

CHAPTER

28

Script Statements

Introduction 247

Dictionary 249

Introduction

The statements that can be used in writing scripts are listed in this chapter. The heading for each script statement contains

- the purpose of the statement
- the type of connection you can use with the statement.

Table 28.1 on page 247 can help you decide which statements you need to use in your script. The table shows:

- the script statement name
- a brief description of the statement
- the type of connection the statement applies to: TCP/IP, TELNET, or EHELLAPI
- the frequency that the statements are used in scripts.

An alphabetical list of all script statements, their syntax and complete descriptions are presented after the table.

For examples of scripts that use these statements, see Chapter 23, “Starting and Stopping SAS/CONNECT Software,” on page 193.

Table 28.1 Summary of Script Statements

Statement	Purpose	Connection Type	Frequent (F) or Infrequent (I)
ABORT	stops execution of a script; signals an error condition	all	F
CALL	invokes a routine	all	F
DELAY	specifies the time to pause between characters being sent to the remote host	3270	I
ECHO	controls display of characters sent from the remote host while a WAITFOR statement executes	TELNET, TCP/IP	I

Statement	Purpose	Connection Type	Frequent (F) or Infrequent (I)
EOPCHAR	specifies an end-of-packet character	TELNET	I
GOTO	jumps execution to the specified script statement	all	F
IF	checks conditions before execution of labeled script statements	all	F
INPUT	displays a prompt to the user that requests a response for the remote system	all	F
LINEWAIT	specifies the time to wait after a prompt or after the last character in a packet from the remote host	TELNET, 3270	I
LOG	sends a message to the local host SAS LOG window	all	F
MAXI	specifies the maximum packet length that can be sent from the local host to the remote host (inbound packet)	TELNET	I
MAXO	specifies the maximum packet length that can be sent from the remote host to the local host (outbound packet)	TELNET	I
MODEL	specifies the IBM 3270 model being emulated	3270	I
MSG	controls remote host messages while the link is active	all	I
NOTIFY	sends a message in a window to the local SAS session	all	F
RETRY	specifies the number of attempts to re-send a packet when an error occurs	all	I
RETURN	signals the end of a routine	all	F
SCANFOR	is an alias for WAITFOR	all	I
SNAPSHOT	copies the contents of the remote host session screen to the local SAS Log window	3270	F
SOPCHAR	changes the start-of-packet character	TELNET, 3270	I
STOP	stops script execution normally	all	F
TIMEOUT	sets the time the local host waits for a packet from the remote host before re-sending the last local host packet	TELNET	F
TRACE	displays script statements as they execute	all	F
TYPE	sends characters to the remote host as if they were typed at a terminal	all	F
WAITFOR	specifies a pause until conditions are met	all	F

Statement	Purpose	Connection Type	Frequent (F) or Infrequent (I)
XCLOCK	specifies time to wait for the status line X-clock after typing an AID key	3270	I
XTIME	specifies time to wait after the status line X-clock disappears	3270	I

Dictionary

ABORT

Stops execution of a script; signals an error condition

All connections

Syntax

ABORT;

Details

The ABORT statement immediately stops execution of a script and terminates the SIGNON or the SIGNOFF function. ABORT prevents other script statements from executing when the communication link has not been established successfully. When it executes, the ABORT statement signals an error condition, and an error message is issued and displayed in the SAS Log window. To terminate script execution under normal conditions, use the STOP statement.

CALL

Invokes a routine

All connections

Release 6.06 and later

Syntax

CALL *label*;

Syntax Description

label

identifies the starting point for executing a block of statements until a RETURN statement is reached.

Details

The CALL statement causes the statements following *label* to be executed until a RETURN statement is encountered. When a RETURN statement is reached, script processing resumes at the statement that follows the CALL statement.

DELAY

Specifies the time to pause between characters that are being sent to the remote host

EHELLAPI

Syntax

DELAY *n* SECONDS;

Syntax Description

n

specifies the number of seconds to delay. The default value is .1. SECOND is an alias for SECONDS.

