Network Working Group
Request for Comments: 2229
Category: Informational

R. Faith
U. North Carolina, Chapel Hill
B. Martin
Miranda Productions

October 1997

A Dictionary Server Protocol

Status of this Memo

Copyright Notice

Abstract

The Dictionary Server Protocol (DICT) is a TCP transaction based query/response protocol that allows a client to access dictionary definitions from a set of natural language dictionary databases.

Table of Contents

1. Introduction

For many years, the Internet community has relied on the "webster" protocol for access to natural language definitions. The webster protocol supports access to a single dictionary and (optionally) to a single thesaurus. In recent years, the number of publicly available webster servers on the Internet has dramatically decreased.

Fortunately, several freely-distributable dictionaries and lexicons have recently become available on the Internet. However, these freely-distributable databases are not accessible via a uniform interface, and are not accessible from a single site. They are often small and incomplete individually, but would collectively provide an interesting and useful database of English words. Examples include the Jargon file [JARGON], the WordNet database [WORDNET], MICRA's version of the 1913 Webster's Revised Unabridged Dictionary [WEB1913], and the Free Online Dictionary of Computing [FOLDOC]. Translating and non-English dictionaries are also becoming available (for example, the FOLDOC dictionary is being translated into Spanish).

The webster protocol is not suitable for providing access to a large number of separate dictionary databases, and extensions to the current webster protocol were not felt to be a clean solution to the dictionary database problem.

The DICT protocol is designed to provide access to multiple databases. Word definitions can be requested, the word index can be searched (using an easily extended set of algorithms), information about the server can be provided (e.g., which index search strategies are supported, or which databases are available), and information about a database can be provided (e.g., copyright, citation, or distribution information). Further, the DICT protocol has hooks that can be used to restrict access to some or all of the databases.

1.1. Requirements

In this document, we adopt the convention discussed in Section 1.3.2 of [RFC1122] of using the capitalized words MUST, REQUIRED, SHOULD, RECOMMENDED, MAY, and OPTIONAL to define the significance of each particular requirement specified in this document.

In brief: "MUST" (or "REQUIRED") means that the item is an absolute requirement of the specification; "SHOULD" (or "RECOMMENDED") means there may exist valid reasons for ignoring this item, but the full implications should be understood before doing so; and "MAY" (or "OPTIONAL") means that his item is optional, and may be omitted without careful consideration.

2. Protocol Overview

2.1. Link Level

The DICT protocol assumes a reliable data stream such as provided by TCP. When TCP is used, a DICT server listens on port 2628.

This server is only an interface between programs and the dictionary databases. It does not perform any user interaction or presentation-level functions.

2.2. Lexical Tokens

Commands and replies are composed of characters from the UCS character set [ISO10646] using the UTF-8 [RFC2044] encoding. More specifically, using the grammar conventions from [RFC822]:

```
                                          ; (  Octal, Decimal.)
CHAR        =  <any UTF-8 character (1 to 6 octets)>
CTL         =  <any ASCII control         ; (  0- 37,  0.- 31.)
               character and DEL>         ; (    177,    127.)
CR          =  <ASCII CR, carriage return>  ; (     15,     13.)
LF          =  <ASCII LF, linefeed>         ; (     12,     10.)
SPACE       =  <ASCII SP, space>            ; (     40,     32.)
HTAB        =  <ASCII HT, horizontal-tab>   ; (     11,      9.)
<">         =  <ASCII quote mark>           ; (     42,     34.)
<'>         =  <ASCII single quote mark>    ; (     47,     39.)
CRLF        =  CR LF
WS          =  1*(SPACE / HTAB)

dqstring    =  <"> *(dqtext/quoted-pair) <">
dqtext      =  <any CHAR except <">, "\", and CTLs>
sqstring    =  <'> *(dqtext/quoted-pair) <'>
sqtext      =  <any CHAR except <'>, "\", and CTLs>
quoted-pair =  "\" CHAR

atom        =  1*<any CHAR except SPACE, CTLs, <'>, <">, and "\">
string      =  *<dqstring / sqstring / quoted-pair>
word        =  *<atom / string>
description =  *<word / WS>
text        =  *<word / WS>
```

2.3. Commands

Commands consist of a command word followed by zero or more parameters. Commands with parameters must separate the parameters from each other and from the command by one or more space or tab characters. Command lines must be complete with all required parameters, and may not contain more than one command.

Each command line must be terminated by a CRLF.

The grammar for commands is:

```
          command     = cmd-word *<WS cmd-param>
          cmd-word    = atom
          cmd-param   = database / strategy / word
          database    = atom
          strategy    = atom
```

Commands are not case sensitive.

Command lines MUST NOT exceed 1024 characters in length, counting all characters including spaces, separators, punctuation, and the trailing CRLF. There is no provision for the continuation of command lines. Since UTF-8 may encode a character using up to 6 octets, the command line buffer MUST be able to accept up to 6144 octets.

