Why You Should About STN

STN script commands have been available for many years, starting with the first version of STN Express, which used script language for login scripts that accessed STN and other online database systems. Over the years, STN International has introduced other applications of script language, most notably to generate Derwent World Patents Index (WPI) strategies, as well as scripted command files from lists of accession, patent, or CAS Registry number identifiers or SciFinder Answer Keys (see Figure 1 on page 41). However, in most of these cases, the functionality and power of the script language lies hidden from the user. Even for WPI search strategies, most users undoubtedly edit the scripted command files for content and search accuracy, without recognizing the broader potential of the arcane-looking script language.

Searchers should look at the script language in a new light. The command language is easy enough to understand and incorporate into everyday search work. Others have published articles and presented talks over the years describing aspects of script language. Chemical Abstracts Service presented an e-seminar (1); FIZ Karlsruhe provides helpful documentation on reducing online time and automating searches (2, 3). Litscher and Neuhaus have actively published on STN Express scripts and worked to develop a user community (4–6). However, much of the emphasis on these scripted command files has focused on processing data from external files in conjunction with online searching. Sample scripts focus narrowly on such areas as selecting specific answers from an answer set, saving specific answer sets, or remembering E-numbers after a SELECT command. What has been missing is a good explanation on "what is in it for me" — why STN searchers should use STN script commands regularly. The reasons are straightforward: to save online time and costs, to reduce errors, and to improve results by planning ahead and automating repetitive tasks.

Simple Command Files

The STN Express Command Window (see Figure 2 on page 41) is frequently referred to as a "type-ahead" buffer, because searchers often compose search strategies before hourly connect time charges take effect, either before starting search sessions or before entering expensive target databases. The strategies so composed may be considered "simple" command files (see Table 1 on page 42). These simple command files just compile search commands exactly as a searcher would enter them at a system prompt and contain no programming logic. The files can be pasted into the Command Window and run continuously one after the other, or be run line-by-line, which gives the searcher the opportunity to edit the strategy at any time in response to results returned as the search progresses.

Care Script Commands

by **Thomas E. Wolff**
Wolff Information Consulting LLC

Command files may be prepared in advance for one-time use during a search session or become part of a library for repeated use. Searchers often create and save "search hedges" or sets of search terms on concepts used frequently. These hedges might include sets of compounds, product lines, companies, authors, patent assignees, journals, or terms covering other search topics. Running these mini-search strategies can reduce the expense of saving answer sets, as long as recreating the answer sets online does not exceed the cost of storing the answer sets. Possibly the best use of command files is for running periodic update searches, also known as SDIs or alerts. Saved command files for update searches usually only need to have the update codes changed to reflect the most recent search period of interest.

The biggest issue with simple command files is the necessity to anticipate search statement L-numbers accurately in advance. While relatively easy to develop a search strategy for the beginning of a search session, since the first answer set is always L1, simple command files would need editing to account for their invocation in the middle of a search session, especially if that session did not go as anticipated and the last L-number prior to invoking the simple command file differed from what the searcher expected when composing the command file. Changing L-numbers throughout command files with many search statements and complicated

