>
</tr>
<tr>
<td>203</td>
<td>Bye</td>
</tr>
<tr>
<td></td>
<td>Server is closing connection</td>
</tr>
<tr>
<td>220</td>
<td>Service Ready</td>
</tr>
<tr>
<td></td>
<td>Greeting message. Server is accepting commands.</td>
</tr>
<tr>
<td>226</td>
<td>Transaction complete</td>
</tr>
<tr>
<td></td>
<td>End of data. All responses to query are sent.</td>
</tr>
<tr>
<td>430</td>
<td>Authentication needed</td>
</tr>
<tr>
<td></td>
<td>Client requested information that needs authentication.</td>
</tr>
<tr>
<td>500</td>
<td>Syntax error</td>
</tr>
<tr>
<td>502</td>
<td>Search expression too complicated</td>
</tr>
<tr>
<td></td>
<td>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).</td>
</tr>
<tr>
<td>530</td>
<td>Authentication failed</td>
</tr>
<tr>
<td></td>
<td>The authentication phase failed.</td>
</tr>
<tr>
<td>600</td>
<td>&lt;token&gt;</td>
</tr>
<tr>
<td></td>
<td>Subsequent attribute values are encoded in the character set specified by &lt;token&gt;.</td>
</tr>
</tbody>
</table>

Table V - System response codes
Appendix F - The WHOIS++ BNF Grammar

whois-command = { system-command ["": "hold"] / terms ["": globalcnstrnts] } NL

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

terms = and-expr *("or" and-expr)

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

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

term = generalterm / specificterm / shorthand / combinedterm

generalterm = string *(";" localcnstrnt)

specificterm = specificname "=" string *(";" localcnstrnt)

specificname = "handle" / "value"

shorthand = "!" string *(";" localcnstrnt)

combinedterm = string "=" string *(";" localcnstrnt)

globalcnstrnts = globalcnstrnt *(";" globalcnstrnt)

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

opt-globalcnst = "hold" / "authenticate" "=" auth-method / "name" "=" string / "password" "=" string / "language" "=" language / "incharset" "=" characterset / "ignore" "=" string / "include" "=" string
format = "full" / "abridged" / "handle" / "summary" / "servers-to-ask"

language = <The two-letter language code defined in ISO 639-1988 can be used as language constraint. In these the case of the letters are insignificant. Other language codes shall be interpreted according to the Internet standard for language codes in RFC 822/MIME messages, if and when such a standard is adopted>

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" / charset-value

charset-value = 1*char

localcnstrnt = "search" "=" searchvalue / "case" "=" casevalue

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

casevalue = "ignore" / "consider"

auth-method = "password"

string = 0*char

char = \" specialchar

/ <Characters 0-255 (decimal) except specialchar>

specialchar = " " / <tab> / "=" / "," / ",," / ",;" / ",:" / ",\" / "*** / ",,* / ",{" / ",}" / ",[" / ",]" / ",=" / ",!" / ",?"

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

NL = <CR LF (decimal 13 10)>

NOTE: Significant blanks must be escaped. The following characters, when significant to the query, may be preceded and/or followed by a single blank: : ; , ( ) = !
Appendix G - Description of Regular expressions

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

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

<table>
<thead>
<tr>
<th>Character</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;any except those listed in this table&gt;</td>
<td>Matches itself</td>
</tr>
<tr>
<td>.</td>
<td>Matches any character</td>
</tr>
<tr>
<td>a*</td>
<td>Matches zero or more 'a'</td>
</tr>
<tr>
<td>[ab]</td>
<td>Matches 'a' or 'b'</td>
</tr>
<tr>
<td>[a-c]</td>
<td>Matches 'a', 'b' or 'c'</td>
</tr>
<tr>
<td>^</td>
<td>Matches beginning of a token</td>
</tr>
<tr>
<td>$</td>
<td>Matches end of a token</td>
</tr>
</tbody>
</table>

Examples

<table>
<thead>
<tr>
<th>String</th>
<th>Matches</th>
<th>Matches not</th>
</tr>
</thead>
<tbody>
<tr>
<td>hello</td>
<td>xhelloy</td>
<td>heello</td>
</tr>
<tr>
<td>h.llo</td>
<td>hello</td>
<td>hello</td>
</tr>
<tr>
<td>h.*o</td>
<td>hello</td>
<td>helloa</td>
</tr>
<tr>
<td>h[a-f]llo</td>
<td>hello</td>
<td>hglio</td>
</tr>
<tr>
<td>^he.*</td>
<td>hello</td>
<td>ehello</td>
</tr>
<tr>
<td>.*1o$</td>
<td>hello</td>
<td>helloo</td>
</tr>
</tbody>
</table>
References


[IAFA] Internet Anonymous FTP Archives Working Group (now closed).

[IAFA1] Emtage A., and Deutsch P.. IAFA (Internet Anonymous FTP Archives)
(not yet issued, pending some editing...)


Expires: 25 July 95