Details

The DELAY statement specifies the time delay between characters that are being sent from the local host to the remote host by the TYPE script statement. The DELAY statement is especially useful for correcting inhibited input problems with slow 3270 controllers.

ECHO

Controls display of characters that are sent from the remote host while a WAITFOR statement executes

TELNET, TCP/IP

Syntax

ECHO ON | OFF;

Syntax Description

ON

specifies the characters that are displayed.

OFF

specifies that the characters are not displayed. This is the default.

Details

The ECHO statement determines whether or not characters sent by the remote host are displayed while a WAITFOR statement executes. The ECHO statement is useful when you are debugging a script.

EOPCHAR

Specifies an end-of-packet character

TELNET

Syntax

EOPCHAR *character*;

Syntax Description***character***

is the ASCII character used by the remote host to detect the end of a terminal input message.

Details

The EOPCHAR statement specifies the end-of-packet character for packets that are sent from the local host to the remote host. This statement is used only for asynchronous connections that do not terminate input with a carriage return character (CR). This statement is seldom needed.

character is the ASCII character used by the remote host to detect the end of a terminal input message. You can specify *character* in one of the following forms:

- a literal character enclosed in quotes, such as '?'
- a hex constant enclosed in quotes, such as '0D'X'
- an ASCII mnemonic, such as CR.*

The default is CR (carriage return).

The end-of-packet special character cannot be an alphanumeric character, a period (.), a slash (/), or the same character that is used for the start-of-packet character.

* See the information about ASCII control character mnemonics in the description of the TYPE script statement later in this chapter.

GOTO

Jumps execution to the specified script statement

All connections

Syntax

GOTO *label*;

Syntax Description

label

is a reference to a labeled statement elsewhere in the script.

Details

The GOTO statement causes script execution to jump to the specified script statement. The GOTO statement can also be written as GO TO.

IF

Checks conditions before execution of labeled script statements

All connections

Syntax

IF *condition* **GOTO** *label*;

IF NOT *condition* **GOTO** *label*;

Syntax Description

condition

is the test performed to determine if a set of statements should be executed.

label

is a reference to a labeled statement in the script.

Details

The IF statement conditionally jumps to another statement in the script. The IF statement can check two conditions: connection type and whether the script has been called by the SIGNON command or the SIGNOFF command.

If the statement is testing for sign on or sign off, *condition* should be one of the following:

SIGNON

specifies that the SIGNON command invoked this script.

SIGNOFF

specifies that the SIGNOFF command invoked this script.

If the statement is testing for connection type, *condition* should be either FULL SCREEN or one of the values for the COMAMID= system option.

The value FULLSCREEN can be used to detect any full-screen 3270 connection. The remaining values correspond to values for the COMAMID= system option. For more information about COMAMID= values for emulation software, see “COMAMID=” on page 213.

label must be a reference to a labeled statement in the script. For example, in the following IF statement, ENDIT is a label followed by one or more statements that terminate the link when the user has issued a SIGNOFF command:

```
if signoff then goto endit;
```

INPUT

Displays a prompt to the user that requests a response for the remote system

All connections

Syntax

```
INPUT <NODISPLAY> 'prompt';
```

Syntax Description

NODISPLAY

is an optional parameter used to indicate that the input will not be displayed on the screen. This parameter is commonly used when a user is prompted to enter a password so that the password is not displayed as it is entered.

'prompt'

is a character string and must be enclosed in quotes.

Details

The INPUT statement specifies a character string that is displayed to the user when the script executes. The specified string should be a prompt that requests a response from the user, and the user must respond by at least pressing ENTER or RETURN before script execution can continue. For example, in automatic logon scripts, the INPUT statement is used to issue prompts to the user for the userid and the password that are needed to log on to the remote host.