Figure 1: Command File Functions on STN Express®

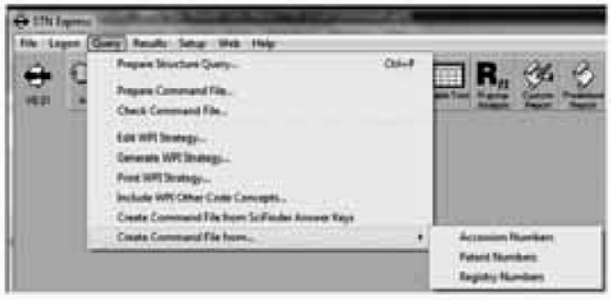

Figure 1

Figure 2: Running Command Files on STN Express®

Simple – STN Command Window (Ctrl-W), run line-by-line or multiple lines

Scripted – Upload (Ctrl-R), runs automatically

Figure 2

Table 1: Examples of Simple and Scripted Command Files

Simple	Scripted
file reg	=>file reg
act goodrns/a	=>act goodrns/a \>_line1
file hcaplus	=>file hcaplus
s l1	=>s _line1 \>_line2
s l1/p (notl) l1/dp	=>s _line1/p (notl) _line1/dp \>_line3
s l2 (s) catal?	=>s _line2 (s) catal? \>_line4
s l3 and l4	=>s _line3 and _line4 \>_line5
save l5 catal/a	=>save _line5 catal/a
dis l5 tot	=>dis _line5 tot
file stng;d his	=>file stng;d his

structure can prove an irksome chore, fraught with potential for making mistakes.

Scripted Command Files

Applying STN Script language to simple command files both overcomes the disadvantage of having to know L-numbers in advance and enhances the usefulness of the command files. The right column of Table 1 (above) shows a scripted command file with no added functionality other than variables for L-number sets. While you can prepare scripted command files in any text editor, such as STN Edit, included with STN Express, Microsoft Notepad or WordPad, I recommend using a program such as KEDIT for Windows (7, 8), a program produced by the Mansfield Software Group and available for downloading for $129 from http://www.kedit.com. It has a macro language capability to ease the task of converting fixed to variable L-numbers. STN Express also has a valuable facility to check command files before trying to run them online. In the STN Online and Results screen, navigate to Query … Check Command File (figures 1 and 2).

The features comparison of Table 2 below summarizes the implications of using scripted command files. The most evident feature of the scripted command file is the use of the primary command prompt, =>, at the beginning of each STN messenger command line. You need to use this prompt to synchronize submissions from the script processor with responses from the online host. Note that a scripted command file does not run in the usual way via the STN Command Window; instead it "uploads" to the STN session via the Query … Run Command File (CTRL-R) function (see Figure 2). The scripted command file then runs automatically from top to bottom, except for user input or intervention or for programming logic in the script file itself.

The full-featured script language offers character strings and numbers, operators, variables, conditional logic, user interaction options, file commands, and script language instructions, as explained in the STN Script Language Basics sidebar on page 43. The variables are particularly powerful. In Table 1, I only show the search statement line variables, which begin with the underscore, _, character, contain 1–12 alphanumeric characters, and are assigned using the \> operator. These are then referenced in place of L-numbers in subsequent search statements. Each STN Messenger command that generates an L-number, including activate, query, search, sort, and transfer, can be assigned a variable. In the case of commands that generate more than one L-number, such as transfer, the line variable is assigned to the last L-number generated. Any valid variable, such as _catal or _patents, can

(Text continued on page 46)

Table 2: Features Comparison of Simple and Scripted Command Files

Feature	Simple	Script
Prompts	None – looks like regular online session	=> or : (STN) ? (Dialog, Questel)
User control	Paste into STN Command Window (Ctrl-W); run line-by-line and edit lines prior to submission	Upload (Ctrl-R); runs automatically; include commands for user input
Programming logic	None	Full featured with functions, variables, operators, etc.
Variables	None	User-provided or results-based variables used in place of defined names and numbers
L-numbers	Must be accurate or requires editing during session	Programming flexibility by using variables for L-numbers and corresponding numbers of answers (STN only)

STN Script Language Basics

A script is a text file that uses features of the STN Express Script language to carry out search session actions, such as invoking STN Messenger commands (1), working with computer files, and implementing strategic decisions based on the results of the search or concurrent input by the searcher. The most common script files are the login scripts provided with STN Express for access to several online hosts including STN. These login scripts are undoubtedly the reason script language was developed, but seldom require any user attention. The login scripts are not particularly helpful in understanding script language for developing useful scripted command files for regular online searching.

You can prepare script files using STN Edit, available via the menu item Query ... Prepare Command File, or with any text editor, such as Microsoft Notepad or WordPad. KEDIT for Windows has a powerful macro language and works well for preparing script files and post-processing session capture files (2, 3). Microsoft Word will work, but I do not recommend it because of potential, residual nontext characters that STN Express does not accept, even if you save files as plain text (*.txt) with MS-DOS encoding. STN Express has a very useful function for checking script files in advance, activated by Query ... Check Command File.

