Using Caché Terminal

Version 2009.1
30 June 2009

InterSystems Corporation 1 Memorial Drive Cambridge MA 02142 www.intersystems.com

Using Caché Terminal
Caché Version 2009.1 30 June 2009

This book was assembled and formatted in Adobe Page Description Format (PDF) using tools and information from the following sources: Sun Microsystems, RenderX, Inc., Adobe Systems, and the World Wide Web Consortium at www.w3c.org. The primary document development tools were special-purpose XML-processing applications built by InterSystems using Caché and Java.

InterSystems CACHÉ® and InterSystems ENSEMBLE®

Caché WEBLINK, Distributed Cache Protocol, M/SQL, M/NET, and M/PACT are registered trademarks of InterSystems Corporation.

INTERSYSTEMS, InterSystems HEALTHSHARE™, InterSystems DEEPSEE™ and InterSystems TRAKCARE™

InterSystems Jalapeño Technology, Enterprise Cache Protocol, ECP, and InterSystems Zen are trademarks of InterSystems Corporation.

All other brand or product names used herein are trademarks or registered trademarks of their respective companies or organizations.

For Support questions about any InterSystems products, contact:

InterSystems Worldwide Customer Support
Tel: +1 617 621-0700
Fax: +1 617 374-9391
Email: support@InterSystems.com

Table of Contents

List of Tables

About This Book

This book describes how to use the Caché Terminal, the command-line interface for Caché. It contains the following chapters:

- Introduction to the Caché Terminal
- Using the Terminal Interactively
- Controlling the Appearance and Behavior of the Terminal
- Using Terminal Scripts
- Script Command Reference
- Connecting to Remote Hosts
- Using the Terminal in Batch Mode
- Advanced Topics

For a detailed outline, see the table of contents.

For general information, see the *InterSystems Documentation Guide*.

1

Introduction to the Caché Terminal

The Caché Terminal presents a simple command-line interface for entering Caché commands and displaying current values. It is useful during learning, development, and debugging.

This chapter discusses the following topics:

- How to start the Terminal
- Background information about the Terminal
- An introduction to what you use the Terminal for
- An introduction to the ZWELCOME routine
- Information about the startup namespace that the Terminal uses
- Information about the Terminal prompt and various prompts you might see
- How to exit the Terminal

1.1 Starting the Caché Terminal

You can use the Caché Terminal interactively or in batch mode.

To use the Terminal interactively, do one of the following:

- To work with the Terminal using a local database, click the Caché cube and then click **Terminal**.
- To work with the Terminal using a database on a remote server, click the Caché cube and then click **Remote System Access > Terminal**. Then click a server name.

  Or click the Caché cube, click **Remote System Access > Telnet**. Then log on to the Caché system with your username and password.

For more information and additional options, see the chapter "Connecting to Remote Servers" in the *Caché System Administration Guide*.

In either case, you then see the Terminal window. The prompt displayed in this window indicates the namespace in which you are currently working. For example:

```
USER>
```

In batch mode, you invoke the Terminal from the operating system command line, passing to it the name of a script file to run. This mode is not available for all operating systems.

1.2 Background

Caché Terminal was designed to work with Caché applications. It emulates many of the functions of a Digital Equipment Corporation VT320 Terminal.

Caché Terminal uses two methods to communicate with Caché: local and network. The title bar indicates the communication mode currently in use.

- Local communication is used when a Caché Terminal communicates with the Caché server with which it was installed. In this case, the title bar displays **Cache TRM:***pid***(***configname***)** where:
  - *pid* is the process ID of the Caché process with which the Caché Terminal is communicating
  - *configname* is the Caché configuration in which the Caché server process is running.
- Network communication uses the TELNET protocol over TCP/IP to communicate with either a Windows Caché server or with a UNIX or OpenVMS host. In this case, the title bar displays **(***server* **NT — Caché Telnet)** where *server* is the host name of the remote server.

The communications stack for Caché is Winsock. Winsock is a network programming interface for Microsoft Windows which is based on the socket paradigm popularized in the Berkeley Software Distribution (BSD). The host entry can be either from the local file, an IP address, or, if the Winsock implementation has access to a name server, a general host name. The host name may be followed by an optional #nnn to specify a nonstandard port number.

Errors reported from this communications mode are the names of the Winsock error codes. For example, `WSAECONNREFUSED` means the connection was refused.

1.3 General Use

In the Caché Terminal, you can enter Caché commands of all kinds. For example:

```
d ^myroutine

set dirname = "c:\test"

set obj=##class(Test.MyClass).%New()
write obj.Prop1
```

Note: The Caché Terminal implicitly issues the `Use 0` command after each line you enter. This means that if you issue a Use command to direct output to some other device, that command is essentially ignored.

Also, large input buffers may defer the action of keys that attempt to stop the input flow such as **Ctrl-c** or **Ctrl-s**. This is also dependent on processor and connection speed. A special effort was made to respond to keystrokes before host input.

You can also run Terminal scripts, which are files with the extension .scr existing in your file system. The Terminal provides a small set of commands you can use in these scripts, including a command that sends a Caché command to the Terminal, as if you had typed it manually.