2.4. Responses

Responses are of two kinds, status and textual.

2.4.1. Status Responses

Status responses indicate the server's response to the last command received from the client.

Status response lines begin with a 3 digit numeric code which is sufficient to distinguish all responses. Some of these may herald the subsequent transmission of text.

The first digit of the response broadly indicates the success, failure, or progress of the previous command (based generally on [RFC640,RFC821]):

```
1yz - Positive Preliminary reply
2yz - Positive Completion reply
3yz - Positive Intermediate reply
4yz - Transient Negative Completion reply
5yz - Permanent Negative Completion reply
```

The next digit in the code indicates the response category:

```
x0z - Syntax
x1z - Information (e.g., help)
x2z - Connections
x3z - Authentication
x4z - Unspecified as yet
x5z - DICT System (These replies indicate the status of the
      receiver DICT system vis-a-vis the requested transfer
      or other DICT system action.)
x8z - Nonstandard (private implementation) extensions
```

The exact response codes that should be expected from each command are detailed in the description of that command.

Certain status responses contain parameters such as numbers and strings. The number and type of such parameters is fixed for each response code to simplify interpretation of the response. Other status responses do not require specific text identifiers. Parameter

requirements are detailed in the description of relevant commands. Except for specifically detailed parameters, the text following response codes is server-dependent.

Parameters are separated from the numeric response code and from each other by a single space. All numeric parameters are decimal, and may have leading zeros. All string parameters MUST conform to the "atom" or "dqstring" grammar productions.

If no parameters are present, and the server implementation provides no implementation-specific text, then there MAY or MAY NOT be a space after the response code.

Response codes not specified in this standard may be used for any installation-specific additional commands also not specified.

These should be chosen to fit the pattern of x8z specified above. The use of unspecified response codes for standard commands is prohibited.

2.4.2. General Status Responses

In response to every command, the following general status responses are possible:

```
500 Syntax error, command not recognized
501 Syntax error, illegal parameters
502 Command not implemented
503 Command parameter not implemented
420 Server temporarily unavailable
421 Server shutting down at operator request
```

2.4.3. Text Responses

Before text is sent a numeric status response line, using a 1yz code, will be sent indicating text will follow. Text is sent as a series of successive lines of textual matter, each terminated with a CRLF. A single line containing only a period (decimal code 46, ".") is sent to indicate the end of the text (i.e., the server will send a CRLF at the end of the last line of text, a period, and another CRLF).

If a line of original text contained a period as the first character of the line, that first period is doubled by the DICT server. Therefore, the client must examine the first character of each line received. Those that begin with two periods must have those two periods collapsed into one period. Those that contain only a single period followed by a CRLF indicate the end of the text response.

If the OPTION MIME command has been given, all textual responses will be prefaced by a MIME header [RFC2045] followed by a single blank line (CRLF). See section 3.10.1 for more details on OPTION MIME.

Following a text response, a 2yz response code will be sent.

Text lines MUST NOT exceed 1024 characters in length, counting all characters including spaces, separators, punctuation, the extra initial period (if needed), and the trailing CRLF. Since UTF-8 may encode a character using up to 6 octets, the text line input buffer MUST be able to accept up to 6144 octets.

By default, the text of the definitions MUST be composed of characters from the UCS character set [ISO10644] using the UTF-8 [RFC2044] encoding. The UTF-8 encoding has the advantage of preserving the full range of 7-bit US ASCII [USASCII] values. Clients and servers MUST support UTF-8, even if only in some minimal fashion.

3. Command and Response Details

Below, each DICT command and appropriate responses are detailed. Each command is shown in upper case for clarity, but the DICT server is case-insensitive.

Except for the AUTH and SASLAUTH commands, every command described in this section MUST be implemented by all DICT servers.

3.1. Initial Connection

When a client initially connects to a DICT server, a code 220 is sent if the client's IP is allowed to connect:

```
220 text capabilities msg-id
```

The code 220 is a banner, usually containing host name and DICT server version information.

The second-to-last sequence of characters in the banner is the optional capabilities string, which will allow servers to declare support for extensions to the DICT protocol. The capabilities string is defined below:

```
capabilities =  ["<" msg-atom *("." msg-atom) ">"]
msg-atom     =  1*<any CHAR except SPACE, CTLs,
                   "<", ">", ".", and "\">
```

Individual capabilities are described by a single msg-atom. For example, the string <html.gzip> might be used to describe a server that supports extensions which allow HTML or compressed output. Capability names beginning with "x" or "X" are reserved for experimental extensions, and SHOULD NOT be defined in any future DICT protocol specification. Some of these capabilities may inform the client that certain functionality is available or can be requested. The following capabilities are currently defined:

```
mime        The OPTION MIME command is supported
auth        The AUTH command is supported
kerberos_v4 The SASL Kerberos version 4 mechanism is supported
gssapi      The SASL GSSAPI [RFC2078] mechanism is supported
skey        The SASL S/Key [RFC1760] mechanism is supported
external    The SASL external mechanism is supported
```

The last sequence of characters in the banner is a msg-id, similar to the format specified in [RFC822]. The simplified description is given below:

```
msg-id      =  "<" spec ">"            ; Unique message id
spec        =  local-part "@" domain
local-part  =  msg-atom *("." msg-atom)
domain      =  msg-atom *("." msg-atom)
```

Note that, in contrast to [RFC822], spaces and quoted pairs are not allowed in the msg-id. This restriction makes the msg-id much easier for the client to locate and parse but does not significantly decrease any security benefits, since the msg-id may be arbitrarily long (as bounded by the response length limits set forth elsewhere in this document).