Script language has many of the same kinds of components found in other programming languages, such as character strings, numbers, variables, operators and conditions, and functions for working with computer files, communications devices, and online hosts (4). Note that all content in script files, including commands, are *not* case-sensitive; uppercase is used below for clarity only. Here are highlights of Script language components most useful for user scripts; functions specific for login scripts are not included:

System Commands. STN system primary and secondary prompts, =>, :, are used before STN Messenger command statements in the script. Though the system does not recognize all script language variations, it does recognize Dialog and Questel question mark prompts (5).

Character Strings, Numbers, File Names, and Labels. Text data appears enclosed in double quotation marks with strings up to 140 characters with 80 characters per line. The backslash character, \, allows the user to continue long strings on new lines. Numbers are either positive integers or zero; negative integers are created by subtracting from zero. Computer file names are enclosed in angle brackets, < >. Statement labels begin with the at sign, @, and are used in conjunction with programming logic.

Variables. Variables are places to store information such as numerical values, text strings, file names, or the results of search statements. They begin with an underscore character, _, contain 1–12 alphanumeric characters, and are not case-sensitive. Special variables include the search statement or L-number variable, which is assigned to the results of a search statement with the special \> operator. A variable could be assigned to any STN Messenger command that generates an L-number, including activate, analyze, fsearch, fsort, query, search, sort, and transfer. The assigned variable would correspond to the last L-number in the case of commands that generate more than one L-number. The number of hits in a search statement is retrieved using the # operator. For example, the L-number created by the following search command is stored in the variable called _catal and the number answers in the L-numbered answer set called #_catal:

```
=> s catal? and titanium \> _catal
```

Useful predefined variables include _$LNUM and _$LANS for the most recent L-number and the number of answers in the most recent L-number set, and _$ENUM for the most current (largest) E-number.

Operators. Script language has all the usual operators: arithmetic, comparative (equality, greater than, less than, includes), string concatenation, and Boolean.

Conditional Logic. In IF/THEN/ELSE programming, THEN is optional and ELSE provides an alternative set of instructions. The BEGIN ... END functions can follow the IF/THEN/ELSE to execute a series of programming statements in a block.

User Interaction. The ECHO command displays the text to the user and is captured in the transcript file; by adding [NOCR] to the statement, you can eliminate the carriage return, which would add a blank line in the transcript. The GET command opens a blank user input window to allow specification of text for inclusion in an STN Messenger command. The EDIT operator, also \!, works similarly, but the user input window is preloaded with data that the user can modify before the STN Messenger command is sent to the system. The USER command temporarily transfers control to the user; script processing continues when the user hits the END key.

File Commands. These include CAPTURE, OPEN, CLOSE, DELETE files; EXEC (execute) another file, e.g., a secondary script; or UPLOAD, used for structures files. The READ and WRITE commands are used to read from and write to a single line in the OPEN file.

Script Language Instructions. These include EXIT for leaving the script immediately, even if not at the end of the script file; GOSUB for invoking a subroutine and GOTO for directing the script to a new location as designated by a label, e.g., @targetline; ONEXIT for invoking a block of statements just prior to leaving the script; PAGE for inserting a page break in the transcript; RETURN for sending the processing back just after the GOSUB statement. Comments are inserted with the * characters; all text after * is ignored in processing.

1 Mastering STN Commands [http://www.cas.org/training/stncommands].

2 "KEDIT: A Powerful Text Editor for Post-Processing Searches" (Thomas E. Wolff, *DATABASE*, June 1992, pp. 43–49 [http://tinyurl.com/28r7vt].

3 "Why Patent Searchers (And Others) Need KEDIT When They Already Have a Word Processor; Or, Post-Processing At the Power Level" (Sandra Unger, *DATABASE*, August 1994, pp. 63–67 [http://tinyurl.com/2xslyk].