1.4 The ZWELCOME Routine

When Caché Terminal begins execution, the code checks for the existence of a routine called **ZWELCOME** in the %SYS namespace. If such a routine is found, it is invoked immediately prior to the terminal login sequence, if any. The name of the routine implies its intended use, as a custom identification and welcome message to users.

Note: The installation of **ZWELCOME** into the %SYS namespace requires an individual with administrator privileges and write access to the CACHESYS database.

CAUTION: The **ZWELCOME** routine executes in the %SYS namespace with an empty *$USERNAME* and *$ROLES* set to **%ALL**. Care should be taken to ensure that the failure modes of **ZWELCOME** are benign.

Here is a simple example:

```
ZWELCOME() PUBLIC ;
    ; Example
    Write !

    Set ME = ##class(%SYS.ProcessQuery).%OpenId($JOB)

    Write "Now: ", $ZDATETIME($HOROLOG, 3, 1), !
    Write "Pid/JobNo: ", ME.Pid, "/", ME.JobNumber, !
    Write "Priority: ", ME.Priority, !

    Quit
```

1.5 The Startup Namespace

When you first start the Caché Terminal, it opens in a particular namespace. This option is controlled by the **Startup Namespace** option within the user configuration. See the chapter "Users" in the *Caché Security Administration Guide*.

The command prompt displays the current namespace, such as:

```
USER>
```

1.5.1 Changing Namespaces

There are several ways to change namespaces in the Caché Terminal:

- The **ZNSPACE** command
- The **^%CD** utility
- The **$ZU(5)** function

Using the ZN Command

You can change to a new namespace with the **ZNSPACE** command (which has a short form of **ZN**):

```
USER>ZN "SAMPLES"
SAMPLES>
```

The **ZNSPACE** command takes a single string as argument that is the name of the namespace to change to. If you enter an invalid namespace name, **ZNSPACE** throws a <NAMESPACE> error. See the ZNSPACE reference page in the *Caché ObjectScript Reference* for more information.

Using the ^%CD Utility

You can change to a new namespace with the **^%CD** (change directory) utility:

```
SAMPLES>d ^%CD

Namespace: user
You're in namespace USER
Default directory is c:\intersystems\cache\mgr\user\
USER>
```

The **^%CD** utility prompts for a new namespace to change to. As in the example, the namespace name is not case sensitive. If you enter "?" at the prompt, **^%CD** displays a list of available namespaces. If you enter an invalid namespace name, **^%CD** presents an error message.

Using the $ZUTIL(5) Function

You can also change to a new namespace with the **$ZUTIL(5)** function (which has a short form of **$ZU(5)**):

```
USER>Do $ZUtil(5,"Samples")
SAMPLES>
```

This function takes as its second argument the name of the namespace to change to. As in the example, the namespace name is not case sensitive. If you enter an invalid namespace name, **$ZUTIL(5)** throws a <NAMESPACE> error. See the $ZUTIL(5) reference page in the *Caché ObjectScript Reference*.

1.6 The Terminal Prompt

As noted previously, the Terminal prompt indicates the namespace in which you are currently working. The prompt may display additional information, to indicate the transaction level or the program stack level.

1.6.1 The Transaction Level

If you are within a transaction, a prefix is appended to the prompt to indicate the transaction level. The prefix is of the form `TL`*n*`:`, where *n* is the transaction level. For example, if you are in the User namespace and you enter the Caché ObjectScript command **TSTART**, the prompt changes as follows:

```
USER>tstart

TL1:USER>
```

If you exit the Terminal, that rolls back the transaction.

1.6.2 The Program Stack Level

If an error occurs, a suffix is added to the prompt to indicate the program stack level. For example:

```
USER 5d3>
```

Enter the **Quit** command to exit the debug prompt. Or debug the error; see the chapter "Command-line Routine Debugging" in the book *Using Caché ObjectScript*.

1.6.3 The TSQL Shell

To access the TSQL shell, type `DO $SYSTEM.SQL.TSQLShell()` and press **Enter**. The prompt is then displayed with the string (`:TSQL`), as follows:

```
USER>DO $SYSTEM.SQL.TSQLShell()
Current settings :-
No current settings
Compiler is NEW

USER:TSQL>
```

To exit the TSQL shell, enter the ^ command.

For information on the TSQL shell, see the *Caché Transact-SQL (TSQL) Migration Guide*.

1.6.4 The MV Shell

To access the MV shell, type `MV` and press **Enter**. The prompt is then displayed with a colon (`:`) at the end rather than a right angle bracket (>), as follows:

```
USER>MV

USER:
```

If the MV shell has not yet been initialized, you may see messages before this prompt.

To exit the MV shell, enter the **Quit** command.

For information on the MV shell, see the chapter "Starting MultiValue" in the book *Using the MultiValue Features of Caché*.