The INPUT statement does not automatically transmit a carriage return or ENTER key. Therefore, when writing a script, you must follow an INPUT statement with a

TYPE statement if you want to transmit a carriage return or ENTER key to the remote session.

LINEWAIT

Specifies the time to wait after a prompt or after the last character in a packet from the remote host
TELNET, EHLLAPI

Syntax

LINEWAIT *n* SECONDS;

Syntax Description

n
specifies the number of seconds to wait. The default value is .25. After the LINEWAIT time elapses, normal execution resumes. SECOND is an alias for SECONDS.

Details

The LINEWAIT statement specifies the number of seconds that the local host waits after receiving a prompt from the remote host or the last character in a packet from the remote host. If the LINEWAIT time is too short, information that is sent by the remote host can be lost.

Do not confuse the LINEWAIT statement with the CHARWAIT statement. CHARWAIT specifies how long the local host waits for a prompt from the remote host or for successive characters in a packet from the remote host. LINEWAIT specifies how long the local host waits after receiving a prompt or complete packet.

LOG

Sends a message to the local host SAS log
All connections

Syntax

LOG *'message'*;

Syntax Description

'message'

is a text string that must be enclosed in quotes.

Details

The LOG statement specifies a message to be written to the SAS log. You can use this statement to issue informative notes or error messages to the user as the script executes.

message is a text string that must be enclosed in quotes. For example, the sample scripts from SAS Institute use the following LOG statement to inform users that the SIGNOFF completed successfully:

```
log 'NOTE: SAS/CONNECT conversation terminated.';
```

MAXI

Specifies the maximum packet length for data sent from the local host to the remote host

TELNET

Syntax

MAXI *bytes*;

Syntax Description***bytes***

is an integer from 72 through 4100. The default value is 512.

Details

The MAXI statement specifies the maximum packet length for data sent from the local host to the remote host (an inbound packet).

This statement is typically used in conjunction with the MAXO statement. The MAXI and MAXO statements can be used to restrict packet lengths when the length of the packet is limited by the hardware connections or maximum message sizes.

You can also indirectly control the transmission time between the local host and the remote host with the MAXI and MAXO statements. For example, suppose you specify 600 bytes for both the MAXI and MAXO statements and the baud rate is 1200. With 600-byte packets at 1200 baud, it takes about 6 seconds to transmit one packet between the local host and the remote host. If you specify a packet length less than 600 in the MAXI and MAXO statements, the packets are smaller and more of them are necessary to send the same data, but the transmission time for each packet is less. However, the total time may increase because of the overhead required to process the extra packets.

You may want to change the default value to smaller values when there is interference in the communication line. The advantage is that fewer errors can occur with smaller data packets, which means fewer repeat transmissions are needed.

MAXO

Specifies the maximum packet length for data sent from the remote host to the local host

TELNET

Syntax

MAXO *bytes*;

Syntax Description

bytes

is an integer from 72 through 4100. The default value is 512.

Details

The MAXO statement specifies the maximum packet length for data sent from the remote host to the local host (outbound packet). Typically, this statement is used in conjunction with the MAXI statement.

You can restrict packet lengths or indirectly control the transmission time between the local host and the remote host with the MAXI and MAXO statements.

MODEL

Specifies the IBM 3270 model being emulated

EHLLAPI

Syntax

MODEL *n*;

Syntax Description

n

is the number of the 3270 model that is to be emulated. The model number can be 2, 3, 4, or 5. The default model number is 2.

Details

The MODEL statement is usually unnecessary and is ignored when not needed. You may need to specify it with an older version of IRMA software if SAS/CONNECT cannot detect the appropriate number of the 3270 model that you are emulating.

MSG

Controls the remote host messages while the link is active

All connections

Syntax

MSG ON | OFF;

Syntax Description

ON

specifies that messages will be detected. This is the default and recommended value.

OFF

specifies that messages will be ignored.

