Network Working Group
Request for Comments: 2259
Category: Informational

J. Elliott
Epic Systems Corporation
J. Ordille
Bell Labs, Lucent Technologies

January 1998

Simple Nomenclator Query Protocol (SNQP)

Status of this Memo

Copyright Notice

Abstract

The Simple Nomenclator Query Protocol (SNQP) allows a client to communicate with a descriptive name service or other relational-style query service. The protocol is useful to services that search many data repositories for query responses. Clients can pose queries on relations, list descriptions of relations, and obtain advice on reducing the search time and cost of their queries. Clients are informed of the age of information in caches, and may request more recent information. SNQP provides support for graphical user interfaces. It also supports different types of comparison operators, so services can use SNQP with a variety of back-end servers, e.g. relational database servers, CCSO servers, and servers providing relational views of X.500.

SNQP is an ASCII protocol in the request-reply style of SMTP. It was specifically designed for use with the Nomenclator name and information service, and has been useful elsewhere.

1. Introduction

The Simple Nomenclator Query Protocol (SNQP) is a protocol for querying servers that search collections of data repositories. Users retrieve information from an SNQP server by describing attributes of the information. SNQP servers contact one or many data repositories to retrieve the response to a user query. If the data repositories

differ in protocol or data format, it is responsibility of the SNQP server to translate protocols and data formats to provide one, integrated answer to the user's query.

SNQP servers share the protocol needs of centralized data repositories that answer queries with locally stored data. SNQP servers also require specialized protocol features due to their distributed search characteristics.

In highly distributed environments, it is unreasonable to expect all data repositories that need to be searched to be available when queries are posed. SNQP servers require facilities for returning partial results in the presence of communications errors with data repositories. The partial results must indicate how to resubmit the query only to those data repositories that are unavailable.

In addition, users may pose queries without realizing the cost of the search for query responses. SNQP provides facilities for informing users of query costs and advising them on limiting that cost. Costs and advice are returned before queries are executed.

Finally, SNQP servers may cache data and meta-data to speed query responses. Servers can inform users of the t-bound for their query response. A t-bound is the time after which changes may have occurred to the data that are not reflected in the query response [6,2]. A t-bound is the time of the oldest cache entry used to calculate the response. Users can request that query responses are more current then a particular t-bound. Making such a request flushes older items from the cache.

SNQP provides support for graphical user interfaces. It also supports different types of comparison operators, so SNQP servers can query a variety of back-end data repositories, e.g. relational databases, CCSO servers [3], and servers providing relational views of X.500 [10].

SNQP is a connection-oriented protocol. A client initiates a query session with an SNQP server by making a TCP connection to a well-known port. The client then executes a series of SNQP commands. These commands are listed briefly in Table 1. Section 2 provides some typical scenarios for using these commands, and Section 3 describes the commands fully. The server replies to each command using the theory of reply codes described for the Simple Mail Transfer Protocol (SMTP) [9]. The theory of reply codes and the defined reply codes are described in Section 4.

Command	Description
advice	Provide advice on query costs without executing query.
attributes	List the attributes for a relation.
compare	Set type of comparison operation.
help	Explain the SNQP commands.
imagui	Format replies for a graphical user interface.
next	Stop processing current query, continue with next query in block.
noadvice	Provide responses to queries. Do not advise on costs.
noimagui	Format replies for people.
query	Submit a block of one or more SQL query statements.
relations	List the relations available through the SNQP server.
stop	End processing of current query, and cancel any queries remaining in block.
quit	Terminate the query session.

Table 1: SNQP Commands

SNQP queries are posed in SQL, a standard relational database query language [4,12]. Information that is obtained through SNQP servers is organized by type into database relations. SQL queries may often have more functionality then a server supports or an application demands. Moreover, advice on query costs, some types of comparison operations or t-bounds may not be supported by a particular server. SNQP defines a minimal subset of functionality for a working SNQP protocol. Functionality beyond this subset is optional. Servers that do not support optional functionality must return replies that indicate this to the user. The required and optional features of SNQP are summarized in Section 5.

SNQP was specifically designed for use with the Nomenclator name and information service [8,7,5]. Nomenclator produces query responses by integrating information from data repositories with different protocols and data formats. It constrains the searches for query responses through a variety of distributed indexing techniques. SNQP has also been found useful elsewhere, even as a query language for a single data repository.

SNQP is defined for US-ASCII only, and use with other character sets will require further work.