Note also that the open and close brackets are part of the msg-id and should be included in the string that is used to compute the MD5 checksum.

This message id will be used by the client when formulating the authentication string used in the AUTH command.

If the client's IP is not allowed to connect, then a code 530 is sent instead:

```
530 Access denied
```

Transient failure responses are also possible:

```
420 Server temporarily unavailable
421 Server shutting down at operator request
```

For example, response code 420 should be used if the server cannot currently fork a server process (or cannot currently obtain other resources required to proceed with a usable connection), but expects to be able to fork or obtain these resources in the near future.

Response code 421 should be used when the server has been shut down at operator request, or when conditions indicate that the ability to service more requests in the near future will be impossible. This may be used to allow a graceful operator-mediated temporary shutdown of a server, or to indicate that a well known server has been permanently removed from service (in which case, the text message might provide more information).

3.2. The DEFINE Command

DEFINE database word

3.2.1. Description

This command will look up the specified word in the specified database. All DICT servers MUST implement this command.

If the database name is specified with an exclamation point (decimal code 33, "!"), then all of the databases will be searched until a match is found, and all matches in that database will be displayed. If the database name is specified with a star (decimal code 42, "*"), then all of the matches in all available databases will be displayed. In both of these special cases, the databases will be searched in the same order as that printed by the "SHOW DB" command.

If the word was not found, then status code 552 is sent.

If the word was found, then status code 150 is sent, indicating that one or more definitions follow.

For each definition, status code 151 is sent, followed by the textual body of the definition. The first three space-delimited parameters following status code 151 give the word retrieved, the name of the database (which is the same as the first column of the SHOW DB command), and a short description for the database (which is the same as the second column of the SHOW DB command). The short name is suitable for printing as:

From name:

before the definition is printed. This provides source information for the user.

The textual body of each definition is terminated with a CRLF period CRLF sequence.

After all of the definitions have been sent, status code 250 is sent. This command can provide optional timing information (which is server dependent and is not intended to be parsable by the client). This additional information is useful when debugging and tuning the server.

3.2.2. Responses

```
550 Invalid database, use "SHOW DB" for list of databases
552 No match
150 n definitions retrieved - definitions follow
151 word database name - text follows
250 ok (optional timing information here)
```

Response codes 150 and 151 require special parameters as part of their text. The client can use these parameters to display information on the user's terminal.

For code 150, parameters 1 indicates the number of definitions retrieved.

For code 151, parameter 1 is the word retrieved, parameter 2 is the database name (the first name as shown by "SHOW DB") from which the definition has been retrieved, and parameter 3 is the the short database description (the second column of the "SHOW DB" command).

3.3. The MATCH Command

MATCH database strategy word

3.3.1. Description

This command searches an index for the dictionary, and reports words which were found using a particular strategy. Not all strategies are useful for all dictionaries, and some dictionaries may support additional search strategies (e.g., reverse lookup). All DICT servers MUST implement the MATCH command, and MUST support the "exact" and "prefix" strategies. These are easy to implement and are generally the most useful. Other strategies are server dependent.

The "exact" strategy matches a word exactly, although different servers may treat non-alphanumeric data differently. We have found that a case-insensitive comparison which ignores non-alphanumeric

characters and which folds whitespace is useful for English-language dictionaries. Other comparisons may be more appropriate for other languages or when using extended character sets.

The "prefix" strategy is similar to "exact", except that it only compares the first part of the word.

Different servers may implement these algorithms differently. The requirement is that strategies with the names "exact" and "prefix" exist so that a simple client can use them.

Other strategies that might be considered by a server implementor are matches based on substring, suffix, regular expressions, soundex [KNUTH73], and Levenshtein [PZ85] algorithms. These last two are especially useful for correcting spelling errors. Other useful strategies perform some sort of "reverse" lookup (i.e., by searching definitions to find the word that the query suggests).

If the database name is specified with an exclamation point (decimal code 33, "!"), then all of the databases will be searched until a match is found, and all matches in that database will be displayed. If the database name is specified with a star (decimal code 42, "*"), then all of the matches in all available databases will be displayed. In both of these special cases, the databases will be searched in the same order as that printed by the "SHOW DB" command.

