Network Information Services

by Brian A. Lantz/KO4KS
TPALAN Tampa Local Area Network

INTRODUCTION

In the early days of packet radio, you were lucky to find a station or two that you could connect to for a digital QSO. As time has progressed, BBSs took the role of "generic information providers", with bulletins, and E-Mail as their strengths. Current BBSs are used nearly 100% for message handling.

The BBSs have a distinct weakness in the area of providing information to the user. The two main ways that BBSs provide generic information are:

1) Requiring the user to read certain messages

2) Requiring the user to download certain files.

Both of these provide a learning hurdle to a new user, which will probably cause the user to try to "stumble through it" without the needed information.

ROSE (and other networking tools) have taken the first steps of network information services by placing a screenful of text in the memory of the switch/node to answer a few questions the user might have. This is made simple for the user, requiring him to connect to an alias; no new actions to learn.

The average new user has limited computer experience and very probably NO experience with other E-Mail systems. Networks are a concept that scares even many long-time packeteers. An information provider should take these into account and also provide a way to be interactive, at least to a point.

TNOS NETWORK INFORMATION SERVER

One of the new, unique features of TNOS is the Information Server. This is a Hypertext driver that allows tutorials, help systems, on-line surveys, etc. to be easily added to NOS. Sub-directories can be added easily to allow multi-level nesting within each Server.

The user connects to an alias (i.e. "INFO") and is presented with a numbered menu of available options. Each entry can be either a Service file, or a Subdirectory file. Their location and all other details are not needed by the user; he simply selects what he wishes from the menus.

The whole Information Server process is driven by standard ASCII text files. Lines that begin with a tilde ('~') are control lines. All other lines are simply text. These files can be uploaded from the BBS prompt, or can be automatically uploaded using a Request Server like the TNOS REQSVR.

TNOS supports three Information Servers, which all act alike, but have different aliases, directory structures and Service files. The three standard TNOS Information Servers are INFO, NEWS, and TUTOR. INFO is used for informational content; network access frequencies, BBSs available, etc. NEWS is reserved for Amateur Radio related news. The TUTOR Server is generally used for more interactive Service files. Samples include surveys, tutorials, databases, and much more.

This rest of this paper will (hopefully) serve to wet your appetite and also to document the file format used by the TNOS Information Server until better documentation is available.

NECESSARY COMPONENTS

There are only two necessary components of a file that is used as either a Subdirectory File or a Service File for the TNOS Information Server, a File Name and a Title Line.

FILE NAME:
We all can probably agree that one of the worst "features" of MS-DOS is the size limitation of filenames; 8 character name with 3 character extension. This leads to VERY CRYPTIC filenames, at best. One definite design decision of the Information Server was to NOT inflict upon the user the responsibility of having to make sense out of MS-DOS file names. With the Information Server, you can use whatever filename you want. The file MUST have a ".tut" file extension, though.

Should you use descriptive filenames? Who cares! The user never sees thern, they see the description from the file's title line, and a number that they use to choose the Service File.

Each file must be in the proper directory for the server desired. These are (by default) "spool/INFO", "spool/NEWS", and "spool/TUTOR".

Those familiar with NOS will note that the NEWS directory is used in all other variations of NOS for the NNTP directory. TNOS names the NNTP directory "NNEWS" by default.

TITLE LINE:
The first non-blank line of the Service File is special! It serves to provide the Information server with the description of this particular item. 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 when the Service File is executed. The rest of the title line will be displayed as the file's description in the Information Server menu.

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.

Table 1 lists the many filenames in an Information Server directory and their title lines. Figure 1 shows how the user sees this information.

Filename	Title Line
tpalan.tut	~ tpalan Information on the Tampa Local Area Network
tri-link.tut	~ trilink information on the <TRI-LINK>
hamfests.tut	~ hamfests ARRL HAMFEST and CONVENTION CALENDAR
rose.tut	~ rose General Information on the ROSE Network