Section 6 concludes this document with a description of security considerations.

2. Scenarios

This section illustrates the basic SNQP commands by presenting several client scenarios. The scenarios include a new user, a user who prefers CCSO style comparisons and more current responses, a graphical user interface program, a user with a change of mind, and a user worried about costs. Although SNQP will work for a human client on a bare connection (like one provided by telnet), it also works for client programs. Several of these programs have been written and provide enhanced interfaces.

2.1 New User

A new SNQP user will first make a tcp connection to an SNQP server. For purposes of illustration, we will assume that the user makes the connection with the Unix telnet command, and that the server is located at nomen.research.bell-labs.com on port 4224. The user enters a relation command to discover what relations are available, and an attributes command to discover the attributes for a particular relation. The user eventually asks for people with a given name of "J*" and a surname of "Ordille" who work for "Lucent Tech*". The response is current through June 11, 1996 at 11 p.m. EDT. Figure 1a and Figure 1b provide this scenario.

```
> telnet nomen.research.bell-labs.com 4224
Trying 135.104.70.9...
Connected to nomen.research.bell-labs.com.
Escape character is '^]'.
220 nomen.research.bell-labs.com Nomenclator Query Service ready

relations
211-There is 1 relation defined:
211 People

attributes People
212-There are 20 attributes in relation "People":
212-Given_Name
212-Middle_Name
212-Surname
212-Name_Suffix
212-Title
212-Organization
212-Division
212-Department
212-Building
212-Street
212-City
212-State_or_Province
212-Postal_Code
212-Country
212-Phone
212-Fax
212-Email
212-MHSmail
212-Last_Modified
212 Source
```

Figure 1a: New User Queries Server

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

      query
      350 Send the query text, end with .

      select * from People where
             given_name = "J*" and surname = "Ordille" and
             organization = "Lucent Tech*";
      .
      351 Partial response follows, ended with .

      Given_Name: Joann
      Middle_Name: J.
      Surname: Ordille
      Title: MTS
      Organization: Lucent Technologies
      Division: Bell Laboratories
      Department: Computing Sciences Research Center
      Building: 2C-301
      Street: 700 Mountain Avenue
      City: Murray Hill
      State_or_Province: New Jersey
      Postal_Code: 07974
      Country: United States
      Phone: +1 908 582 7114
      Email: joann@bell-labs.com
      Source: nomen://bell-labs.com:17036/email=joann@bell-labs.com

      .
      250 All queries processed.  Current through 11-Jun-1996 23:00 EDT.

      quit
      221 nomen.research.bell-labs.com closing transmission channel

      Connection closed by foreign host.
```

Figure 1b: New User Queries Server (continued)

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

2.2 User with CCSO and Currentness Preferences

A user who is accustomed to CCSO name servers prefers CCSO word-based matching within attribute strings. Each word in the query string for an attribute must appear in some order in the response string. The wildcard "*" matches any substring within a word. The default

matching, illustrated in Figure 1b, is exact matching of a query string. The query string may include "*" wildcards which match any substring within the response string. Both types of matching are case insensitive.

In Figure 2, the CCSO-style user connects to the SNQP server, enables csso matching, and requests some information about Ordille who works in research at a lab division of some company. The request asks for information that is more current than June 11, 1996 at 11 p.m. if it is available.

```
compare ccso
213 Performing ccso equality comparisons

query 11-Jun-1996 23:00
350 Send the query text, end with .

select given_name, surname, organization, division, department,
       email from People
       where surname = "Ordille" and department = "research"
       and division = "lab*";
.

351 Partial response follows, ended with .

Given_Name: Joann
Surname: Ordille
Organization: Lucent Technologies
Division: Bell Laboratories
Department: Computing Sciences Research Center
Email: joann@bell-labs.com

.
250 All queries processed.  Current through 12-Jun-1996 22:35 EDT.
```

Figure 2: User with CCSO Preferences Queries Server

2.3 Graphical User Interface Program

A user designs a Windows program as a front end to the SNQP server. In Figure 3, the program requests replies formatted for a graphical user interface program. The program submits two SQL queries, and

receives detailed responses that indicate the type and position of errors. The error messages are discussed in more detail in Section 3.

```
imagui
214 GUI responses enabled

query
350 Send the query text, end with .

select * from Peple where name = "Elliott";
.
735 00000001a000015 e Unknown relation, "Peple"

735 00000001a000027 e Attribute "name" not found in any relation used.