If the strategy is specified using a period (decimal code 46, "."), then the word will be matched using a server-dependent default strategy, which should be the best strategy available for interactive spell checking. This is usually a derivative of the Levenshtein algorithm [PZ85].

If no matches are found in any of the searched databases, then status code 552 will be returned.

Otherwise, status code 152 will be returned followed by a list of matched words, one per line, in the form:

```
database word
```

This makes the responses directly useful in a DEFINE command.

The textual body of the match list is terminated with a CRLF period CRLF sequence.

Following the list, status code 250 is sent, which may include server-specific timing and statistical information, as discussed in the section on the DEFINE command.

3.3.2. Responses

```
550 Invalid database, use "SHOW DB" for list of databases
551 Invalid strategy, use "SHOW STRAT" for a list of strategies
552 No match
152 n matches found - text follows
250 ok (optional timing information here)
```

Response code 152 requires a special parameter as part of its text. Parameter 1 must be the number of matches retrieved.

3.4. A Note on Virtual Databases

The ability to search all of the provided databases using a single command is given using the special "*" and "!" databases.

However, sometimes, a client may want to search over some but not all of the databases that a particular server provides. One alternative is for the client to use the SHOW DB command to obtain a list of databases and descriptions, and then (perhaps with the help of a human), select a subset of these databases for an interactive search. Once this selection has been done once, the results can be saved, for example, in a client configuration file.

Another alternative is for the server to provide "virtual" databases which merge several of the regular databases into one. For example, a virtual database may be provided which includes all of the translating dictionaries, but which does not include regular dictionaries or thesauri. The special "*" and "!" databases can be considered as names of virtual databases which provide access to all of the databases. If a server implements virtual databases, then the special "*" and "!" databases should probably exclude other virtual databases (since they merely provide information duplicated in other databases). If virtual databases are supported, they should be listed as a regular database with the SHOW DB command (although, since "*" and "!" are required, they need not be listed).

Virtual databases are an implementation-specific detail which has absolutely no impact on the DICT protocol. The DICT protocol views virtual and non-virtual databases the same way.

We mention virtual databases here, however, because they solve a problem of database selection which could also have been solved by changes in the protocol. For example, each dictionary could be assigned attributes, and the protocol could be extended to specify searches over databases with certain attributes. However, this needlessly complicates the parsing and analysis that must be performed by the implementation. Further, unless the classification

system is extremely general, there is a risk that it would restrict the types of databases that can be used with the DICT protocol (although the protocol has been designed with human-language databases in mind, it is applicable to any read-only database application, especially those with a single semi-unique alphanumeric key and textual data).

3.5. The SHOW Command

3.5.1. SHOW DB

SHOW DB
SHOW DATABASES

3.5.1.1. Description

Displays the list of currently accessible databases, one per line, in the form:

database description

The textual body of the database list is terminated with a CRLF period CRLF sequence. All DICT servers MUST implement this command.

Note that some databases may be restricted due to client domain or lack of user authentication (see the AUTH and SASLAUTH commands in sections 3.11 and 3.12). Information about these databases is not available until authentication is performed. Until that time, the client will interact with the server as if the additional databases did not exist.

3.5.1.2. Responses

110 n databases present - text follows
554 No databases present

Response code 110 requires a special parameter. Parameter 1 must be the number of databases available to the user.

3.5.2. SHOW STRAT

SHOW STRAT
SHOW STRATEGIES

3.5.2.1. Description

Displays the list of currently supported search strategies, one per line, in the form:

```
strategy description
```

The textual body of the strategy list is terminated with a CRLF period CRLF sequence. All DICT servers MUST implement this command.

3.5.2.2. Responses

```
111 n strategies available - text follows
555 No strategies available
```

Response code 111 requires a special parameter. Parameter 1 must be the number of strategies available.

3.5.3. SHOW INFO

SHOW INFO database

3.5.3.1. Description

Displays the source, copyright, and licensing information about the specified database. The information is free-form text and is suitable for display to the user in the same manner as a definition. The textual body of the information is terminated with a CRLF period CRLF sequence. All DICT servers MUST implement this command.

3.5.3.2. Responses

```
550 Invalid database, use "SHOW DB" for list of databases
112 database information follows
```

These response codes require no special parameters.

3.5.4. SHOW SERVER

SHOW SERVER

3.5.4.1. Description

Displays local server information written by the local administrator. This could include information about local databases or strategies, or administrative information such as who to contact for access to databases requiring authentication. All DICT servers MUST implement this command.

3.5.4.2. Responses

114 server information follows

This response code requires no special parameters.

3.6. The CLIENT Command

CLIENT text

3.6.1. Description

This command allows the client to provide information about itself for possible logging and statistical purposes. All clients SHOULD send this command after connecting to the server. All DICT servers MUST implement this command (note, though, that the server doesn't have to do anything with the information provided by the client).

3.6.2. Responses

250 ok (optional timing information here)

This response code requires no special parameters.