Table 1 - Filenames and their Title Lines

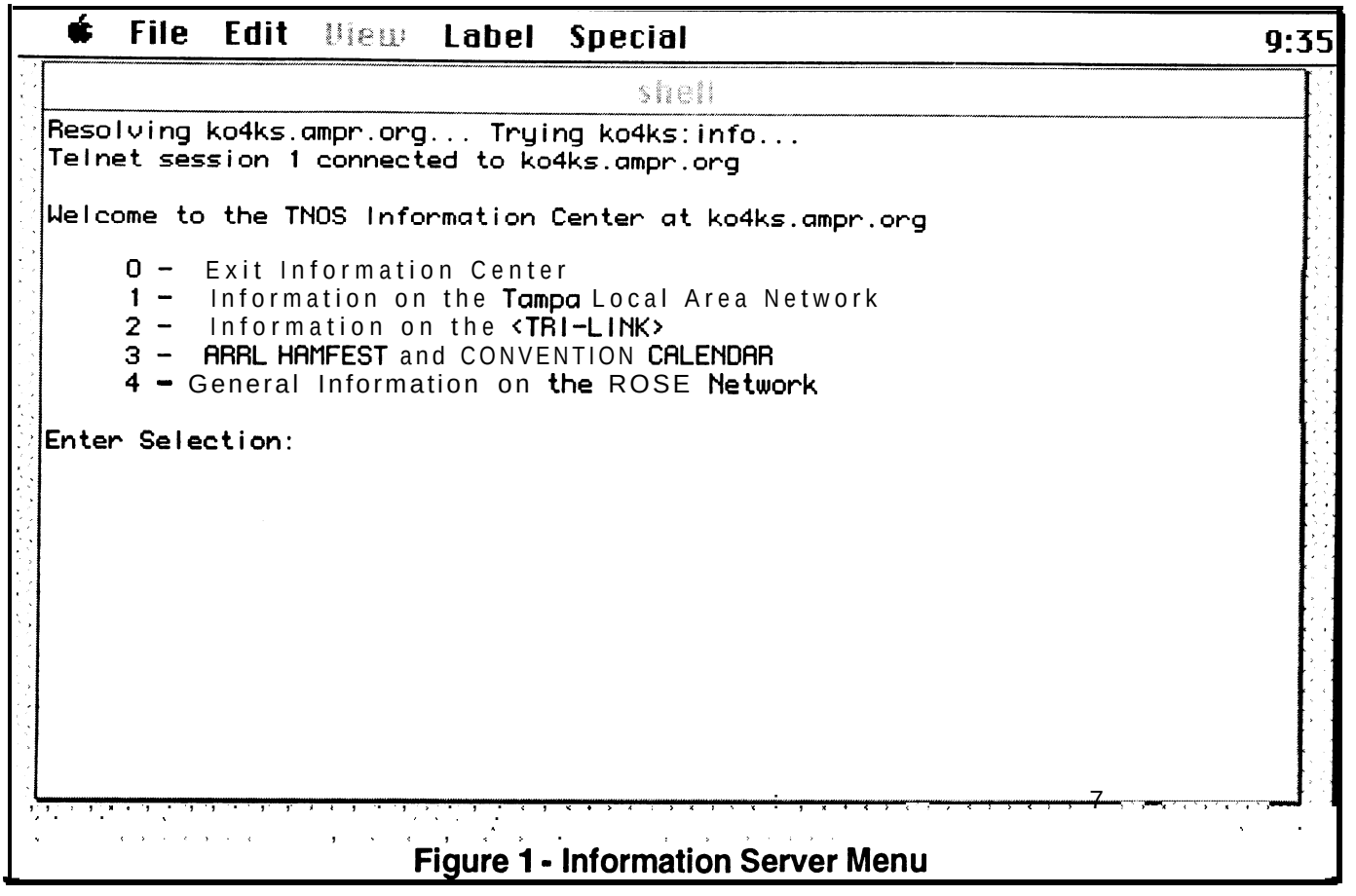

Figure 1 - Information Server Menu

ANATOMY OF A SCRIPT FILE

The Tilde character ("~") is treated specially at all time. All lines that do NOT begin with the tilde "~" character are simply sent to the user after a few simple substitutions. All substitutions begin with the tilde character and serve as shortcuts for variable data fields. The special control sequences in text lines are:

~c	replaced by the user's callsign	~i0	replaced by value of index 0
~h	replaced by the name of the host computer	~i1	replaced by value of index 1
~d	replaced by the current date	~i2	replaced by value of index 2
~t	replaced by the current time	~i3	replaced by value of index 3
~0	replaced by variable string 0	~i4	replaced by value of index 4
~1	replaced by variable string 1	~i5	replaced by value of index 5
~2	replaced by variable string 2	~i6	replaced by value of index 6
~3	replaced by variable string 3	~i7	replaced by value of index 7
~4	replaced by variable string 4	~i8	replaced by value of index 8
~5	replaced by variable string 5	~i9	replaced by value of index 9
~6	replaced by variable string 6	~b	replaced with a bell character
~7	replaced by variable string 7	~~	replaced by a single '~' character
~8	replaced by variable string 8	~n	replaced with a newline character
~9	replaced by variable string 9	~u	un-terminate current line; remove <CR>

Table 2 - Text Control Sequences

The first four sequences allow you to place the caller's callsign, host computer name, or the current date or time into the text data sent to the user.

