TNOS TScript Reference Manual

Brian A. Lantz, brian@lantz.com v1.00, 27 July 1996

The Tscript manual is the command reference manual for the menu driven hypertext Information Servers in TNOS. This scripting language may also be used to enhance the TNOS PBBS command set.

Contents

1 Introduction

One of the unique features of TNOS is the Information Servers. These are Hypertext drivers that allows tutorials, help systems, on-line surveys, etc. to be easily added to the BBS. The whole process is driven by a standard ASCII script file. The TNOS scripting language is named TScript, and is used for many features throughout TNOS.

Within TScript, all lines that begin with a '~' are control lines. All other lines are simply data. Special Text Control Sequences can be contained within a data line, allowing easy customization.

2 Information server script filenames

Who cares! The user never sees them, they see the description from the file's title line, and a number to use to choose the entry. The file must be in the spool/tutor, spool/news, or spool/info sub-directories, depending on whether your wish this to be a general tutorial, a news item, or an information source. The file *MUST* have a ".tut" file extension to be displayed as a part of one of the Information Server menus.

When using TScript outside of the Information Servers, the filename has no restrictions at all.

3 TScript title line

The first non-blank line of the TScript file is special! It serves to provide the Information servers with the description of this particular entry. The title line can also allow you to nest into sub-directories easily. This allows sub-menus very painlessly.

The title line can start with preceeding spaces or tabs (they are ignored). This allows you to center the line, since this is the first line displayed to the user. The rest of the title line will be displayed as the file's description in the Information Server menu.

The title line is also used by the TScript command that allows you to send a mail message. The title line is used as the subject of all these mail messages.

If the title line starts with a tilde (~), it expects a line of this format:

```
~ subdir description
```

The 'subdir' is the name of a sub-directory within that directory. This allows you to define one-line files that make nested sub-menus.

Outside of the Information Servers, the first line of the TScript file has no special significance, other than that already mentioned involving mail files.

4 TScript text control sequences

There are a few special character sequences, which all begin with the '~' character. All other characters pass through unchanged. The special control sequences in data lines are:

~~

replaced by a single '~' character

~n

replaced with a newline character

~c

replaced by the user's callsign

~e

replaced by the Error variable

~b

replaced with a bell character

~h

replaced by the name of the host computer

~d

replaced by the current date

~t

replaced by the current time

~u

un-terminate the current line; remove ending newline character

~l

replaced by the elapsed time of this script, in seconds

~p

replaced with the file position of the current data file

~0 .. ~9

replaced by variable string 0 .. 9

~i0 .. ~i9

replaced by value of index 0 .. 9

5 TScript control lines

All lines that begin with a '~' are treated as control lines by the TScript language. These special control lines are:

~<space>

defines a comment line; not printed or acted upon

~b num*

output 'num' blank lines.

~x

exit the script at this point. (Actually goes to label 'exit', if it exists).

~m

prompt the user with "—MORE (*y/n)—". The script then waits for the user's response. If it is anything other than 'no' (or 'n'), the script continues. If it is a 'n' response, the script ends. (see note in '~x')

~l label

define a named label at this point in the script file, named 'label'.

~q prompt

query the user with the string 'prompt' and '(*y/n)'. If the user responds with anything other than 'no' (or 'n'), the query status is set to 'y' (yes). This status is used by the "~y" and "~n" control lines.

~y label

if the query status is set to 'yes' by a "~q" or "~c" control line, then goto the label line named 'label' and continue with the script. The named label can be anywhere in the script file.

~n label

if the query status is set to 'no' by a "~q" or "~c" control line, then goto the label line named 'label' and continue with the script. The named label can be anywhere in the script file.

~g label

goto the label line named 'label' and continue with the script. The named label can be anywhere in the script file.

~v num prompt

send user 'prompt' string, get response from user, and assign the response to variable string 'num'.

~a num str

assign variable string 'num' with the string 'str'. 'str' can also be any single special character sequence.

~ap to from

appends a string variable, 'from', to the end of another 'to'.

~p to fm s* l*

picks out a sub-string of a variable string 'fm' and places it in variable string 'to'. The sub-string starts at position 's' (0 is the first position). The sub- string will be 'l' characters long, maximum.

~c n1 n2 [lab]

compares two variable strings (n1 & n2). This sets query status to 'y' if equal, 'n' if not, for use with the "~y" and "~n" control lines. If the optional label 'lab' is given and the two strings are equal, then goto the label 'lab'. The named label can be anywhere in the script file.

~j n1 n2 l* [b]

compares first 'l' characters of two variable strings (n1 & n2). This sets query status to 'y' if equal, 'n' if not, for use with the "~y" and "~n" control lines. If the optional label 'b' is given and the two strings are equal, then goto the label 'b'. The named label can be anywhere in the script file.

~t num index

truncates (chops off) variable string number 'num' at position 'index' (0 is the first position). Also, 'index' can be a Text Control Sequence integer variable ("~i1").

~i#=[val*]

assign i# the value 'val' (or 1, if no 'val'). The '#' is i0-i9. The 'val' is either a constant or a "~i#".

~i#+[val*]

add the value 'val' (or 1, if no 'val') to i#. The '#' is i0-i9. The 'val' is either a constant or a "~i#".

~i#-[val*]

subtract the value 'val' (or 1, if no 'val') from i#. The '#' is i0-i9. The 'val' is either a constant or a "~i#".