3.7. The STATUS Command

STATUS

3.7.1. Description

Display some server-specific timing or debugging information. This information may be useful in debugging or tuning a DICT server. All DICT servers MUST implement this command (note, though, that the text part of the response is not specified and may be omitted).

3.7.2. Responses

210 (optional timing and statistical information here)

This response code requires no special parameters.

3.8. The HELP Command

HELP

3.8.1. Description

Provides a short summary of commands that are understood by this implementation of the DICT server. The help text will be presented as a textual response, terminated by a single period on a line by itself. All DICT servers MUST implement this command.

3.8.2. Responses

113 help text follows

This response code requires no special parameters.

3.9. The QUIT Command

QUIT

3.9.1. Description

This command is used by the client to cleanly exit the server. All DICT servers MUST implement this command.

3.9.2. Responses

221 Closing Connection

This response code requires no special parameters.

3.10. The OPTION Command

3.10.1. OPTION MIME

OPTION MIME

3.10.1.1. Description

Requests that all text responses be prefaced by a MIME header [RFC2045] followed by a single blank line (CRLF).

If a client requests this option, then the client MUST be able to parse Content-Type and Content-transfer-encoding headers, and MUST be able to ignore textual responses which have an unsupported content or encoding. A client MUST support the UTF-8 encoding [RFC2044], which minimally means that the client MUST recognize UTF-8 multi-octet encodings and convert these into some symbol that can be printed by the client.

If a client requests this option, then the server will provide a MIME header. If the header is empty, the text response will start with a single blank line (CRLF), in which case a client MUST interpret this as a default header. The default header for SASL authentication is:

```
Content-type: application/octet-stream
Content-transfer-encoding: base64
```

The default header for all other textual responses is:

```
Content-type: text/plain; charset=utf-8
Content-transfer-encoding: 8bit
```

If OPTION MIME is not specified by the client, then the server may restrict the information content provided to the client. For example, a definition may be accompanied by an image and an audio clip, but the server cannot transmit this information unless the client is able to parse MIME format headers.

Note that, because of the line length restrictions and end-of-response semantics, the "binary" content-transfer-encoding MUST NOT be used. In the future, extensions to the protocol may be provided which allow a client to request binary encodings, but the default SHOULD always be that the client can look for a "CRLF . CRLF" sequence to locate the end of the current text response. This allows clients to easily skip over text responses which have unsupported types or encodings.

In the future, after significant experience with large databases in various languages has been gained, and after evaluating the need for specifying character sets and other encodings (e.g., compressed or BASE64 encoding), standard extensions to this protocol should be proposed which allow the client to request certain content types or encodings. Care should be taken that these extensions do not require a handshake which defeats pipelining. In the mean time, private extensions should be used to explore the parameter space to determine how best to implement these extensions.

OPTION MIME is a REQUIRED server capability, all DICT servers MUST implement this command.

3.10.1.2. Responses

```
250 ok (optional timing information here)
```

Note that some older server implementations, completed before this document was finalized, will return a status code 500 if this command is not implemented. Clients SHOULD be able to accept this behavior,

making default assumptions. Clients may also examine the capabilities string in the status code 220 header to determine if a server supports this capability.

3.11. The AUTH Command

AUTH username authentication-string

3.11.1. Description

The client can authenticate itself to the server using a username and password. The authentication-string will be computed as in the APOP protocol discussed in [RFC1939]. Briefly, the authentication-string is the MD5 checksum of the concatenation of the msg-id (obtained from the initial banner) and the "shared secret" that is stored in the server and client configuration files. Since the user does not have to type this shared secret when accessing the server, the shared secret can be an arbitrarily long passphrase. Because of the computational ease of computing the MD5 checksum, the shared secret should be significantly longer than a usual password.

Authentication may make more dictionary databases available for the current session. For example, there may be some publicly distributable databases available to all users, and other private databases available only to authenticated users. Or, a server may require authentication from all users to minimize resource utilization on the server machine.

Authentication is an optional server capability. The AUTH command MAY be implemented by a DICT server.

3.11.2. Responses

```
230 Authentication successful
531 Access denied, use "SHOW INFO" for server information
```

These response codes require no special parameters.

3.12. The SASLAUTH Command

SASLAUTH mechanism initial-response
SASLRESP response

3.12.1. Description

The Simple Authentication and Security Layer (SASL) is currently being developed [RFC2222]. The DICT protocol reserves the SASLAUTH and SASLRESP commands for this method of authentication. The results

of successful authentication with SALSAUTH will be the same as the results of successful AUTH authentication: more dictionary databases may become available for the current session.

The initial-response is an optional parameter for the SASLAUTH command, encoded using BASE64 encoding [RFC2045]. Some SASL mechanisms may allow the use of this parameter. If SASL authentication is supported by a DICT server, then this parameter MUST also be supported.

A typical SASL authentication will be initiated by the client using the SASLAUTH command. The server will reply with status code 130, followed by a challenge. The challenge will be followed by status code 330, indicating that the client must now send a response to the server.