Details

The MSG statement determines whether or not remote host messages from the operator or other users are ignored while SAS/CONNECT is in effect.

The effect of the MSG statement is somewhat different for 3270 connections. When MSG is set to ON, any remote host messages cause the Break window to appear. You can switch to the remote host display to view the message. When MSG is set to OFF, any messages are discarded, the Break window is suppressed, and the link attempts to recover automatically.

NOTIFY

Sends a message in a window to the local SAS session

All connections

Release 6.06 and later

Syntax

NOTIFY *message*;

Syntax Description

message

is a text string that must be enclosed in quotes.

Details

The NOTIFY statement sends a message to the user on the local host by creating a window that displays the message. The user must select CONTINUE to clear the window. The NOTIFY statement is similar to the LOG statement but enables you to highlight messages that might not be noticed in the log.

RETRY

Specifies the number of attempts to re-send a packet when an error occurs

All connections

Syntax

RETRY *n*;

Syntax Description

n

is an integer that specifies the number of re-tries. The default value is 5.

Details

The RETRY statement specifies how many attempts the local host makes to re-send a packet to the remote host when an error occurs. Typically, the remote host waits as long as the local host tries to re-send, or there may be a host-dependent number of re-tries. Setting a limit on re-tries with the RETRY statement can prevent excessive line charges when communication problems occur.

RETURN

Signals the end of a routine

All connections

Syntax

RETURN;

Details

The RETURN statement indicates the end of a group of statements that form a routine in a script. The routine begins with a statement label and is invoked by a CALL statement.

SCANFOR

Is an alias for WAITFOR

All connections

Syntax

SCANFOR *pause-specification-1*
<... *pause-specification-n*>;

Syntax Description

pause-specification

See the description of *pause-specification* in the WAITFOR statement.

Details

The SCANFOR statement is an alias for the WAITFOR statement. See the description of the WAITFOR statement.

SNAPSHOT

Copies the contents of the remote host screen to the local SAS Log window

EHLAPI

Syntax

SNAPSHOT;

Details

The SNAPSHOT statement copies the contents of the remote host 3270 screen to the SAS log. Any remote host messages, including error or operator messages, are written to the SAS Log window. Blank lines are not written to the log. The SNAPSHOT statement is used primarily for debugging a script or for saving error messages from the remote host about problem reports.

SOPCHAR

Changes the start-of-packet character

TELNET, EHLLAPI

Syntax

SOPCHAR *character*;

Syntax Description

character

specifies the start-of-packet character.

Details

The SOPCHAR statement enables you to change the start-of-packet character from the dollar sign(\$), which is the default, to some other ASCII character. Do not change the start-of-packet character in an attempt to resolve unusual translation problems unless directed to do so by your SAS Support Consultant.

character specifies the start-of-packet character that you select. For remote hosts that run SAS Release 5.18, a special zap must also be applied to change the remote host start-of-packet character to the same character that is specified in the SOPCHAR statement. The zap can be obtained from the Technical Support Department at SAS Institute.

STOP

Stops script execution normally

All connections

Syntax

STOP;

Details

The STOP statement halts execution of remaining script statements. The STOP statement is used to terminate script execution under normal conditions. Typically, you use the STOP statement at the end of a group of statements that perform sign-on tasks or sign-off tasks.

To halt execution under abnormal conditions, use the ABORT statement.

TIMEOUT

Sets the amount of time that the local host waits for a packet from the remote host before re-sending the last local host packet

TELNET

Syntax

TIMEOUT *n* SECONDS;

Syntax Description

n

specifies the packet time-out period.

Details

The **TIMEOUT** statement specifies the amount of time that the local host waits for a packet of data from the remote host. This is also called the packet time-out period. A *time-out* is an error condition that is produced when a response is not received within a specified time period. When a time-out occurs, the link either

- tries to re-send the last local host packet to the remote host or
- terminates the link if the re-try number is exceeded.

n specifies the packet time-out period. When the value of *n* is 0, the time-out period is 0; that is, the local host waits indefinitely for a data packet from the remote host. If you use the script to run the link interactively, the recommended **TIMEOUT** value is 0. If the remote host appears not to be responding, you can issue a break signal (usually CTRL-C) to interrupt the time-out. Abort the link and try to sign on again. Do not use this process unless you are sure that the remote host is not responding.