250 All queries processed.  Current through 12-Jun-1996 22:35 EDT.

query
350 Send the query text, end with .

select * from People wher surname = "Elliott";
.
730 00000001a000022 e syntax error

730 00000001a000027 e syntax error

730 00000001a000037 e syntax error

730 00000001a000039 e syntax error


250 All queries processed
```

Figure 3: Graphical User Interface Program Queries Server

2.4 User Changes Mind

An exuberant user decides to search everywhere for family members, then look up a friend who works at Epic Systems, and finally search everywhere for an old school friend. Once the query set starts, the user realizes the folly of searching everywhere, stops the first

query, executes the second query and then stops executing the query block. This scenario is illustrated in Figure 4. The t-bound is represented by <time> in this scenario due to space restrictions.

```
query
350 Send the query text, end with .

select * from people where surname = "Smith";
select given_name, surname, email from people
       where surname = "Elliott"
       and organization = "Epic Systems*";
select * from people where surname = "Brown";
.
next

352 Starting next query.  Any pending responses discarded.

351 Partial response follows, ended with .

Given_Name: Jim
Surname: Elliott
Email: jim@apocalypse.com

.
352 Beginning next query.  Previous current through <time>.

stop

251 All pending queries and responses discarded
```

Figure 4: User Changes Mind About Submitted Queries

2.5 User Worries About Costs

In Figure 5a, the exuberant user decides to apply more caution, and asks for advice on searching for a friend named "Susan Brown". The user can not recall the name of the organization where Susan works, but remembers that the state name begins with "I". The advice response lists the locations of the data repositories that will be contacted. These locations can be supplied to the SNQP server using the "source" attribute. Each location is followed by a blank and a descriptive phrase for the data repository. Continuing in Figure 5b, the SNQP server also supplies a list of attributes that may constrain

the query further. The user recognizes the name Northeastern, and submits the query directly to that location. The user could also have added "organization = "Northeastern*"" to the original query. Other advice options are described in Section 3.

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

   advice
   214 Basic advice enabled.  Query responses disabled.

   query
   350 Send the query text, end with .

   select * from people where surname = "Brown" and
                              given_name = "Susan" and
                              state_or_province = "I*";
   .

   354 The query will contact 8 data repositories, ended with .
   ccso://ns.dacc.cc.il.us:105/* Danville Area Community College
   ccso://ns.eiu.bgu.edu:105/* Eastern Illinois University
   ccso://ns.ilstu.edu:105/* Illinois State University
   ccso://ns.imsa.edu:105/* Illinois Math and Science Academy
   ccso://ns.ne.edu:105/* Northeastern Illinois University
   ccso://ns.uiuc.edu:105/* University of Illinois at Urbana-Champaign
   ccso://ns.iup.edu:105/* Indiana University of Pennsylvania
   ccso://ph.indstate.edu:105/* Indiana State University
   .
```

Figure 5a: User Asks About Costs Before Executing Query

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

      355 There are 8 attributes that may constrain the query, ended with .
      Organization
      Department
      Email
      State_or_Province
      Country
      Postal_Code
      Phone
      Source
      .

      250 All queries processed.  Current through 12-Jun-1996 22:35 EDT.

      noadvice
      214 Query responses returned.  Advice disabled.

      query
      350 Send the query text, end with .

      select * from people where surname = "Brown" and
                               given_name = "Susan" and
                                  state_or_province = "I*" and
                                  source ="ccso://ns.neiu.bgu.edu:105/*";
      .

      351 Partial response follows, ended with .

      Given_Name: Susan
      Middle_Name: W.
      Surname: Brown
      Organization: Northeastern Illinois University
      Email: sw-brown@ne.edu
      Source: ccso://ne.edu:105/alias=SW-Brown

      .
      250 All queries processed.  Current through 12-Jun-1996 22:35 EDT.

          Figure 5b: User Asks About Costs Before Executing Query

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

3. Commands

SNQP commands are case insensitive and terminated with a newline <LF> or carriage return <CR>. In the following descriptions, SNQP commands are in upper case and SNQP replies are in mixed case. Items in a command list are separated by blanks.

Most SNQP replies are short. They have a rely code (see Section 4), followed by a continuation character and reply text. If the continuation character is blank, the reply is complete. If the continuation character is a dash ("-"), the reply continues on the next line. Text within the reply can vary, but the reply code remains the same. A two line reply example is given below:

```
nnn-Message1
nnn Message2
```

In some cases commands or replies may be long, so these commands/replies use the '.'-terminated block structure that is used for message bodies in SMTP. Blocks are comprised of lines of text that constitute the command/reply. Blocks are terminated with a period on a line by itself.