4 STN Express User Guide, http://www.cas.org/support/stnexp/, *Appendix: STN Express Script Language.*

5 STN Express recognizes the question mark system prompts for Dialog and Questel. Most Script language features work for these non-STN services, including user interaction commands, operators, and programming language and logic. Unfortunately, line number and answer count variables do not work. One might still gain value by using scripted command files with substantial inclusion of ECHO, EDIT, and GET commands for user input.

Figure 3: Transcript of "Multifile with Select"

```
SCRIPT: MULTIFILE WITH SELECT

   FILE 'REGISTRY' ENTERED
L41        3 S (9003-07-0 OR 25085-53-4 OR 26063-22-9)/RN

   FILE 'RAPRA' ENTERED
L42        34 S PROPYLENE HOMOPOLYMER/CT AND
              P/DT AND UP>20070301
           SEL L42 PN 1-

   FILE 'HCAPLUS' ENTERED
L43        42 S E209-E312
L44        20 S L41/P AND P/DT AND UP>20070301
L45        62 S L43-L44
           DIS L45 BIB AB TOT
```

Figure 3

(Text continued from page 42)

be used for line assignments and variables in search statements. When creating scripted command files, the convention used by STN Express in creating WPI strategies helps keep track of search logic: set L-numbers are replaced with variables in the form _lineNN. For example, L15 in the simple command file would become variable _line15 in the scripted command file. The script processor creates a new answer count variable from each line assignment. For example, line assignments _line25 and _catal have corresponding answer count variables accessible in script language as #_line25 and #_catal. These line assignments and answer count variables work exclusively on STN, whereas most of the remaining script language features work on Dialog and Questel as well.

The scripted file in Table 1 provides a good opportunity to explore some of the script language features available through straightforward editing. The number of answers from the statement assigned to variable _line5 may be larger or smaller than the user would want to save long-term or to display the total set in the default format. To ask the user for some input on the matter, replace the save and display lines with the following lines:

```
if (#_line5 > 0)
      begin
      echo "What name do you want to use to
save answer set _line5 \
      (blank response will void save command):
" NOCR
      get _savename
      if (_savename <> "") then =>save _line5
_savename
      echo "Enter display format: " NOCR
      get _displayformat
      =>dis _line5 _displayformat tot
      end
```

The first IF statement will only call the following BEGIN … END block if the number of answers for set _line5 is greater than zero. If so, the user will be asked, via the ECHO command and a "Enter User Data" query box, to provide a name to the saved set (variable _savename), which the GET command will retrieve. If the variable is blank, the save command will not be implemented. Note that the first ECHO command was too long for this page and has a continuation character backslash (\). In this case, no provision is made for input of a valid saved set name (1–12 characters followed by /A), but one could probably elaborate the script to account for this. Instead, if an invalid name is entered, STN Messenger will prompt for a valid name even while this script is running. The same is true for the display format — although you probably could not write a script sophisticated enough to accept only valid display formats because there are so many. This script could also ask the user for the number of answers to display, instead of always printing them all (tot stands for total). The NOCR commands after each ECHO statement avoid having an extra new line ("carriage control") inserted in the transcript.

More Script Examples

Here are some additional examples that take advantage of script language features to facilitate multifile searching and answer displays.

The following script selects patent numbers from one file and then searches them in another. The resulting transcript appears in Figure 3 above.

```
echo "Script: Multifile with Select"
=>file reg
=>s 9003-07-0 or 25085-53-4 or 26063-22-9
\>_line1
=>file rapra
=>s propylene homopolymer/ct and p/dt and
up>20070301 \>_line2
=>sel _line2 pn 1-
=>file hcaplus
=>s e1-e10 \! \>_line3 \* user edits this line*
=>s _line1/p and p/dt and up>20070301 \>_line4
=>s _line3-_line4 \>_line5
if (#_line5 > 0) and (#_line5 < 100) then =>dis
_line5 bib ab tot
```

```
else =>dis _ line5 scan ti sc st hitrn
=>file stng;d his full
```