The next 10 sequences allow you to place a string variable into the text flow. There are 10 variable strings given for the script writer's usage. Datafile text, user's responses, and other data can be stored in variable strings and displayed with the "~0" through "~9" control sequences.

The next 10 sequences allow you to output a numeric variable. As with string variables, the script writer has 10 integer variables at his/her disposal, named "~i0" through "~i9".

The next two allow you to place a bell character or a tilde character into the stream. The last two sequences allow you to imbed a carriage return into a string, or mark a line to ignore its own carriage return at the end of the text line.

All lines that BEGIN with a '~' are treated as control lines by the Information Server. In all of the following tables, parameters marked with an astrick (*) can be either a literal number, an index counter (~i0 - ~i9), or an variable string (-0 - -9).

-<space>	defines a comment line; not printed or acted upon
~~ textline	a control line beginning with '~~' is treated as a textline with a '~~' control sequence at the beginning.
~b num*	output 'num' blank lines.
~m	prompt the user with "---MORE (*y/n)---". The Server then waits for the user's response. If it is anything other than 'no' (or 'n'), the Service File continues. If it is a 'n' response, the script ends. (see note in '~x')
~x	exit the Service File at this point. (Actually goes to label 'exit', if it exists).

Table 3 - Basic Control Lines

The "-<space>" control line allows you to imbed non-displayable, no operation lines that can be used to help document the script file. The "~b" control lines allow you to place a fixed number of blanks lines into the data flow.When the "~~" control line is used, the "~~" is simply treated as ordinary characters at the beginning of an ordinary text line.

The more control line ("~m") asks the user if he/she wants more, and if so continues. If a negative response is given, the script will do an implicit goto "~g" the label named "exit". If "exit" doesn't exist, the script is complete. You can also exit the script at any time with "~x".

You can easily take a regular text file, using these few control commands, and make a decent Service File. You should pace the display so that there is a user interaction or a "more" command for every screenful of data.

~l label	define a named label at this point in the Service file, named 'label'.
~g label	goto the label line named 'label' and continue with the Server. The named label can be anywhere in the Service file.
~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 Server. The named label can be anywhere in the Service 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 Server. The named label can be anywhere in the Service file.
~v num prompt	send user 'prompt' string, get response from user, and assign the response to variable string 'num'.

Table 4 - Query and Flow Control Lines

You can use labels very much like they are used in the C programming language and assembly language. A label is any text name that makes sense to you. Spaces are not allows in a label name. A label is defined with a "4" control line.

You can go to any label in a script file with the goto "~g" control line. The label can precede or follow the got0 line in the script file.

Simple quering can be done with the query "~q" control line. This command puts out a prompt string, asks for a yes or no reply, and sets the script's response variable to either 'y' or 'n'. The query variable is used for conditional gotos in the '~y' and "~n" control lines. These lines will go to the label given if the query variable matches ('y' for the "~y" command, 'n' for the "~n" command).

A different query variation exists with the "~v" (variable query) control lines. The prompt is sent to the user, and their response is placed in the string variable that matches "num".

-a num str	assign variable string 'num' with the string 'str'. 'str' can also be any single special character sequence.
~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.
~t num index	truncates (chops off) variable string number 'num' at position 'index' (0 is the first position).
~c nl 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 Service file.
~j nl n2 l* [b]	compares first 'l' characters of two variable strings (nl & 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 Service file.
~z var index	gets the length of variable string 'var and places the length in index counter number 'index'
~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 Service file.

Table 5 - Assignment and Comparison Control Lines

You can assign an escape sequence or a literal string into a numbered string variable with the "-a" control lines. You can place a sub-string of one numbered string variable into a second numbered string variable with the "~p" control lines. The "~t" control lines allow you to truncate (clip off) a string variable at a certain point.