SNQP generally ignores blank lines in both directions, except that blanks lines separate tuples within query response blocks.

Whenever a time is listed in a command or response, it has the format:

```
DD-MMM-YYYY HH:MM ZZZ
```

---

where DD is the day, MMM are the first three characters of the month, YYYY is the year, HH the hours on a 24 hour clock, MM the minutes, and ZZZ the commonly used US timezone abbreviations. If time zone is unspecified in a command, the timezone of the SNQP server is assumed.

SNQP servers support a source attribute in every relation. In queries, the source attribute directs the SNQP server to a particular data repository. In query responses, the source attribute indicates the origin of the information in a tuple. In advice and error

messages, the source attribute is provided so the client can contact the source in later queries. The source attribute has two possible forms:

<protocol>://<domain-name>:<port>

<protocol>://<domain-name>:<port>/<tuple-id>

<protocol> identifies the protocol used to contact the data repository. The data repository can be (was) contacted at <domain-name> and <port>. When present, <tuple-id> identifies a specific entry in the data repository. It is missing when the data repository does not have an attribute that uniquely identifies its entries. Although the source string is similar to a URL, the protocols listed may or may not be supported by World-Wide Web browsers. An effort should be made to keep the protocol identifiers consistent with accepted standards, but in the end they are specific to SNQP servers.

When a connection is established with an SNQP server, the server returns the following greeting where <domain-name> is the domain name of the server host, e.g. nomen.research.bell-labs.com, and <service-name> is the name of the service, e.g. Nomenclator:

---

220 <domain-name> <snqp-service-name> Query Service ready

The following sections describe each command in detail. The commands are ordered alphabetically. Typical reply messages are explained with each command. Exceptional error conditions, for example system errors or rejection of connections due to load, may sometimes occur. These error replies are documented in Section 4.

3.1 Advice

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

   ADVICE
   214 Basic advice enabled. Query responses disabled.
   514 Advice not available

   ADVICE <RELATION> <ATTRIBUTE>
   214 Advice enabled for "<attribute>" in "<relation>"
   553 Unknown relation
   554 Unknown attribute
   514 Advice not available for "<attribute>"

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

In all cases, advice disables query searches. When queries are submitted, advice is returned about the cost of the query or ways of constraining the query further. There are two forms of the advice command.

The first form of command does not include an attribute name. When an SQL query is processed, the SNQP server returns a list of data repositories that it will contact. It also returns a list of attributes that may constrain the query further. The specific values of the attributes will determine whether the query is constrained further. If advice is not available from the server, an error is returned.

The second form of advice includes the name of a relation and the name of an attribute in that relation. SQL queries return a list of possible values for the attribute. The list may be complete, or may only include values that are known to constrain the search. This distinction is described further in the query command. If advice is not available on the attribute or the relation or attribute is unknown, an error is returned. When advice is not available on an attribute, basic advice and advice on other attributes may be available.

Basic advice and advice for one or more attributes can be enabled simultaneously. They are not mutually exclusive.

The advice command is useful to application programs which present lists of alternatives to the user. A query-form program can enable advice for an attribute, submit an empty query, and obtain the list of options for the attribute. The list will indicate whether it is a full list of all values for the attribute, or a constraint list of

only those values known to constrain queries. The program can use full lists to create a selection menu on its query form. A program can also enable basic advice, submit the query, and then ask the user to select the data repositories to search from the resulting list.

3.2 Attributes

```
ATTRIBUTES <RELATION> <TIME>
212-There are <n> attributes in relation "<RELATION>":
212-<Attribute-name>
212-<Attribute-name>
212 Current through <TIME>

553 Unknown relation.  Current through <TIME>.
556 T-bounds not supported
```

The attributes command lists the attributes defined for the given relation. Since characteristics of relations may be defined outside the SNQP server and cached there, the user may ask for an answer that is more recent than <TIME>. The SNQP server will endeavor to provide this information. The first line of the reply notes the number of attributes <n>. Subsequent lines list the attribute names. The information in the response is current through the time returned, but may have changed after that time. Accepting requests to improve a t-bound and indicating the t-bound of the result are optional for SNQP servers.

If the relation is unknown, an error is returned. If <TIME> is submitted when t-bounds are not supported, an error is returned.