This script is designed to search for polypropylene references added to the RAPRA and HCAPLUS databases after March 1, 2007. It contains five L-number sets, which became L41–L45 in the actual transcript. Patent numbers from the RAPRA records were assigned E209–E312. The user sees an "Enter User Data" query box pre-filled with the text e1–e10 to help guide the user as to what is being asked for. Once the user corrects this data and clicks on "OK," the script will continue to the combined set of new records in L45. Since there were fewer than 100 records in set L45, all were displayed in BIB AB format according to the IF statement. If there were > 100 records, one answer would be displayed in scan ti sc st hitrn format.

An alternative to the user-edited "s e1–e10" statement above appears in the getenums script described as "Remembering the first and last E-number after SELECT command" (9). This script was written to automatically capture assigned E-numbers. It provides an excellent example of both the power and shortcomings of STN Script language. The getenums script is modular, can be called by an EXEC command from any other script, and carries out the following steps:

- Asks the user for the full SELECT statement (L-number, terms and answer numbers to select from), e.g. sel l12 pn apps 1
- Carries out SELECT in current file
- Asks the user for the name of the target database in which to search the selected E-numbers
- Asks the user for the name of the original transcript file
- Searches the assigned E-numbers and assigns the final L-number as variable _lgetenums

While the getenums script was written to simplify the E-number search process, it actually requires more user intervention than the simpler script above as a result of deficiencies in STN Script language. The simple facility to recall and process the most recent response from an STN session does not exist. In this case, capturing E-number assignments after the SELECT statement requires writing the assignment statement to a new temporary capture file (called seltemp.trn) and parsing the E-numbers from the first line of that temporary file. You will need this temporary transcript because the READ file command starts at the beginning of the open file and cannot be directed to the middle of a file or to an active transcript file. Conceptually, the script should be able to close the temporary capture file and simply continue capturing the session in the original transcript file. However, STN Script language has no means for retaining the name of the original transcript file before redirecting the transcript to the temporary transcript file. Therefore, the getenums script must ask the user for the name of the original transcript file. Since the script needs to ask the user for such "obvious" matters, the user is probably just as well off using the EDIT command to verify the E-numbers manually.

Searches frequently involve multiple files whose results need to be transferred into one target file for further processing, including duplicate elimination. The following script assumes that the user wants to retrieve patent numbers retrieved from three major files — HCAPLUS, IFICDB and ENCOMPPAT — and to combine them for entry into the Derwent subscriber WPIX file. This script could be part of a very long script covering all four online databases. However, it is easier to keep the script files for each database separate in order to avoid having to edit many _line-numbers due to changes earlier in the script file. In other words, try to keep scripted command files modular to simplify their maintenance.

```
echo "Script: Bringing Together Multifile Results"
=>file wpix
\* User will be asked to edit each TRA line below
=>tra l14 pn.b apps.b / pn apps \! \>_line1 \* from
HCAPLUS
=>tra l30 pn apps \! \>_line2 \* from IFICDB
=>tra l47 os / an \! \>_line3 \* from ENCOMPPAT
\* don't use _line1-_line3 shortcut since L-number are not consecutive in this case
=>s _line1 or _line2 or _line3 \>_line4
```

The user is asked to verify each transfer command statement, because L-numbers could have changed since the script was written. Note that ranging _line1-_line3 will not work in this case because each transfer statement produces multiple L-numbers. The correct method is to use OR logic for each separate _line-number.