1.6.5 Operating-system Shells

In the Caché Terminal, you can also open various operating-system shells. To do so, type `!` and press **Enter**. The Terminal then opens your default operating-system shell. The prompt then shows you the working directory. For example:

```
USER>!

c:\intersystems\cache\mgr\user\>
```

Note: On Macintosh, you cannot open the C-shell this way; you receive a permission denied error. You can, however, use other shells (Bash, Bourne, or Korn).

To exit the shell, use the `quit` or `exit` command as appropriate for the shell.

1.7 Exiting the Terminal

To exit Caché Terminal, do either of the following:

- Click **File > Exit**.
- Press **Alt+F4**.

This causes this copy of the Terminal to exit, closing any open files.

If this copy of the Terminal was connected to a server at startup, it exits on its own when the communications channel is closed.

If you accessed this copy of Terminal via **Cache Telnet** in the Caché cube, then it does not exit automatically when the communications channel is closed; instead it remains active so you that can connect again via the **Connect** menu.

2

Using the Terminal Interactively

This chapter describes the basics of how you work with the Terminal in interactive mode. It covers the following topics:

- How to scroll the display
- How to repeat previous commands
- How to copy and paste text
- How to print
- How to clear the Terminal screen (either by refreshing or by clearing your past entries)
- How to log the Terminal session

Note: If the Terminal displays a dialog box with the message **Spy Mode On**, this means that you have accidentally pressed **Alt+Shift+s**. To exit this mode, press **Alt+Shift+s** again. Spy mode is useful for DLL applications that are using the communications capabilities of Caché Terminal; this mode is not for general use and is not documented.

2.1 Scrolling

Whenever active text arrives, Caché Terminal scrolls the window to the newly arrived text. Use the scrollbar on the right to scroll up or down.

You can also use the keyboard to scroll within the Terminal as follows:

Using the Keyboard to Scroll the Terminal Display

Key Combination	Effect
Ctrl+Home	Scroll to the top of the buffer.
Ctrl+End	Scroll down to the cursor.
Ctrl+Page Up	Scroll up by one page.
Ctrl+Page Dn	Scroll down by one page.
Ctrl+Line Up	Scroll up by one line.
Ctrl+Line Dn	Scroll down by one line.

2.2 Repeating Previous Commands

To repeat a previous command, press the up arrow key repeatedly until the desired command is displayed. To enter the command, press **Enter** as usual.

2.3 Copying and Pasting Text

You can copy and paste text in the Terminal. To do this, you can use the right-click menu, the **Edit** menu, or various keyboard shortcuts. In general, three options are available:

- **Copy** copies the selected text to the clipboard.
- **Paste** pastes the contents of the clipboard, line by line, to the current position of the cursor (which is the end of the Terminal scrollback buffer). The text becomes visible in the Terminal window unless echoing has been disabled.
- **Copy + Paste** copies the selected text to the clipboard and then pastes it, line by line, to the current location of the cursor.

2.3.1 Keyboard Shortcuts

You can use the following keyboard shortcuts:

Shortcuts for Copy and Paste

Action	Basic Shortcut	Windows Shortcut
Copy	**Ctrl+Ins**	**Ctrl+c**
Paste	**Shift+Ins**	**Ctrl+v**
Copy + Paste		**Ctrl+Shift+v**

The shortcuts listed in the "Basic Shortcut" column are always enabled.

The shortcuts listed in the "Windows Shortcut" column are enabled only if you set the **Windows edit accelerators** option to Yes; for information on this setting, see the section "User Settings," in the chapter "Controlling the Appearance and Behavior of the Terminal."

2.3.2 Notes about Copying and Pasting

- As noted above, if you set the **Windows edit accelerators** option to Yes, **Ctrl+c** copies the selected text to the Windows clipboard. To transmit the **Ctrl+c** character to the Terminal, you must instead press **Ctrl+Shift+c**
- If the host has a mouse request outstanding and you wish to do a local cut and paste, press **Ctrl** while selecting the region; that mouse action is not reported to the host.
- If the copied text includes a line boundary, it is saved on the clipboard as a carriage return and a line feed. If you do not want to paste line feeds, see "User Settings."
- Caché Terminal can often paste data faster than a host can accept it. See "User Settings" for settings to control the speed of pasting. Also, line feeds can be discarded during a paste command.

2.4 Printing

To print from the Terminal, use the following options of the **File** menu:

- To select a printer and set it up for use with Terminal, click **File > Printer Setup**.
- To print the contents of the Terminal screen, click **File > Print**.
- To print the log file (or any other ASCII file), click **File > Print Log**. This option lets you select the file to print, and it does no special processing except to try to be reasonable in processing form feed characters. During printing, mouse and keyboard input is locked out of the main window and a cancel dialog box appears. Printing is done in draft mode.

2.5 Clearing the Screen

The **Edit** menu provides two different options to clear the screen:

- To reset the screen, click **Edit > Reset**. This option resets the margins, scroll region and other processing on the current page and causes Caché Terminal to repaint the window.
- To reinitialize the screen, click **Edit > Erase** or press **Ctrl+Del**. This option reinitializes the Caché Terminal window, erases all session data, and resets the scrollback region to zero.