3.3 Compare

```
COMPARE <COMPARISON-TYPE>
213 Performing <COMPARISON-TYPE> comparisons

555 Unknown comparison type
```

The compare command lists the type of equality comparison performed for SQL queries. The compare command can be followed by a comparison type to set the type. Reply 555 is returned if the comparison type

is unknown or unsupported. "Default" and "CCSO" are defined comparison types. The default equality comparison is exact string matching. The query string may include "*" wildcards which match any substring within the response string. The CCSO equality comparison matches words within strings. Each word in the query string for an attribute must appear in some order in the response string. Words are delimited by blank, comma, colon, semi-colon, tab, and newline. The wildcard "*" matches any substring within a word. Both string and word comparisons are case insensitive.

3.4 Help

```
HELP
210-The following commands are available:
210-<comma-separated-command-list>
210 <comma-separated-command-list>

HELP <COMMAND>
210-<explanation of <COMMAND>>
210 <explanation of <COMMAND>>

500 Sorry, no help available for "<COMMAND>"
```

The help command returns the list of available commands. If some commands are not supported, for example advice, they should not be listed. Use of unsupported commands should still return an informative error message. Help can be followed by a command name for information on that command. If no help is available for a command or the command does not exist, Reply 500 is returned.

3.5 Imagui

```
IMAGUI
215 GUI responses enabled
```

The imagui command informs the server that the client is a graphical user interface (GUI). The client requests more comprehensive, program-oriented errors and progress reports. It replies that GUI responses are enabled. See Section 4 for more information on GUI responses.

3.6 Next

NEXT
353 Starting next query. Any pending responses discarded.

450 No query in progress

The next command stops processing of the current SQL query. It starts the next SQL query in the block submitted with the last query command. If none remain, the query command is completed. An error is returned if no query is in progress.

3.7 Noadvice

NOADVICE
216 Query responses enabled. Advice disabled.

The noadvice command disables advice for query commands. It activates query searches, so queries will return responses. See the advice command for more information.

3.8 Noimagui

NOIMAGUI
215 GUI responses disabled

The noimagui command disables detailed messages to a graphical user interface program. It replies that GUI responses are disabled. See Section 4 for more information on GUI responses.

3.9 Query

The query command behaves differently depending on whether responses or advice are enabled. We first describe the submission of a query and the possible immediate error responses. We then describe the

possible replies to the query command when responses are enabled. We finish by describing the possible replies to the query command when advice is enabled.

QUERY <TIME>
350 Send query text, end with .

450 Query already in progress
552 Query blocks are limited to one SQL query
556 T-bounds not supported

The query command submits a block of SQL queries to the SNQP server. Each SQL query must be terminated with a semi-colon, and the entire block is terminated with a line containing a single period. Special characters in query string constants can be included using the C language conventions, e.g. "\n" is the newline character.

Since a variety of cached information can be used in processing the SQL queries, the user may ask for answers that are more recent than <TIME>. The SNQP server will endeavor to provide this information. Accepting requests to improve a t-bound is optional for SNQP servers.

If a query command is already in process, the entire block is refused. If multiple SQL queries are submitted in one block to a server that does not support multi-query blocks, an error is returned. If <TIME> is submitted when t-bounds are not supported, an error is returned.

---

351 Partial response follows ended with .
352 Beginning next query. Previous query current through <TIME>.
250 All queries processed. Current through <TIME>.

653 <Communications err> with <location> <location description>
660 <Error> from <location> <location description>

700 <SQL query parsing error>
750 <SQL query semantic error>
761 <Requirements Error> for <location> <location description>

Responses are returned in blocks as they arrive from data repositories. Reply 351 begins a response block. Response blocks are terminated with periods. Tuples are sent within the block as a list of attribute name and value pairs:

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

<attribute-name>: <attribute-value>
                : <attribute-value>

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

Only the first line of a multi-line attribute returns the <attribute-name>. Successive tuples are separated with blank lines. Attributes with null or blank values are suppressed at the option of the SNQP server.

In between response blocks, error replies can be reported. Replies 653, 660 and 761 are examples of such errors. Reply 653 reports a communication error with the data repository identified by the source location and described by the associated string. Reply 660 reports an error returned by a data repository. Reply 761 reports a known requirement of the data repository that the query failed to satisfy. Reply 761 reflects comparison of the query with known characteristics of the data repository by the SNQP server. For example, some data repositories refuse queries that do not contain a specific subset of attributes in the relation. Other replies are possible. It is best to check the the type and severity of the reply against the theory of reply codes in Section 4.