The default value for **TIMEOUT** is 0. **SECOND** is an alias for **SECONDS**.

If the time-out period specified is too short, a time-out may occur before the remote host can respond to the local host. In this case, a packet re-sent by the local host is discarded by the remote host. If the time-out period specified is too long, the system may wait longer than necessary before responding to errors that result from an unstable communications line.

The best reason for re-setting the **TIMEOUT** value to something other than 0 is when the local host needs to run unattended. In this case, set **TIMEOUT** to a value greater than the maximum time that the remote host needs to respond to a request from the local host. For example, if you remote-submit a **PROC** step that requires 2 minutes for the remote host to complete, the **TIMEOUT** value must be at least 120 seconds. See also **MAXI** and **MAXO** statements.

TRACE

Displays script statements as they execute

All connections

Syntax

TRACE ON | OFF;

Syntax Description

ON

specifies that statements are displayed.

OFF

specifies that statements are not displayed. This is the default.

Details

The TRACE statement specifies whether script statements are displayed in the SAS Log window as the script executes. This statement is most useful when debugging a script.

You can set the TRACE statement on or off several times in a script in order to trace execution of selected statements.

TYPE

Sends characters to the remote host as if they were typed at a terminal

All connections

Syntax

TYPE *text*;

Syntax Description

text

is the user-specified string of characters sent to the remote host.

Details

The TYPE statement sends characters to the remote host as if they had been typed on a terminal that is attached to that system. For example, in a script that automatically logs on to the remote host, you use a TYPE statement to issue the remote host logon command.

text can be any combination of the following:

- literal string(s) enclosed in quotes, such as 'any string'.
- hex character string(s) enclosed in quotes, such as '01020304X'.
- 3270 key mnemonics if you have a 3270 connection.

If you use TYPE statements in the script and some characters that are specified by the statement are not typed, try using the WAITFOR statement to establish a pause in script execution between TYPE statements.

To use a TYPE statement greater than 80 characters in a sign-on script, divide the TYPE statement into two or more TYPE statements. To divide the TYPE statement,

insert a hyphen (-) at the division point. For example, to divide the following TYPE statement:

```
type "sas options ('dmr comamid=pclink')"  
enter;
```

change it to:

```
type "sas options ('dmr comamid=-" enter;  
type "pclink')" enter;
```

Note: Do not add any spaces around the hyphen. △

ASCII Control Character Mnemonics

To specify an ASCII control character in the TYPE statement, use a mnemonic representation of the character. The following table lists the ASCII control characters and corresponding mnemonics, decimal codes, and hex values.

Note: As you use these control characters,

- Do not enclose an ASCII mnemonic in quotes.
- In the TYPE statement, use only the values from decimal 0 to 127 (hex 0 to 7F). Do not use any of the extended ASCII characters above decimal 127.

△

Table 28.2 ASCII Character Mnemonics

ASCII Control Character	Mnemonic Representation	Decimal Value	Hexadecimal Value
Null character	NUL	0	00
Start of header	SOH or CTL_A	1	01
Start of text	STX or CTL_B	2	02
End of Text	ETX or CTL_C	3	03
End of transmission	EOT or CTL_D	4	04
Enquiry	ENQ or CTL_E	5	05
Acknowledge positive	ACK or CTL_F	6	06
Bell	BEL or CTL_G	7	07
Backspace	BS or CTL_H	8	08
Horizontal tabulation	HT or CTL_I	9	09
Line feed	LF or CTL_J	10	0A
Vertical tabulation	VT or CTL_K	11	0B
Form feed	FF or CTL_L	12	0C
Carriage return	CR or CTL_M	13	0D
Shift out	SO or CTL_N	14	0E
Shift in	SI or CTL_O	15	0F
Data link escape	DLE or CTL_P	16	10