There are two control lines that can be used to compare string variables, "~c" and "~j". The first "~c" compares the entire string of each, while the "~j" command compares only the first part of the two strings. You can get the length of a string variable's data with the "~z" control lines.

The "~i" control lines handle four different functions with integer variables. The character after the 'i' (0 through 9) indicates which numbered integer variable to use. The next character is the character that determines the function. The '=' assigns a value. The '+' adds to the current value of a variable. The '-'

subtracts from the current value. The '?' compares the variable to a set value and sets the query variable accordingly.

~f [filename]	close any open I/O file and then create a new I/O file named 'filename'. To close the current I/O file, give no 'filename'. The query status is set to 'y' if the file open was successful.
-0 [filename]	close any open I/O file and then open an old I/O file named 'filename'. To close the current I/O file, give no 'filename'. The query status is set to 'y' if the file open was successful.
~s	seek to beginning of current I/O file.
-se	seek to end of current I/O file.
~e [lab]	test for an end-of-file condition on the current I/O 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 I/O file is at the end-of-file, then goto the label 'lab'. The named label can be anywhere in the Service file.
~w textline	write the 'textline' to the current I/O 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 I/O file into the variable string number 'num'.

Table 6 - Data File Control Lines

An I/O datafile can be opened with either the "~f" or the "~o" control lines. Both commands close any open I/O file, and open the desired file. If no filename is given, these commands simply close any previously opened file. They differ in that the "~f" creates a NEW file and sets the query variable to 'n' if the file couldn't be created or already existed. The "~o" command opens a new file and sets the query variable to 'n' if the file couldn't be opened or didn't already exist.

You can seek to the beginning ("~s") or end ("~se") of the data file. You can check for the end of file with the "~e" control lines. The query variable gets set to 'y' if at the end of the file. If a label is given on the control line and it IS the end of file, then a "goto" is executed, moving to the label line

You write to an open data file with the "~w" control lines. The rest of the line is treated the same as regular text lines, with imbedded control sequences allowed. You read from an open data file with the "~r" control lines. The next line in the data file is placed into the numbered variable string.

~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 tutorial will continue, after sending the file, with the next line in the Service 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 Service File. 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.
~? 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

Table 7 - File Utilities Control Lines

You can upload a separate text file with the "~u" control line. There is NO special processing of an uploaded file, so normal Information Server escape sequences are not examined, substituted, or acted upon.

You can use the "kill" command ("~k") to delete files on the host system's disk. FOR THIS REASON, you should limit only those that you can TRUST to be permitted to add Information Server files on your system.

You can send the contents of a file in a mail message to a user with the "~d" control lines.

Information about a user can be examined and set using the "~?" and "~!" control lines. You can use these to check the user's desire to use color. If the user came through the TNOS BBS and has ANSI color graphics mode on in the BBS, the color option will be on. Others will be initially set to off.

The username of the user is ONLY valid on starting a script if the connect was an AX25 connect, NOT a TCP/IP connect. You can find out if the user connected with IP using the "~? I" control line. If this is found to be an IP connect, you may wish to ask the user for a usemame and also ask them if they would like an ANSI display (if your script uses these features).

~* 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.

Table 8 - ANSI Color File Control Lines

The Information Server allows you to give users (that desire it) a more colorful display. You can change the text colorset, imbed ANSI control sequences in the text flow or add pre-saved ANSI color files at a particular spot in a Service File.

The "~*" control lines are used to mark the beginning and end of an imbedded ANSI sequence. The Information Server will remove KNOWN sequences from the data flow if the user has not selected ANSI graphics, but this is NOT without flaws. It is suggested that you use conditional looping based on the user's color setting (using a "~? C" control line).

A pre-saved color file is inserted into the data stream with the "~%" control line, but ONLY is displayed if the user has color set to ON. Otherwise, the file will be skipped.

The current text colorset is changed with the "~@ " control lines. The "colorcode" is a two digit, hexadecimal value that determines the colorset desired. The first digit determines the background color and the blink attribute. The second digit determines the foreground color and the high intensity attribute. These color codes are listed in Table 9.

IMPLEMENTATION

While this is already implemented in TNOS, the same code could be easily integrated into other BBSs, other variants of NOS, or any other possible information provider.