2.6 Logging the Terminal Session

To start logging the Terminal session, click **File > Logging** (or click **Alt+l**).

This starts the logging of a session. You specify the name of the file in which you wish to capture the data. Only the output from a connection is logged (independent of the current wrap mode). If a log file is currently active, you are asked if you want to stop logging to the currently active file.

By default, log files are appended to rather than newly created. You can select to overwrite the log file by clicking on the **Overwrite** button.

The default log file name and location is specified in the registry in one of the following locations:

- If Terminal is not in learning mode:
  HKEY_CURRENT_USER/Software/Microsoft/Windows/CurrentVersion/Explorer/ComDlg32/OpenSaveMRU/log
- If Terminal is in learning mode:
  HKEY_CURRENT_USER/Software/Microsoft/Windows/CurrentVersion/Explorer/ComDlg32/OpenSaveMRU/scr

See the section "Learning Mode" in the chapter "Advanced Topics."

3

Controlling the Appearance and Behavior of the Terminal

This chapter describes different ways to control the appearance and behavior of the Terminal in interactive mode. It covers the following topics:

- How to specify the font
- How to specify the colors
- How to specify the window size
- How to define custom key combinations
- How to specify user settings
- How to specify the network encoding
- How to specify the physical character setting of the display

3.1 Specifying the Font

To specify the font size to use, click **Edit > Font**. This displays a dialog box where you can select a typeface, size, and style appropriate to your monitor and resolution.

Note: If you choose a font size that would expand the window beyond the borders of your screen, Terminal automatically resizes both screen and font to the largest size available.

Also, whenever you switch to a different size screen, the Terminal attempts to use the preselected font for that size.

3.2 Specifying the Colors

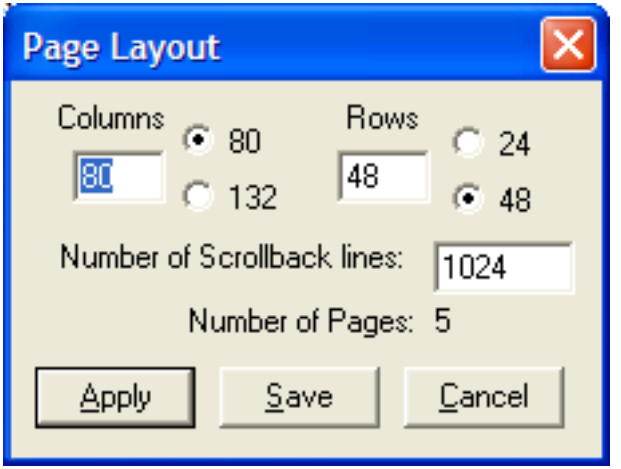
To specify the colors to use, click **Edit > Color**. This displays a dialog box where you can select the default foreground and background colors for Caché Terminal. Then click one of the following:

- **Apply** changes only the current instance of Terminal.
- **Save** does not change the current instance but saves the color information for new instances of Caché Terminal.

You can adjust the colors from the expected ANSI-named colors to any colors that the display board can deliver. These colors are saved along with the foreground and background choices. You can select the default colors by using the **Default** button.

3.3 Specifying the Window Size

To specify the Terminal window size, click **Edit > Window Size**. This displays a dialog box where you specify the window size:



The maximum number of columns is 132 and the maximum number of rows is 64. As you make changes, the dialog box updates the number of scrollback lines and the number of pages available. Then click one of the following:

- **Apply** changes only the current instance of Terminal.
- **Save** does not change the current instance but saves the window size information for new instances of Caché Terminal.

Note: When you change the window size, Terminal erases all current data in both the current display page and all back pages. Further, if there is a font selected for that size, Terminal selects it.

3.4 Defining Custom Key Combinations

To define custom key combinations, click **Edit > User Keys**. This displays a dialog box where you can associate a Caché command with any of the following key combinations: **Alt+Shift+F1** through **Alt+Shift+F10**.

Selecting **Ok+Save** updates the current instance and saves the key sequences for future instances of Caché Terminal.

To include a non-printable characters in the command, use the decimal equivalent (*nnn*) of the character. You may also use one of <CR>, <F10>, <F7>, <DO>, <TAB>, <LF>, <ESC>, <CSI>, <NL> (= <CR><LF>).

You can also use <P1>, <P2>, and other command line parameters.

Note: There are known problems with the User Keys facility. For up-to-date information, please contact the InterSystems Worldwide Response Center.