When an SQL query in a block is successfully completed, the SNQP server sends Reply 352 to indicate that the next query is being started. Reply 352 reports the t-bound of the previous query if it is available. Reply 352 is sent even if the previous query terminated due to permanent errors. The one exception is that permanent errors generated by the next or stop command supersede Reply 352.

When all SQL queries are complete, the SNQP server sends Reply 250 to indicate that all queries have been processed. A query block containing one query that has no responses will only return Reply 250. Reply 250 reports the t-bound of the last SQL query in the block if it is available. Reply 250 is sent even if the last query in the block terminated due to permanent errors. The one exception is that permanent errors generated by the stop command supersede Reply 352.

Note that this command follows the convention that "intermediate" reply codes, as defined in Section 4, are used until the SQL query is complete. Final query completion error codes abort the processing of the SQL query. Examples of these errors include parsing errors (Reply 700) and semantic errors (Reply 750) in the SQL query. The SNQP server will attempt to continue with the next query if possible. The block of queries will be terminated with Reply 250 or 251 (from the stop command) to indicate that another query will be accepted. Indicating the t-bound of a query response is optional for SNQP servers.

354-The query will contact <n> data repositories, ended with .
<location> <location-description>

355-There are <n> attributes that may constrain the query, ended with
.
<attribute-name>

356-There are <n> possible values for "<attribute>", ended with .
<attribute-value>

357-There are <n> constraining values for "<attribute>", ended with .
357-<attribute-value>
357 <attribute-value>

352 Beginning next query. Previous query current through <TIME>.
250 All queries processed. Current through <TIME>.

700 <SQL query parsing error>
750 <SQL query semantic error>

Different kinds of advice are returned in different blocks. Basic advice about the number, n, of data repositories that will be searched is returned with Reply 354. Subsequent lines list location and location description for the data repositories that will be searched. The data repository locations can be supplied to the SNQP server using the "source" attribute. Each location is followed by a blank and a descriptive phrase for the data repository.

Basic advice about the attributes that may constrain the query is returned with reply code 355. The first line of the reply includes the number, n, of attributes. Subsequent lines list the names of the attributes. The specific values of the attribute will determine whether the query is constrained further.

Advice for a particular attribute has two forms. First, the advice may list the n possible values for the attribute in reply 356. The list is complete; no other values for the attribute exist within the context of the query. Second, the advice may list the n values for the attribute that are known to constrain the query. The list is incomplete; other values of the attribute may exist within the context of the query.

When advice for an SQL query in a block is successfully completed, the SNQP server sends reply 352 to indicate that the next query is being started. Reply 352 reports the t-bound of the advice for the previous query if it is available. Reply 352 is sent even if the previous query terminated due to permanent errors. The one exception is that permanent errors generated by the next or stop command supersede Reply 352.

When all SQL queries are complete, the SNQP server sends Reply 250 to indicate that all queries have been processed. Reply 250 reports the t-bound of the last SQL query in the block if it is available. Reply 250 is sent even if the last query in the block terminated due to permanent errors. The one exception is that permanent errors generated by the stop command supersede Reply 352.

Final query completion error codes abort the processing of the SQL query. Examples of these errors include parsing errors and semantic errors in the SQL query. The SNQP server will attempt to continue with the next query if possible. The block of queries will be terminated with Reply 250 or 251 (from the stop command) to indicate that another query will be accepted. Indicating the t-bound of advice is optional for SNQP servers.

3.10 Relations

```
RELATIONS <TIME>
211-There are <n> relations defined:
211-<Relation-name>
211-<Relation-name>
211 Current through <TIME>.

556 T-bounds not supported
```

The relations command lists the currently available relation names. Since characteristics of relations can be cached, the user may ask for an answer that is more recent than <TIME>. The SNQP server will

endeavor to provide this information. The first line of the reply notes the number of relations <n>. Subsequent lines list the relation names. The information in the response is current through the time returned, but may have changed after that time. Accepting requests to improve a t-bound and indicating the t-bound of the result are optional for SNQP servers.

If <TIME> is submitted when t-bounds are not supported, an error is returned.

3.11 Stop

STOP
251 All pending queries and responses discarded

450 No query in progress

The stop command ends processing of the current SQL query, and cancels any that may have followed it in the last query command. An error is returned if no queries are in progress.