ASCII Control Character	Mnemonic Representation	Decimal Value	Hexadecimal Value
Device control 1 (XON)	DC1 or CTL_Q	17	11
Device control 2	DC2 or CTL_R	18	12
Device control 3 (XOFF)	DC3 or CTL_S	19	13
Device control 4	DC4 or CTL_T	20	14
Negative acknowledge	NAK or CTL_U	21	15
Synchronization	SYN or CTL_V	22	16
End of text block	ETB or CTL_W	23	17
Cancel	CAN or CTL_X	24	18
End of medium	EM or CTL_Y	25	19
Substitute	SUB or CTL_Z	26	1A
Escape	ESC	27	1B
File separator	FS	28	1C
Group separator	GS	29	1D
Record separator	RS	30	1E
Unit separator	US	31	1F
Blank space	SP	32	20
Delete or rubout	DEL	127	7F

3270 Key Mnemonics

Users with 3270 connections can specify certain 3270 keyboard keys in the TYPE statement by specifying a mnemonic representation of the key. One type of key that can be specified is an attention ID key (abbreviated AID key). An *AID key* requests the attention of the remote host so that the remote host receives input from the emulated 3270 terminal. The other type of key that can be specified is one that performs a local 3270 function, for example, cursor keys. Keys performing local functions do not solicit a response from the remote host.

The following list shows the mnemonics for AID keys. Note that the mnemonics correspond to the names of the keys on the 3270 keyboard.

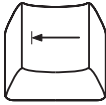
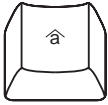
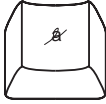
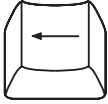
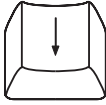
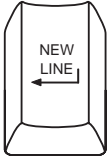


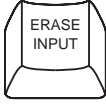
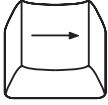

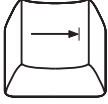


PA1	PF1	PF9	PF17
PA2	PF2	PF10	PF18
PA3	PF3	PF11	PF19
CLEAR	PF4	PF12	PF20
ATTN	PF5	PF13	PF21
ENTER	PF6	PF14	PF22
CURSEL	PF7	PF15	PF23
SYSREQ	PF8	PF16	PF24
TAB			

AID keys cannot always be used in rapid succession, just as you cannot necessarily press them in succession on your keyboard. You may need to use a WAITFOR statement

between TYPE statements that specify successive AID keys in order to wait for the appropriate response from the remote host. This is particularly true for CMS hosts.

The following figure shows the 3270 local function key mnemonics and the corresponding 3270 keys.

Figure 28.1 3270 Function Keys and Mnemonics

Key	Mnemonic Representation	Key	Mnemonic Representation
	BACKTAB		INSERT
	DELETE		LEFT
	DOWN		NEWLINE
	EREOF		RESET
	ERINP		RIGHT
	FLDMARK		TAB
	HOME		UP

3270 TYPE Statement Example

The following example of the TYPE statement for 3270 connections combines a literal string with an AID key mnemonic:

```
type 'sas options(dmr comamid=pclink)' ENTER;
```

WAITFOR

Specifies a pause until conditions are met

All connections

Syntax

WAITFOR *pause-specification-1*
 < . . . *pause-specification-n* >;

Syntax Description

pause-specification

is the criteria used to determine when the pause is terminated for the WAITFOR statement and processing continues.

Details

The WAITFOR statement directs the SAS System on the local host to do one of the following:

- pause for a specified time
- pause for a specified time or until specified characters from the remote host are received
- pause until specified characters from the remote host are received.