3.5 Specifying Other User Settings

To specify user settings, click **Edit > User Settings**. This displays a dialog box where you can specify both the current setup and initial values of various parameters used by Caché Terminal. The settings are as follows:

User Settings

Setting	Description
Wrap	Automatic wrapping at right hand column
<- Key sends ^H	Set to have <X> key send Ctrl+H rather than Delete
Application Keypad	Enable Application Mode keypad
Force Numeric Pad	Force standard PC keypad
Disable Special ID	Do not send special Terminal ID

Setting	Description
Disable Mouse Reports	Do not sent mouse reports
Enable Fast Paint	Enable fast paint mode
Paste Keeps Linefeed	Set to send line feeds from clipboard (along with carriage return)
Pass XOFF Through	Let remote host handle XOFF/XON.
Windows edit accelerators	Specifies whether the Terminal enables the common Windows edit shortcuts (**Ctrl+c**, **Ctrl+v**, **Ctrl+Shift+v**), in addition to the basic Terminal edit shortcuts (**Ctrl+Insert** and **Shift+Insert**). If these Windows shortcuts are not enabled, the characters are passed in the data stream to the Caché server process.
Paste burst size (in bytes)	The number of characters to paste in one transmission
Paste pause time (in msec)	The number of milliseconds between successive transmissions of pasted material longer than the burst size

Some systems cannot accept data as fast as Caché Terminal can send it. Thus, the **Paste burst size** determines how much to send at once and the **Paste pause time** determines the pause time between transmissions. If either setting is less than 1, the entire clipboard is sent at once.

3.6 Specifying the Network Encoding

The network encoding of the Terminal controls how characters are translated at the following times:

- When the Terminal translates keyboard input into its display memory, which uses Unicode. Characters received from the keyboard are translated from the single or multibyte character stream using the current Windows input character set. This means that if you change your input language, the Caché Terminal recognizes the change and adapts to it. This enables you to type in mixed languages; the Terminal recognizes it and translates properly into its internal Unicode representation. If you use mixed-language input, select UTF8 as your network encoding and your Caché **$ZMODE** I/O translation table.
- When the Terminal communicates with a peer server. Characters transmitted to the server are translated from the internal Unicode representation to a network encoding and characters received from the server are translated from the network encoding to Unicode.

The default network encoding is UTF8, even on 8-bit systems.

To specify the network encoding, click **Edit > Network Encoding**. This displays a dialog box where you can choose the network encoding for Caché Terminal to use. There are 4 choices: UTF8, Windows, ISO, and EUC. Because these encodings are not all relevant to every input locale, only the relevant choices are enabled on the menu.

3.6.1 UTF8 Encoding

When you select the UTF8 option, the Terminal translates the internal Unicode characters to UTF8 on output to the server and from UTF8 when received from the server. If you select UTF8, the Caché I/O translation for your principal I/O device must be UTF8. You can determine the I/O translation from **$ZMODE**. It is the fourth field; fields are delimited by backslashes (\).

3.6.2 Windows Encoding

When you select the Windows option, the Terminal uses the current Windows input code page to translate I/O between the Terminal and the server, to and from the internal Unicode character set encoding. When you use the Windows encoding, make sure to set the Caché I/O translation (**$ZMODE**) to that it expects the character set represented by the active Windows code page.

3.6.3 ISO Encoding

When you select the ISO option, the Terminal uses the following ISO 8859-X code pages to translate I/O to and from the peer server. The Terminal selects the appropriate ISO code page based on the current Windows input code page. The following mappings are enabled:

ISO Encoding

Language Region	ISO Standard	Windows Code Page	Network Code Page
Western European	8859-15	1252	28605
Central European	8859-2	1250	28592
Cyrillic	8859-1	1251	28591
Greek	8859-7	1253	28597
Turkish	8859-9	1254	28599
Hebrew	8859-8	1255	28598
Arabic	8859-6	1256	28596
Baltic Rim	8859-4	1257	28594

Language Region	ISO Standard	Windows Code Page	Network Code Page
Korea	iso-2022-kr	949	50225
Japan (JIS)	N/A	932	50220

All other Windows input code pages use the Windows code page if the ISO network encoding is selected.

When using the ISO encoding, you must ensure that the Caché I/O translation shown in **$ZMODE** is consistent with the character set represented by the active ISO code page used by Caché Terminal.

3.6.4 EUC Encoding

The EUC encoding is relevant to far Eastern languages and is used to communicate with certain UNIX systems. When you select the EUC option, the Terminal uses the following code pages to translate I/O to and from the server. The Terminal selects the appropriate EUC code page based on the current Windows input code page. The following mappings are enabled:

EUC Encoding

Language Region	ISO Standard	Windows Code Page	Network Code Page
Japanese	N/A	932	51932
Simplified Chinese	N/A	936	51936
Korean	N/A	949	51949

Japanese (JIS) support is provided under the ISO network encoding using the 50220 code page to translate to/from the internal Unicode.

3.7 Specifying the Physical Character Setting of the Display

To specify the physical character setting, click **Edit > Display Physical Character Setting** and then click either **Logical** and **Physical.** This option controls the aspect of the characters displayed in the Terminal window. The difference is apparent only when using multibyte character sets.

4

Using Terminal Scripts

This chapter discusses how to create and use script files for Caché Terminal.

- How to start execution of a script
- How to pause execution of a script
- How to stop execution of a script
- Contents of a script file
- A list of the commands that you can use in a script file
- General information on providing arguments to these commands
- A sample Terminal script file

Also see the next chapter, "Script Command Reference" and see the section "Learning Mode" in the chapter "Advanced Topics."