There is a stand-alone application included in the TNOS distribution that allows ANY MS-DOS user to design/view Information Server files without needing to run TNOS.

1st Digit	Blink	2nd Digit	Hi Intensity	Color
0	Normal	0	Normal	Black
1	Normal	1	Normal	Blue
2	Normal	2	Normal	Green
3	Normal	3	Normal	Cyan
4	Normal	4	Normal	Red
5	Normal	5	Normal	Magenta
6	Normal	6	Normal	Brown
7	Normal	7	Normal	White
a	Blinks FG	8	Hi Intensity	Black
9	Blinks FG	9	Hi Intensity	Blue
A	Blinks FG	A	Hi Intensity	Green
B	Blinks FG	B	Hi Intensity	Cyan
C	Blinks FG	C	Hi Intensity	Red
D	Blinks FG	D	Hi Intensity	Magenta
E	Blinks FG	E	Hi Intensity	Brown
F	Blinks FG	F	Hi Intensity	White

Table 9 - Color Code Table

SAMPLE SCRIPT

What follows is a sample Information Server scriptthat serves no REAL useful purpose, but does test (and serve as a syntax example) most of the Information Server features.

```
                    First Test Tutorial

-1 loop
This is the first test at '~h'.

Now we will test the MORE function....
~b 3
~m
~b 3
It's time to test the QUERY function....
~b 2
~q Loop back
~y loop
~b 3
Tired of looping, huh!
One more test of the QUERY function....
~b 2
~q Loop back
~n noloop
~g loop
-1 noloop
~b 3
Really tired of looping, huh!

Time to test variables....
~b 3
~v 1 Enter variable #1 (at least 5 characters)

The variable entered was ~1!
~i9=4
~p 6 1 0 ~i9
The first four characters were '-6'. ~u
~z 1 4
```

```
The length of the string was ~i4.
~t 1 4
The truncated first four characters are '~1'.
Your call sign is ~c.
-a 4 This is variable #4
--Variable number 4 is '~4'.

~v 2 Enter your call sign
~v 3 Enter your call sign again
~b 2
~c 2 3 equal
They were NOT equal-b
~g next
-1 equal
They were equal and they were '~2'.~b
-1 next
~j 2 3 2 nowokay
Even the first two characters didn't match!
~g skipit
-1 nowokay
The first two characters matched!
-1 skipit
~m
~f /newtest
~w Call sign is '~c' - Variable 4 is '~4'.~n
Writing the following to disk:
'Call sign is '~c' - Variable 4 is '~4'.'~n
~f
-0 /newtest
~r 5
Variable 5 is '~5'.
Rewinding......
~s
~r 6
Variable 6 is '~6'.
~f
~k /newtest
~n xx1
Deleted temp file
~g xx2
~l xx1
Couldn't delete temp file-b
-1 xx2
~b2
The date is ~d.~nThe time is ~t.
~m
~ now for the index counters test
~b 2
Here is the equivalent code for "for (k=1; k < 10; k++)"
~i1=
-1 indexloop
The current value of index1 is -il.
~i1?10
~i1+
~n indexloop
~i2=~i1
The copied index from index1 to index2 is ~i2.
~m
~b2
And here is the equivalent code for "for (k=10; k; k--)"
```

```
~i1=10
-1  indexloop
The current value of index1 is -il.
~i1?1
~i1-
-n  indexloop
-1 lasttest
~b2
~v 9 Enter in a number from 1 to 4
It was ~u
~i9=~9
~i9?1 was1
~i9?2 was2
~i9?3 was3
~i9?4 was4
-1 invalid
not one of the numbers asked for! Your answer was-u
~i9?0
~n badone
 either zero-nor started with a non-digit! ~u
~g reloop
-1 badone
 ~i9! ~u
-1 reloop
Try again!-b
~g lasttest
~
-1 was1
One!
~g through
-1 was2
Two!
~g through
~l was3
Three!
~g through
-1 was4
Four!
-1 through
~m
Your current CONFIG.SYS file is....
~u /config.sys
~m
I'll now CONFIG.SYS to you in a mail message
~d ~c /config.sys
~n lastoops
Sent successfully!
~g lastout
-1 lastoops
Couldn't send it!
-1 lastout
~b 2
~q Ready to exit
-n loop
~b 3
-1 exit
Goodbye,  come back and see us!-b
```

Listing 1 - A Sample, Illustration File