Depending on the details of the SASL mechanism currently in use, the server will either continue the exchange using status code 130, a challenge, and status code 330; or the server will use status code 230 or 531 to indicate authentication was successful or has failed.

The challenges sent by the server are defined by the mechanisms as binary tokens of arbitrary length, and should be sent using a standard DICT textual response, as described in section 2.4.3. If OPTION MIME is not set, then BASE64 encoding MUST be used. If

OPTION MIME is set, then BASE64 is the default encoding, as specified in section 3.10.1.

The client will send all responses using the SASLRESP command and a BASE64-encoded parameter. The responses sent by the client are defined by the mechanisms as binary tokens of arbitrary length. Remember that DICT command lines may only be 1024 characters in length, so the response provided by a DICT client is limited.

If the mechanism specified in the SASLAUTH command is not supported, then status code 532 will be returned.

Authentication is an optional server capability. The SASLAUTH command MAY be implemented by a DICT server.

3.12.2. Responses

```
130 challenge follows
330 send response
230 Authentication successful
531 Access denied, use "SHOW INFO" for server information
532 Access denied, unknown mechanism
```

These response codes require no special parameters.

4. Command Pipelining

All DICT servers MUST be able to accept multiple commands in a single TCP send operation. Using a single TCP send operation for multiple commands can improved DICT performance significantly, especially in the face of high latency network links.

The possible implementation problems for a DICT server which would prevent command pipelining are similar to the problems that prevent pipelining in an SMTP server. These problems are discussed in detail in [RFC1854], which should be consulted by all DICT server implementors.

The main implication is that a DICT server implementation MUST NOT flush or otherwise lose the contents of the TCP input buffer under any circumstances whatsoever.

A DICT client may pipeline several commands and must check the responses to each command individually. If the server has shut down, it is possible that all of the commands will not be processed. For example, a simple DICT client may pipeline a CLIENT, DEFINE, and QUIT command sequence as it is connecting to the server. If the server is shut down, the initial response code sent by the server may be 420 (temporarily unavailable) instead of 220 (banner). In this case, the definition cannot be retrieved, and the client should report and error or retry the command. If the server is working, it may be able to send back the banner, definition, and termination message in a single TCP send operation.

5. URL Specification

The DICT URL scheme is used to refer to definitions or word lists available using the DICT protocol:

dict://<user>;<auth>@<host>:<port>/d:<word>:<database>:<n>
dict://<user>;<auth>@<host>:<port>/m:<word>:<database>:<strat>:<n>

The "/d" syntax specifies the DEFINE command (section 3.2), whereas the "/m" specifies the MATCH command (section 3.3).

Some or all of "<user>;<auth>@", ":<port>", "<database>", "<strat>", and "<n>" may be omitted.

"<n>" will usually be omitted, but when included, it specifies the nth definition or match of a word. A method for extracting exactly this information from the server is not available using the DICT protocol. However, a client using the URL specification could obtain all of the definitions or matches, and then select the one that is specified.

If "<user>;<auth>@" is omitted, no authentication is done. If ":<port>" is omitted, the default port (2628) SHOULD be used. If "<database>" is omitted, "!" SHOULD be used (see section 3.2). If "<strat>" is omitted, "." SHOULD be used (see section 3.3).

"<user>;<auth>@" specifies the username and the type of authentication performed. For "<auth>", the string "AUTH" indicates that APOP authentication using the AUTH command will be performed, whereas the string "SASLAUTH=<auth_type>" indicates that the SASLAUTH and SASLRESP commands will be used, with "<auth_type>" indicating the type of SASL authentication which will be used. If "<auth_type>" is a star (decimal code 42, "*"), then the client will select some type of authentication.

Whenever authentication is required, the client SHOULD request additional information (e.g., a passphrase) from the user. In contrast to [RFC1738], clear text passwords are not permitted in the URL.

Trailing colons may be omitted. For example, the following URLs might specify definitions or matches:

```
dict://dict.org/d:shortcake:
dict://dict.org/d:shortcake:*
dict://dict.org/d:shortcake:wordnet:
dict://dict.org/d:shortcake:wordnet:1
dict://dict.org/d:abcdefgh
dict://dict.org/d:sun
dict://dict.org/d:sun::1

dict://dict.org/m:sun
dict://dict.org/m:sun::soundex
dict://dict.org/m:sun:wordnet::1
dict://dict.org/m:sun::soundex:1
dict://dict.org/m:sun:::
```

6. Extensions

This protocol was designed so that flat text databases can be used with a server after a minimum of analysis and formatting. Our experience is that merely constructing an index for a database may be sufficient to make it useful with a DICT server. The ability to serve preformatted text is especially important since freely-available databases are often distributed as flat text files without any semantic mark-up information (and often contain "ASCII art" which precludes the automation of even simple formatting).