4.1 Starting Script Execution

Script files (with default extension .scr) are normally found in the working directory but could be anywhere.

To run a script, click **File > Script** or press **Alt+s**. A standard Windows file lookup box is presented and the selection of a script is made.

If a script is given as an argument on a command line, it is started immediately if no switch is locking the command mode, or deferred until after a host connection is made if there is a switch.

Note: If you edit the communications choices to a single mode, that is equivalent to locking Caché Terminal to a single choice and thus any script file is invoked after the host connection is made.

4.2 Pausing Script Execution

To pause the execution of a script, click **File > Pause** or press **Alt+p**. You are prompted to confirm that you want to pause the current script.

4.3 Stopping Script Execution

To stop a script, click **File > Script** or press **Alt+s**. You are prompted to confirm that you want to stop the current script.

4.4 Contents of a Script File

Script files are line oriented; there is no line-continuation convention. Each line is separate from any other. Lines beginning with a semicolon are considered comments. You can use blank lines liberally to improve readability. Normally, invalid lines are ignored. Script commands may be preceded by spaces and/or tabs.

The general format for a line that contains a script command is as follows. Note that arguments are interpreted as strings or numbers:

```
ScriptCommand: ScriptArguments
```

Here *ScriptCommand* is one of the Terminal script commands and *ScriptArguments* is the argument list for that command (see details for the specific commands). Note that if script command consists of two or more words, the words of the command must be separated from each other by a single space. Also note that there is no space between the command and the colon.

Or, for a command that has no arguments:

```
ScriptCommand
```

You can use labels to define points of control transfer. A label begins with a dollar sign ($), is not case-sensitive, and can have embedded spaces. A label must appear by itself on a line.

Also see the section "Learning Mode" in the chapter "Advanced Topics."

4.5 Script Command Summary

The following table gives the list of available script commands:

Terminal Script Commands

Command	Action
break	Transmit a break for those communications devices that support it
call script	Exit the current script and start another
case match	Indicate if the "wait for" string must match in case
closelog	Close a log file
connect	Force a host connection if not connected
debug	Enable/disable debugging for scripts
disconnect	Force a disconnect if connected
display	Send text to the display
echo	Turn on/off echo of incoming characters
execute	Execute a Windows program
exit	Exit the script
goto	Transfer control to another place in the script
if empty	Transfer control if last test string was empty
key_starttime	Simulate key timing start
key_stoptime	Simulate key timing stop
key_timer	Turn key timing on and off
logfile	Start a log file
multiwait for	Wait for any of several strings from the communications device
notify	Display a dialog box and wait for user response
on error	Indicate label to branch to if timer fires
pause	Pause the script

Command	Action
return	Return from a subroutine in the script file
send	Send text to the communications device
subroutine	Call a subroutine in the script file
terminate	Exit the emulator entirely
test	Build a string to be tested
timer	Control the timer for "wait for"
title	Set the window title
wait for	Wait for a particular string from the communications device

For reference information on these commands, see the later section "Script Command Reference."

4.6 Script Command Arguments

All spaces and tabs at the beginning and end of arguments are ignored.

All numeric arguments are integers. A required numeric argument defaults to 0 if not provided. Additionally, OFF is equivalent to 0 and ON is equivalent to 1.

Strings are simply the concatenation of all data on a line after the command (with the exception of the leading and trailing white space). Quotation marks are not needed. Additionally, parameter substitution is accomplished with the use of one of the following:

<P1>, <P2>, ..., <P*n*>

This substitutes the *n*-th command line parameter in place of <P*n*>.

To ease operation, certain ASCII characters have equivalent shortcut representations as shown in the following table.

Note: Any ASCII (extended) character except NUL (000) can be produced via <*ddd*> where *ddd* is the decimal value of the character.

Special Characters

Character(s)	Interpretation	Transmitted Sequence
<CR>	Carriage return	<13>
<F10>	F10 key	<27>[21-

Character(s)	Interpretation	Transmitted Sequence
<F7>	F7 key	<27>[18-
<DO>	Do key	<27>[29-
<TAB>	Tab key	<9>
<LF>	Line feed	<10>
<ESC>	Escape key	<27>
<DCS>	Device control string introducer	<144>
<ST>	Stop device control string	<156>
<EMU>	Start of extended emulator command	<144>i
<NL>	Newline	<CR><LF>
<CSI>	Control string introducer	<155>

4.7 A Sample Script

The following is a sample Terminal script:

```
; initialization -- turn match off to make comparisons more lenient
case match: off

; wait for the terminal to initialize and ask for our identification
echo: off
wait for:Username
send: SYS<CR>
wait for:Password
send: XXX<CR>
title: Terminal Example
echo: on

; log everything in a log
logfile: C:\TermExample.log
; wait a second
pause:10

; display a header to let the user know we are ready
; you need <CR><LF> because "display" does not
; have a prompt to advance to another line
display:<CR><LF>
display:------------------------------<CR><LF>
display:<<< Cache Terminal Example >>><CR><LF>
display:------------------------------<CR><LF>
; wait a second
pause:10

; switch to the USER namespace
send: znspace "USER"<CR>
wait for:USER>
```

```
; display some basic information about the system
; Use the debugging routine to do so
send: Do ^%STACK<CR>
wait for: action:

; have it outline our options
send: ?<CR>
wait for: action:

; wait 5 seconds for user to absorb
pause: 50

; ask for the basic process info
send: *s
pause: 50
send: <CR>
wait for: action:

; wait another 10 seconds
pause: 100

; finish the session
send: <CR>

; close the log file
closelog

; finished
terminate
```

5

Script Command Reference

This chapter provides reference information for the Terminal script commands.