Typically, a WAITFOR statement is used after a TYPE statement sends input to the remote host that causes the local host to wait for the remote host's response to the input. For example, in the sample scripts in Chapter 23, "Starting and Stopping SAS/CONNECT Software," on page 193, a WAITFOR statement follows the TYPE statement that invokes the SAS System on the remote host. The WAITFOR statement is also used after any TYPE statement that sends a 3270 AID key to the remote host.

You can include one or more pause specifications in a WAITFOR statement. When you include more than one pause specification, use commas to separate the clauses.

The value of *pause-specification* can be either of the following:

time-clause< *:timeout-label*>

where

time-clause

specifies a time period in the form *n* SECONDS.

n is the number of seconds that the local host is to wait. If you specify 0 SECONDS, a time-out occurs almost immediately. In most cases, you should specify a value greater than 0. You can specify only one time clause in a WAITFOR statement.

:timeout-label

specifies the label of a statement later in the script. The label must be preceded by a colon (:). When you specify a label, script execution passes to the labeled statement after a time-out occurs. If no label is specified, execution proceeds with the statement that follows the WAITFOR statement.

< *screen-location*> *text-clause*< *:text-label*>

where

screen-location

indicates the screen location at which the specified text clause should be found. This specification applies only to 3270 connections. If *screen-location* is

specified, it must precede the *text-clause* to which it applies. The screen location is specified by the following, where *n* is any valid column or row number:

<COL *n*><ROW *n*>

Maximum values depend on the 3270 model being emulated (see the following table). If you specify a row without a column, the WAITFOR statement scans all columns of the given row. If you specify a column without a row, all rows of the column are scanned. The entire screen is scanned if you do not include a screen specification.

text-clause

specifies a string that the local host waits to receive from the remote host. The string can be

- a character literal enclosed in quotes
- a hex string enclosed in quotes.

When *text-clause* is specified, the SAS System on the local host reads input from the remote host, searching for the specified string. With 3270 connections, the SAS System on the local host scans the remote host screen (instead of reading characters sequentially).

:text-label

specifies the label of a statement later in the script. The label must be preceded by a colon (:). When you specify a label, script execution passes to the labeled statement after a time-out (if the label follows a time clause) or after the specified string has been read (if the label follows a text clause). If no label is specified, execution proceeds with the statement that follows the WAITFOR statement.

3270 Model	Maximum Row	Maximum Column
model 2	24	80
model 3	32	80
model 4	43	80
model 5	27	132

Usage Notes

- You must specify either a time clause or a text clause in the WAITFOR statement. Optionally, you can specify multiple text clauses, or you can combine a time clause and one or more text clauses. Labels and screen location specifications are optional.
- If the only specification in the WAITFOR statement is a time clause, there is a pause during the script's execution. When the specified time has elapsed, control passes to the next statement in the script. For example, the following WAITFOR statement causes a 2-second pause in script execution:

```
waitfor 2 seconds;
```

- If the WAITFOR statement contains a time clause followed by a label, a pause occurs and control passes to the labeled statement. The following WAITFOR statement causes a 2-second pause and then passes control to the script statement labeled STARTUP:

```
waitfor 2 seconds :startup;
```

- If the WAITFOR statement contains a time clause and a text clause, the local host waits the specified time for the specified characters from the remote host. If the local host does not receive the expected characters before the time expires, a time-out occurs and control passes to the next statement or to the labeled statement (if a label is specified by the time clause). For example, when the following WAITFOR statement executes, the local host pauses for 5 seconds and reads any input sent by the remote host:

```
waitfor 'Enter your password',
      5 seconds :nohost;
```

If the following string is sent by the remote host within 5 seconds, no time-out occurs and control passes to the next statement in the script:

```
Enter your password
```

If the string is not received within 5 seconds, a time-out occurs and control passes to the statement labeled NOHOST.