You can write scripts in many ways to display answers in predetermined or user-requested formats. This script displays all answers in SCAN format without having to ask the user to instruct the system about how many more answers should be displayed after the first one.

```
echo "Script: Display Scan All Answers"
if (#_line5 = 1) =>dis _line5 scan ti sc st hitrn
else if (#_line5 > 1)
      begin
```

```
        if (#_line5 > 500)
                begin
                echo "Answer set _line5 contains
#_line5 answers."
                echo "Do you want to display all in
SCAN format? Y/(N):" NOCR
                get _reply
                if (substr(_reply,1,1) <> "y") then
goto @noscan
                end
        _numanswers = #_line5 - 1
        =>dis _line5 scan
        : _numanswers
        @noscan
        end
```

In the script above, an answer set of one record will automatically be printed in SCAN format. For answer sets with more answers, the user is asked to confirm that the whole set should be displayed in the SCAN format. Note the use of the SUBSTR (substring) command "substr(_reply,1,1)" that returns the one character of the _reply variable starting at the first position. As long as the response begins with the letter "y," all answers in the set will display in SCAN format. Otherwise the script will skip to the @noscan label and then exit at the END command. If double-checking with the user is considered unnecessary, the script could be made to display the full answer in SCAN format automatically.

Should You or Shouldn't You?

Scripted command files became a regular feature of my online searching once I discovered STN Script language and learned its potential and intricacies. I use scripted command files regularly whenever I have to prepare complicated search strategies in advance or reuse strategies from previous searches. For searches involving multiple databases, I create simple command files for each source database and convert them with KEDIT for Windows to scripted command files. I run the scripted command file for each database, one after another, without concern about what the last L-number was in the previous file. Similarly, I convert search histories into scripted command files with KEDIT for Windows and automatically convert the original L-numbers into self-consistent variables that become actual L-numbers as the search proceeds. Over the years, I have developed a library of current awareness search strategies and search hedges that support a diverse customer community. Rerunning current awareness strategies generally involves simply changing update codes.

So why should you care about STN Script Commands? To streamline your search processes and save time and money. Consider joining the community of STN Script Language users through the references and Web sites provided with this paper. We can all learn from each other. ◆

References

1 "STN: Automating Your Search," CAS e-Seminar, Aug. 31, 2004 [http://tinyurl.com/2hzcda].

2 "Reduce Online Time with Command Files and Scripts" [http://www.stn-international.de/stninterfaces/stnexpress/commandf.html].

3 "Nutzen Sie SCRIPTE um ihre Recherchen zu organisieren?" C.-D. Siems [http://www.stn-international.de/archive/presentations/cominfo04/Bern2004_STN_Scripts.pdf].

4 "InfoLit. STN Express Scripts" [http://www.infolit.ch/177.html].

5 "STN Express scripts for citation analysis in SCISEARCH and CAplus," C. Neuhaus and A. Litscher [http://www.psh.ethz.ch/people/neuhaus/stn].

6 "Using scripts to streamline citation analysis on STN International," C. Neuhaus, A. Litscher, and H.-D. Daniel, *Scientometrics*, vol. 71, no. 1, 2006, pp. 145–150.

7 "KEDIT: A Powerful Text Editor for Post-Processing Searches," Thomas E. Wolff, *DATABASE*, June 1992, pp. 43–49 [http://tinyurl.com/28r7vt].

8 "Why Patent Searchers (And Others) Need KEDIT When They Already Have a Word Processor; Or, Post-Processing At the Power Level," Sandra Unger, *DATABASE*, August 1994, pp. 63–67 [http://tinyurl.com/2xslyk].

9 "Remembering the first and last E-Number after SELECT command" [http://www.infolit.ch/258.html].

Acknowledgments

This paper is based on the presentation "Why You Should Care about STN Script Commands" presented at the Patent Information Users Group (PIUG) 2007 Annual Conference (Costa Mesa, Calif., May 5–10, 2007). The author would like to acknowledge Brian Sweet and Steve Piehler of the Chemical Abstracts Service for online time and guidance in preparation of this paper; Claus-Dieter Siems (FIZ Karlsruhe) and Andreas Litscher (Swiss representative to FIZ Karlsruhe) for feedback; and Ruth Umfleet (Celanese) for wonderful input and editing.

Thomas E. Wolff formed Wolff Information Consulting LLC in 2006 to provide technical and patent information services on a contract basis. He has a B.S. in chemistry from the Massachusetts Institute of Technology and a Ph.D. from Stanford University in bioinorganic chemistry. He is also a registered patent agent.