5.1 Break

Sends a break for those communications nodes that support a break. Otherwise, it does nothing. It has no arguments. Usage example:

```
break
```

5.2 Call Script

Starts running a script. If a script is executing when this command is executed, the Terminal terminates that script before starting the new script. Usage example:

```
call script: login fred <p3>
```

This example stops the current script (if one is running) and starts another called login.scr. The first parameter in the sample script (login.scr) is `fred`. The second parameter is whatever is in the third parameter of the current script file (the one making the call). The default file extension is assumed to be .scr. The current working directory is searched first for an instance of login.scr.

5.3 Case Match

Enables or disables exact case matching in the wait for command. Usage example:

```
case match: off
```

By disabling exact case matching, you can match strings even if the individual characters differ from one another in case. The default for this switch is *on*.

5.4 Closelog

Closes the currently open log file, if any. Usage example:

```
logfile: mydirect.log
send: dir *.*/FULL<CR>
wait for: <NL>$
closelog
```

5.5 Connect

Opens a dialog box to initiate a connection to a remote host. Usage example:

```
connect
```

5.6 Debug

Enables debug mode, which traps invalid script commands, which the Terminal usually ignores. When you enable the debug mode, the first part of an invalid command appears in a message box requiring operator attention. Usage example:

```
debug: on
```

5.7 Disconnect

Disconnects from the host. If Terminal is not connected, the command does nothing. Usage example:

```
    disconnect
```

5.8 Display

Writes data to your screen. It is not sent to the communications device. Usage example:

```
    display: <CSI>H<CSI>J<LF>Here are the choices for today:
```

When this example is executed, it moves the cursor to the home position, clears the window, advances by one line, and writes the text `Here are the choices for today:` and leaves the cursor after the end of the text.

5.9 Echo

Enables or disables displayed output to the window and log file. This is useful if you need to hide output (because it is uninformative to the user, for example). Usage example:

```
    echo: off
    on error: $Failed Login
    timer: 50
    wait for: Name:
    send: <p1><CR>
    wait for: Password:
    send: <p2><CR>
    wait for: <NL>$
    echo: on
    Notify: Login is complete
    display: <CSI>H<CSI>J
    send: <CR>
    goto $Process

$Failed Login
    echo: on
    notify: Login failed.
    exit

$Process
    ;processing begins
```

This example hides the login sequence using a name and password supplied as the first two script parameters, respectively. If the login succeeds, processing begins at the indicated label. If it fails, the script displays a dialog box indicating the failure and then exits when you click **OK**.

5.10 Execute

Launches a Windows program with a SHOW attribute for its window. Usage example:

```
    execute: notepad.exe myfile.not
```

This example executes the Windows Notepad program and opens the file myfile.not inside that application. Notice that you could do the following:

```
    logfile: mydat.lst
    echo: off
    send: dir *.dat/full
    wait for: <NL>$
    closelog
    echo: on
    execute: notepad mydat.lst
```

Note: No test is made to see if the program actually starts and no wait is done for its completion.

5.11 Exit

Exits the script. A script normally exits when it reaches the end of its last line, but you may wish to exit if some event (say a login) does not occur. Usage example:

```
    on error: $byebye
    timer: 40
    wait for: event:
    goto: $Got event

$byebye:
    notify: Did not find event prompt, exiting script
    exit

$Got event:
    timer: 0
    ; more commands
```

5.12 GoTo

Transfers control to another place in the script file. This is useful for managing control flow for looping, and in response to timeout branching. Usage example:

```
    on error: $Not There
    timer: 30
    wait for: abc<CR>
    goto: $Got It

$Not There:
    ;failed to see it, send Ctrl+C
    send: <3>
    goto: $bad

$Got It:
    ;turn timer off because we got abc<CR>
    timer: 0

    ;more commands ...
```

5.13 If Empty

Causes a branch to the given label if the last **test** command found an empty string. Usage example:

```
    test: <p1>
    if empty: $No First Arg
```

The first command determines if the first parameter provided in the command line is missing or empty. The second command branches to the label `$No First Arg` if this is the case.

5.14 Key_starttime

Starts the timing of a key sequence. It takes a single numeric argument. If the argument is zero, the statistics are accumulated when you press **Enter**. Otherwise, statistics are accumulated when you press **F10**. Usage example:

```
    key_starttime: 0
```

To stop timing, use the **key_stoptime** command.