- You can specify labels for both text clauses and time clauses, as in this example:

```
waitfor 'Enter your password' :startlnk,
      5 seconds :nohost;
```

This WAITFOR statement is like the preceding example except that a label is specified after the text clause. Therefore, if the following string is sent by the remote host within 5 seconds, no time-out occurs and control passes to the statement labeled STARTLNK:

```
Enter your password
```

If the string is not received within 5 seconds, a time-out occurs and control passes to the statement labeled NOHOST, as in the previous example.

- If you do not specify a time clause (that is, if you specify only a text clause), a time-out cannot occur, and the local host waits indefinitely for the specified text response from the remote host. Therefore, you should generally specify a time clause to avoid being trapped in an infinite wait.
- If you specify multiple text clauses in a WAITFOR statement, the commas that separate the clauses imply a logical OR operator. In other words, only one of the text clauses needs to be satisfied (true).
- The following is an example of a WAITFOR statement that uses a screen location specification (applicable to 3270 connections only):

```
waitfor row 20 'ready' 8 seconds :noready;
```

This statement directs the local host to wait for the string READY to appear somewhere on row 20. If the string is found, execution continues with the next script statement. If the string is not found, a time-out occurs after 8 seconds and control passes to the statement labeled NOREADY.

XLOCK

Specifies time to wait for the status line X-clock after typing an AID key

EHLAPI (does not apply to CXI or PC 3270 interfaces)

Syntax

XCLOCK *n* SECONDS;

Syntax Description

n

is the number of seconds to wait. The default value is .25 seconds.

Details

The XCLOCK statement specifies the number of seconds to wait for the 3270 status line X-clock to appear after a TYPE statement sends a 3270 AID key, such as ENTER. The X-clock appears when an AID key is sent to the remote host and disappears when the remote host responds.

If the X-clock does not appear within the specified time, a time-out occurs and script processing resumes.

The X-clock normally appears within a fraction of a second. In fact, it is possible for the X-clock to appear and disappear so rapidly that the link cannot detect its presence. In this case, the link waits for a time-out and processing resumes.

XTIME

Specifies time to wait after the status line X-clock disappears

EHELLAPI (does not apply to CXI or PC 3270 interfaces)

Syntax

XTIME *n* SECONDS;

Syntax Description

n

specifies the amount of time that the local host should wait after the X-clock disappears. The default is 0.

Details

The XTIME statement specifies how long the local host waits after the 3270 X-clock symbol disappears. The local host waits the specified amount of time before sending input to the remote host or reading output from the remote host. Normally, the 3270 X-clock appears when you send a 3270 AID key to the remote host (for example, the ENTER key) and disappears after the remote host responds. Sometimes the X-clock re-appears when the remote host updates a 3270 screen in a line-mode session; this is

most likely to happen while you are signing on. The XTIME statement enables you to specify the minimum time the local host waits after the X-clock disappears in order to reduce keyboard-inhibited problems caused by an unexpected display of the X-clock.

The XTIME statement is not commonly used in scripts. It is not necessary to include XTIME in a script unless you are having problems with the X-clock display.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS/CONNECT User's Guide, Version 8*, Cary, NC: SAS Institute Inc., 1999. pp. 537.

SAS/CONNECT User's Guide, Version 8

Copyright © 1999 by SAS Institute Inc., Cary, NC, USA.

ISBN 1-58025-477-2

All rights reserved. Printed in the United States of America. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc.

U.S. Government Restricted Rights Notice. Use, duplication, or disclosure of the software by the government is subject to restrictions as set forth in FAR 52.227-19 Commercial Computer Software-Restricted Rights (June 1987).

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, September 1999

SAS[®] and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. [®] indicates USA registration.

IBM[®], AIX[®], DB2[®], OS/2[®], OS/390[®], RS/6000[®], System/370[™], and System/390[®] are registered trademarks or trademarks of International Business Machines Corporation. ORACLE[®] is a registered trademark or trademark of Oracle Corporation. [®] indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.

The Institute is a private company devoted to the support and further development of its software and related services.