~i#?[val* [lab]]

compare the value 'val' (or 1, if no 'val') to the value of i#. The '#' is i0-i9. This command sets the query status for use with the "~y" and "~n" control lines. If the optional label 'lab' is given and the query status is 'y', then goto the label 'lab'. The named label can be anywhere in the script file.

**** NOTE: in any of the "~i" commands, the 'val' can be the "~p" Text Control Sequence.**

~z var index

gets the length of variable string 'var' and places the length in index counter number 'index'

~f [filename]

close any open io file and then create a new io file named 'filename'. To close the current io file, give no 'filename'. The query status is set to 'y' if the file open was successful. Also, 'filename' can be one of the Text Control Sequence string variables (~0-~9).

~o [filename]

close any open io file and then open an old io file named 'filename'. To close the current io file, give no 'filename'. The query status is set to 'y' if the file open was successful. Also, 'filename' can be one of the Text Control Sequence string variables (~0-~9).

~s

seek to beginning of current io file.

~se

seek to end of current io file.

~sp num

seek to the position held in integer variable 'num'.

~w textline

write the 'textline' to the current io file. The same special characters that apply to Text Lines apply to this 'textline', i.e. you can use the same control sequences.

~r num

read the next line in the current io file into the variable string number 'num'.

~e [lab]

test for an end-of-file condition on the current io file. This command sets the query status for use with the "~y" and "~n" control lines. If the optional label 'lab' is given and the io file is at the end-of-file, then goto the label 'lab'. The named label can be anywhere in the script file.

~u filename

uploads (sends a text file named 'filename' at this point in the script. The file MUST be ONLY ASCII text. Any control lines in the upload file will be only displayed. The script will continue, after sending the file, with the next line in the script file.

~k filename

kills (deletes) a file.

~d user file

delivers (mails) to 'user'. The subject of the message will be the title line of the current script. The message is sent from the MAILER-DAEMON. The content of the mail message will be the contents of the 'file'. The query status is set to 'y' if the message was successfully queued.

~dr replyto user file

delivers (mails) to 'user'. The subject of the message will be the title line of the current script. The message is sent from the MAILER-DAEMON. There is a 'Reply-to:' field added to the message, with 'replyto' as the recepient. The content of the mail message will be the contents of the 'file'. The query status is set to 'y' if the message was successfully queued.

~~ textline

a control line beginning with '~~' is treated as a text line with a '~~' control sequence at the beginning.

~? option

a control line beginning with '~*' is treated as a request for information about 'option'. The query status is set to 'y' (yes) or 'n' (no). This status is used by the "~y" and "~n" control lines.

Options available:

C

- ANSI color graphics permitted

I

- Connect was TCP/IP type

~! option

a control line beginning with '~!' is treated as a request to change information about 'option'. The status of the option is toggled on/off.

Options available:

C

- ANSI color graphics permitted

~* status

begins or ends a ANSI color block. The 'status' parameter is either 'begin' or 'end'. ANSI sequences are "eaten" unless they are permitted for this user.

~% colorfile

displays a color file, if ANSI color graphics is permitted for this user. The query status is set to 'y' (yes) or 'n' (no) depending on whether or not the user permits ANSI. This status is used by the "~y" and "~n" control lines.

~@ colorcode

changes current color set to "colorcode", if ANSI color graphics is permitted for this user.

~$ filename

executes the script named 'filename'. After the script is complete, control returns to the calling script. The query status is set to 'y' if the script was found.

~$$ filename

same as ~$ above, except that the script is executed in the background, and there is no delay waiting for the script to complete.

~h n1 n2 i [lb]

searches string n1 for the substring n2. This sets query status to 'y' if found, 'n' if not, for use with the "~y" and "~n" control lines. The index variable 'i' is set to offset into n1 that n2 was found, or 0 if not found. If the optional label 'lb' is given and the substring is found, then goto the label 'lb'. The named label can be anywhere in the script file.

~(n cstring

opens a connection on stream number 'n' (1-9). The connection is defined by 'cstring', which is in the form of a TNOS 'connect' or 'telnet' command. The current connection stream is NOT changed by this command. This sets query status to 'y' if the connection is made, 'n' if not, for use with the "~y" and "~n" control lines. If the connection is NOT made, the Error variable will contain the reason.

~) n

disconnects the connection on stream number 'n' (1-9). This sets query status to 'y' if the disconnect is made, 'n' if not, for use with the "~y" and "~n" control lines. If the disconnect is NOT made, the Error variable will contain the reason. If 'n' is the current stream, the stream is reset to stream 0, the user's stream.

~# n

changes the current I/O stream to that of stream 'n' (1-9). Stream 0 is the user's interactive I/O stream.

~= n

checks the state of I/O stream 'n' (1-9), and set the query status to 'y' if the connection is still valid, and 'n' if the connection is invalid or terminated.

~^ num u | l

converts string variable 'num' to upper or lower case.

~sql querystr

executes the mSQL 'querystr' by connecting to the configured 'sql host'

** Parameters marked with an astrick (*) can be either a literal number an index counter (~i0 - ~i9), or an variable string (~0 - ~9)*

6 More about TScript

For more information consult the TNOS-tscript-HOWTO.

Updated by Matthias Wermann, dl1bjl@db0fho.ampr.org