3.12 Quit

QUIT
221 <domain-name> closing transmission channel

The quit command ends the session. It closes the TCP connection after signing off with the domain name of the SNQP server.

4. Replies

Most SNQP replies are short. They have a rely code followed by a continuation character and reply text. If the continuation character is blank, the reply is complete. If the continuation character is a dash ("-"), the reply continues on the next line. Text within the reply can vary, but the reply code remains the same. A two line reply example is given below:

```
   nnn-Message1
   nnn Message2
```

In some cases commands or replies may be long, so these commands/replies use the '.'-terminated block structure that is used for message bodies in SMTP. Blocks are comprised of lines of text that constitute the command/reply. Blocks are terminated with a period on a line by itself.

The theory of reply codes explained for SMTP in RFC-821 is used here. Table 2 defines the reply code structure. Reply codes are three digits, xyz. The x digit indicates the command status. The y digit indicates the component of the system that generated the reply. The z digit allows for further distinctions within replies from the same component.

Code	Interpretation
1yz	Positive preliminary reply (not used in SNQP)
2yz	Positive completion reply
3yz	Positive intermediate reply
4yz	Transient negative completion reply
5yz	Permanent negative completion reply
6yx	Transient negative intermediate reply
7yx	Permanent negative intermediate reply
x0z	Syntax or semantic problem
x1z	Informational reply
x2z	Related to transmission channel
x3z	Formatted (location coded) report for GUI
x4z	Status message to be displayed by GUI
x5z	Related to query resolver
x6z	Related to data repository
x9z	Component generating the error is unknown or suspect

Table 2: Reply Code Structure

The GUI-related reply codes are only used if the server has been informed that it is communicating with a graphical user interface, via the imagui command. For such codes in the x3z space, digit "z" takes on the role of digit "y" in other codes. I.e. 735 are permanent negative intermediate replies about the query resolver.

Table 3a and Table 3b list the defined regular (non-GUI) reply codes. Text messages for the reply codes may vary. The codes are sorted numerically.

```
210-The following commands are available:
211-There are <n> relations defined:
212-There are <n> attributes in relation "<relation>":
213 Performing <comparison-type> type equality comparisons
214 Basic advice enabled.  Query responses disabled.
214 Advice enabled for "<attribute>" in "<relation>"
215 GUI responses enabled
215 GUI responses disabled
216 Query responses enabled.  Advice disabled.

220 <domain-name> <snqp-service-name> Query Service ready
221 <domain-name> closing transmission channel

250 All queries processed
250 All queries processed.  Current through <time>.
251 All pending queries and responses discarded

340 Searching <n> data repositories
350 Send the query text, end with .
351 Partial response follows, ended with .
352 Beginning next query in batch
352 Beginning next query in batch.  Previous current through <time>.
353 Starting next query. Any pending responses discarded.
354 The query will contact <n> data repositories, ended with .
355 There are <n> attributes that may constrain the query, ended with
.
356 There are <n> possible values for attribute "<attribute>":
357 There are <n> constraining values for attribute "<attribute>":

420 Too many connections in progress. Try later.
421 Error in communicating with <snqp-service-name>
450 No query in progress
451 Cancel ignored
```

Table 3a: Reply Codes

---

450 Query already in progress
490 Internal error: Invalid query reference number
491 System error: <error number or message>
492 Internal error: Out of client table space
499 <snqp-service-name> shutting down
500 Sorry, no help is available for "<command>"
501 Unknown command
502 Too many arguments for this command
502 Not enough arguments for this command
514 Advice not available
514 Advice not available on <attribute>
552 Query blocks are limited to one SQL query
553 Unknown relation
553 Unknown relation. Current through <TIME>.
554 Unknown attribute
555 Unknown comparison type
556 T-bounds not supported
557 Will not list more than <n> data repositories
557 Will not list more than <n> attribute values
557 Will not list more than <n> responses
557 Too many data repositories to list
557 Too many attribute values to list
557 Too many responses to list
557 Too many data repositories to search
651 <Error message from query resolver>
653 <Communications error> with <location> <location description>
660 <Error> from <location> <location description>
700 <SQL parse error message>
750 <SQL semantic error message>
751 <Error message from query resolver>
761 <Requirements error> for <location> <location description>
790 Internal error: <fatal error from SNQP server>

Table 3b: Reply Codes
(Continued)

Table 4 lists the defined GUI reply codes. Text messages for the reply codes may vary. The codes are sorted numerically. An explanation of the codes follows the table.