However, given a database with sufficient mark-up information, it may be possible to generate output in a variety of different formats (e.g., simple HTML or more sophisticated SGML). The specification of formatting is beyond the scope of this document. The requirements for negotiation of format (including character set and other encodings) is complex and should be examined over time as more experience is gained. We suggest that the use of different formats, as well as other server features, be explored as extensions to the protocol.

6.1. Experimental Command Syntax

Single-letter commands are reserved for debugging and testing, SHOULD NOT be defined in any future DICT protocol specification, and MUST NOT be used by any client software.

Commands beginning with the letter "X" are reserved for experimental extensions, and SHOULD NOT be defined in any future DICT protocol specification. Authors of client software should understand that these commands are not part of the DICT protocol and may not be available on all DICT servers.

6.2. Experimental Commands and Pipelining

Experimental commands should be designed so that a client can pipeline the experimental commands without knowing if a server supports the commands (e.g., instead of using feature negotiation). If the server does not support the commands, then a response code in the 5yz series (usually 500) will be given, notifying the client that the extension is not supported. Of course, depending on the complexity of the extensions added, feature negotiation may be necessary. To help minimize negotiation time, server-supported features may be announced in the banner (code 220) using the optional capabilities parameter.

7. Summary of Response Codes

Below is a summary of response codes. A star (*) in the first column indicates the response has defined arguments that must be provided.

```
  * 110 n databases present - text follows
  * 111 n strategies available - text follows
    112 database information follows
    113 help text follows
    114 server information follows
    130 challenge follows
  * 150 n definitions retrieved - definitions follow
  * 151 word database name - text follows
  * 152 n matches found - text follows
    210 (optional timing and statistical information here)
  * 220 text msg-id
    221 Closing Connection
    230 Authentication successful
    250 ok (optional timing information here)
    330 send response
    420 Server temporarily unavailable
    421 Server shutting down at operator request
    500 Syntax error, command not recognized
    501 Syntax error, illegal parameters
    502 Command not implemented
    503 Command parameter not implemented
    530 Access denied
    531 Access denied, use "SHOW INFO" for server information
    532 Access denied, unknown mechanism
    550 Invalid database, use "SHOW DB" for list of databases
    551 Invalid strategy, use "SHOW STRAT" for a list of strategies
    552 No match
    554 No databases present
    555 No strategies available
```

8. Sample Conversations

Theses are samples of the conversations that might be expected with a typical DICT server. The notation "C:" indicates commands set by the client, and "S:" indicates responses sent by the server. Blank lines are included for clarity and do not indicate actual newlines in the transaction.

8.1. Sample 1 - HELP, DEFINE, and QUIT commands

```
C: [ client initiates connection ]

S: 220 dict.org dictd (version 0.9) <27831.860032493@dict.org>


C: HELP

S: 113 Help text follows
S: DEFINE database word            look up word in database
S: MATCH database strategy word    match word in database using strategy
S: [ more server-dependent help text ]
S: .
S: 250 Command complete


C: DEFINE ! penguin

S: 150 1 definitions found: list follows
S: 151 "penguin" wn "WordNet 1.5" : definition text follows
S: penguin
S:   1. n: short-legged flightless birds of cold southern esp. Antarctic
S:      regions having webbed feet and wings modified as flippers
S: .
S: 250 Command complete


C: DEFINE * shortcake

S: 150 2 definitions found: list follows
S: 151 "shortcake" wn "WordNet 1.5" : text follows
S: shortcake
S:   1. n: very short biscuit spread with sweetened fruit and usu.
S:      whipped cream
S: .
S: 151 "Shortcake" web1913 "Webster's Dictionary (1913)" : text follows
S: Shortcake
S:    \Short"cake'\, n.
S:    An unsweetened breakfast cake shortened with butter or lard,
S:    rolled thin, and baked.
S: .
S: 250 Command complete


C: DEFINE abcdefgh

S: 552 No match
```

```
C: quit

S: 221 Closing connection
```

8.2. Sample 2 - SHOW commands, MATCH command

```
C: SHOW DB

S: 110 3 databases present: list follows
S: wn "WordNet 1.5"
S: foldoc "Free On-Line Dictionary of Computing"
S: jargon "Hacker Jargon File"
S: .
S: 250 Command complete


C: SHOW STRAT

S: 111 5 strategies present: list follows
S: exact "Match words exactly"
S: prefix "Match word prefixes"
S: substring "Match substrings anywhere in word"
S: regex "Match using regular expressions"
S: reverse "Match words given definition keywords"
S: .
S: 250 Command complete


C: MATCH foldoc regex "s.si"

S: 152 7 matches found: list follows
S: foldoc Fast SCSI
S: foldoc SCSI
S: foldoc SCSI-1
S: foldoc SCSI-2
S: foldoc SCSI-3
S: foldoc Ultra-SCSI
S: foldoc Wide SCSI
S: .
S: 250 Command complete


C: MATCH wn substring "abcdefgh"

S: 552 No match
```

8.3. Sample 3 - Server downtime

```
C: [ client initiates connection ]

S: 420 Server temporarily unavailable

C: [ client initiates connection ]

S: 421 Server shutting down at operator request
```

8.4. Sample 4 - Authentication

```
C: [ client initiates connection ]

S: 220 dict.org dictd (version 0.9) <27831.860032493@dict.org>

C: SHOW DB

S: 110 1 database present: list follows
S: free "Free database"
S: .
S: 250 Command complete

C: AUTH joesmith authentication-string

S: 230 Authentication successful

C: SHOW DB

S: 110 2 databases present: list follows
S: free "Free database"
S: licensed "Local licensed database"
S: .
S: 250 Command complete
```

9. Security Considerations

This RFC raises no security issues.