---

331 nnnnnnn! <message>
331 nnnnnnn.mmmmmm <message>
331 nnnnnnn-mmmmmm <message>
730 nnnnnnnammmmmm e <parse error message>
735 nnnnnnnammmmmm e <semantic error message>
340 <status>

Table 4: GUI Reply Codes

In Table 4, nnnnnnn is the line number in a query block, and mmmmmm is the column in the line. Both numbers begin counting with 1. The exclamation point response directs the program to list information after line n. The period response directs the program to break line n at column m. The hyphen response directs the program to flag line n at column m. Replies 730 and 735 direct the GUI to indicate the parsing or semantic error at line n, column m. Response 340 provides status information that can be displayed immediately in the GUI's status line. A sample status message is one that indicates which data repository is being contacted.

5. Protocol Requirements

SQL queries may often have more functionality then a server supports or an application demands. Moreover, query blocks larger than one SQL query, advice on query costs, some types of comparison operations or t-bounds need not be supported by a particular server. SNQP defines a minimal subset of functionality for a working SNQP protocol. Functionality beyond this subset is optional. Servers that do not support optional functionality must return replies that indicate this to the user.

Table 5 lists the minimum functionality for an SNQP server.

Command	Limitations
advice	Not supported.
attributes	List the attributes for a relation.
compare	List type of comparison operation. At least one of CCSO and default comparison types must be supported. Wildcards in SQL query strings can be rejected by the query command with an appropriate semantic error message.
help	Explain the available SNQP commands.
imagui	Not supported.
next	Not supported.
noadvice	Supported, but has no effect since advice is not supported.
noimagui	Supported, but has no effect since imagui is not supported.
query	Submit a block containing one SQL query statement. The minimum supported SQL query statement is a selection query that performs equality comparisons between attribute values and constant strings. Conjunctions of such comparisons are supported. The minimum SQL query does not allow projections, but returns all the attributes for matching tuples.
relations	List the relations available through the SNQP server.
stop	End processing of current query.
quit	Terminate the query session.

Table 5: Minimum SNQP Server Requirements
(Commands do not support t-bounds)

6. Security Considerations

SNQP clients and servers depend on the Domain Name Service. They are subject to all the security issues that arise in that context. This version of the SNQP protocol does not define procedures for protecting the information communicated to and from an SNQP server.

7. References

[1] American National Standards Institute. "SQL," ANSI Standard X3.135-1989. 1989.

[2] H. Garcia-Molina, G. Wiederhold. "Read-Only Transactions in a Distributed Database," ACM Transactions on Database Systems 7(2), pp. 209-234. June 1982.

[3] S. Dorner, P. Pomes. "The CCSO Nameserver: A Description," Computer and Communications Services Office Technical Report, University of Illinois, Urbana, USA. 1992. Available in the current "qi" distribution from <URL:ftp://uiarchive.cso.uiuc.edu/local/packages/ph>

[4] J. Levine, T. Mason, D. Brown. "Parsing SQL," lex yacc, 2nd ed. O'Reilly and Associates, Inc. 1992.

[5] Ordille, J., "The Internet Nomenclator Project", RFC 2258, January 1998.

[6] J. Ordille. "Descriptive Name Services for Large Internets," Ph. D. Dissertation. University of Wisconsin. 1993. <URL:http://cm.bell-labs.com/cm/cs/doc/93/12-01.ps.gz>

[7] J. Ordille, B. Miller. "Distributed Active Catalogs and Meta-Data Caching in Descriptive Name Services," Thirteenth International IEEE Conference on Distributed Computing Systems, pp. 120-129. May 1993. <URL:http://cm.bell-labs.com/cm/cs/doc/93/5-01.ps.gz>

[8] J. Ordille. "Nomenclator Home Page." 1997. <URL:http://cm.bell-labs.com/cm/cs/what/nomenclator/>

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

[10 Yeong, W., Howes, T., and S. Kille. "Lightweight Directory Access Protocol", RFC 1777, March 1995.

8. Authors' Addresses

Jim Elliott
Epic Systems Corporation
5301 Tokay Boulevard
Madison, WI 53711 USA

EMail: jim@apocalypse.org

Joann J. Ordille
Bell Labs, Lucent Technologies
Computing Sciences Research Center
700 Mountain Avenue, Rm 2C-301
Murray Hill, NJ 07974 USA

EMail: joann@bell-labs.com

9. Full Copyright Statement