10. References

[ASCII] US-ASCII. Coded Character Set - 7-Bit American Standard Code for Information Interchange. Standard ANSI X3.4-1986, ANSI, 1986.

[FOLDOC] Howe, Denis, ed. The Free On-Line Dictionary of Computing, <URL:http://wombat.doc.ic.ac.uk/>

[ISO10646] ISO/IEC 10646-1:1993. International Standard -- Information technology -- Universal Multiple-Octet Coded Character Set (UCS) -- Part 1: Architecture and Basic Multilingual Plane. UTF-8 is described in Annex R, adopted but not yet published. UTF-16 is described in Annex Q, adopted but not yet published.

[JARGON] The on-line hacker Jargon File, version 4.0.0, 25 JUL 1996, <URL:http://www.ccil.org/jargon/>

[KNUTH73] Knuth, Donald E. "The Art of Computer Programming", Volume 3: Sorting and Searching (Addison-Wesley Publishing Co., 1973, pages 391 and 392). Knuth notes that the soundex method was originally described by Margaret K. Odell and Robert C. Russell [US Patents 1261167 (1918) and 1435663 (1922)].

[PZ85] Pollock, Joseph J. and Zamora, Antonio, "Automatic spelling correction in scientific and scholarly text," CACM, 27(4): Apr. 1985, 358-368.

[RFC640] Postel, J., "Revised FTP Reply Codes", RFC 640, June, 1975.

[RFC821] Postel, J., "Simple Mail Transfer Protocol", STD 10, RFC 821, August 1982.

[RFC822] Crocker, D., "Standard for the Format of ARPA Internet Text Messages", STD 11, RFC 822, August 1982.

[RFC977] Kantor, B., and P. Lapsley, "Network News Transfer Protocol: A Proposed Standard for the Stream-Based Transmission of News", RFC 977, February 1986.

[RFC2045] Freed, N., and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996.

[RFC1738] Berners-Lee, T., Masinter, L. and M. McCahill, "Uniform Resource Locators (URL)", RFC 1738, December 1994.

[RFC1760] Haller, N., "The S/KEY One-Time Password System", RFC 1760, February 1995.

[RFC1985] Freed, N., and A. Cargille, "SMTP Service Extension for Command Pipelining", RFC 1854, October 1995.

[RFC1939] Myers, J., and M. Rose, "Post Office Protocol - Version 3", STD 53, RFC 1939, May 1996.

[RFC2044] Yergeau, F., "UTF-8, a transformation format of Unicode and ISO 10646", RFC 2044, October 1996.

[RFC2068] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2068, January 1997.

[RFC2078] Linn, J., "Generic Security Service Application Program Interface, Version 2", RFC 2078, January 1997.

[RFC2222] Myers, J., "Simple Authentication and Security Layer (SASL)", RFC 2222, October 1997.

[WEB1913] Webster's Revised Unabridged Dictionary (G & C. Merriam Co., 1913, edited by Noah Porter). Online version prepared by MICRA, Inc., Plainfield, N.J. and edited by Patrick Cassidy <cassidy@micra.com>. For further information, see <URL:ftp://uiarchive.cso.uiuc.edu/pub/etext/gutenberg/etext96/pgw*>, and <URL:http://humanities.uchicago.edu/forms_unrest/webster.form.html>

[WORDNET] Miller, G.A. (1990), ed. WordNet: An On-Line Lexical Database. International Journal of Lexicography. Volume 3, Number 4. <URL:http://www.cogsci.princeton.edu/~wn/>

11. Acknowledgements

Thanks to Arnt Gulbrandsen and Nicolai Langfeldt for many helpful discussions. Thanks to Bennet Yee, Doug Hoffman, Kevin Martin, and Jay Kominek for extensive testing and feedback on the initial implementations of the DICT server. Thanks to Zhong Shao for advice and support.

Thanks to Brian Kanto, Phil Lapsley, and Jon Postel for writing exemplary RFCs which were consulted during the preparation of this document.

Thanks to Harald T. Alvestrand, whose comments helped improve this document.

12. Authors' Addresses

Rickard E. Faith
EMail: faith@cs.unc.edu (or faith@acm.org)

Bret Martin
EMail: bamartin@miranda.org

The majority of this work was completed while Bret Martin was a student at Yale University.

13. Full Copyright Statement