5.15 Key_stoptime

Stops a timing and accumulates statistics, if timing is currently active. Usage example:

```
    key_starttime: 0
    wait for: <esc>[14;22H
    key_stoptime
```

5.16 Key_Timer

Starts or stops the data collection of key timing information. Alternatively, you can start or stop the timer with **Alt+Shift+T**. Usage example:

```
key_timer: on
; rest of your script commands
key_timer: off
```

A file (KEYTIMER.LOG) is constructed in the system manager's directory that contains a histogram of key timings. You can use only one such timing sequence because it does not append to the current statistics file but overwrites it.

Note: To drive timings exclusively from a script file, you must use <CR> and <F10> in place of <13> and "<27>[21-" , respectively.

5.17 Logfile

Starts the collection of received data in the log file specified. If there is currently a log file active, it is quietly stopped. Use the **closelog** command to stop the logging. Usage example:

```
logfile: mydirect.log
send: dir *.*/FULL<CR>
wait for: <NL>$
closelog
```

The default directory is the Caché system manager's directory.

Log files are normally opened for append; that is, if they already exist, new data is added on the end.

5.18 MultiWait For

Synchronizes the script file with the host. Processing is suspended until the data that arrives from the host matches one of several strings given in the argument. Usage example:

```
multiwait for:  =USER>=***ERROR,=DONE
```

This example causes the script file to wait (maybe forever) until any one of three strings of characters arrives. The first non-blank character of the argument (in this instance, the equals sign) is treated as

the delimiter that breaks up the argument into substrings. So this command causes the example script to wait until one of the following sequences arrives:

```
USER>
***ERROR,
DONE
```

You can use a timer to break out of a **multiwait for** command.

See the **case match** command to turn exact case matching on or off.

Because a **case match** command may have only a single substring argument, the following two script commands are identical in function:

```
multiwait for:  =USER>
wait for: USER>
```

5.19 Notify

Displays a Windows message box and waits until the user presses **OK**. You can use this for messages to the user who runs the script. Usage example:

```
notify: Ready to send commands...
send: copy *.lst backup:*.lst<CR>
send: delete *.lst;*
```

Note: This box is modal; it cannot be interrupted except by the user.

5.20 On Error

Provides a means to specify the label for script commands to be executed if the timer expires (normally while waiting for text to arrive). See the **goto** command for an example.

5.21 Pause

Pauses a running script for a number of tenths of seconds. Usage example:

```
pause: 30
```

This example pauses execution of the script for three seconds. If the argument is 0, that is equivalent to an indefinite pause; to resume from an indefinite pause, use **Alt+P**.

5.22 Return

Used with the **subroutine** command to return to the place in the script from which the subroutine was called. See the **subroutine** command for an example.

5.23 Send

Simulates typed data (such as Caché syntax) that is sent to the currently connected host. Usage example:

```
send: zn "DOCBOOK" <CR>
```

This line changes the namespace to DOCBOOK. The <CR> at the end is necessary because the **send** command does not implicitly add a carriage return.

Another usage example is as follows:

```
send:   1<cr>2<cr>A1234<F10><32>
```

This command is equivalent to typing the following, in sequence:

- The character 1
- The carriage-return key
- The character 2
- The carriage-return key
- The he string of characters A1234
- The F10 key
- A single space character

Notice <32> is the only way to send a leading or trailing space. Those characters are removed by the command interpreter if typed normally.

5.24 Subroutine

Useful if repetitive commands are used in a script. It saves both memory and the possible need to invent many different labels. Use this command with a **return** command. Usage example:

```
    subroutine: $Send It Again
    ; some other processing
    exit

$Send It Again:
    send: <F7>Q
    on error: $skip
    timer: 30
    wait for: [22;5H
    timer: 0
    return

$skip:
    send: <3>
    ; note on error still set to $skip
    timer: 30
    wait for: function:
    timer: 0
    send: <CR>
    exit
```

Note: The subroutine stack holds 16 addresses. If you try to nest subroutine invocations deeper than that, the script fails.

5.25 Terminate

Tells Caché Terminal to exit back to Windows. Any open files are closed, extra windows are removed, and the connections are closed. Usage example:

```
    terminate
```

5.26 Test

Tests if a parameter or window property is non-empty. The command is used in conjunction with the **if empty** command. See the **if empty** command for an example.

5.27 Timer

Sets a timer to use with the **wait for** command. If the text being looked for is never received, or is missed due to timing or case match issues, **timer** is the only way to interrupt a **wait for** and continue execution of the script. Usage example:

```
    timer: 100
```

The argument is the number of tenths of a second to wait. The example sets a timer for ten seconds. See the example in the **goto** command for another example.

Note: To switch off a timer, use the following:

```
timer: 0
```

5.28 Title

Sets the Terminal window title to the specified string. Usage example:

```
title: This is my window
```

This can also be done remotely via the extended emulator commands.

5.29 Wait For

Synchronizes the script file with data that arrives from the host. Usage example:

```
wait for:  USER>
```

This example causes the script file to wait (maybe forever) until the precise string of characters USER>arrives. This particular sequence is the default prompt from Caché Terminal when in the USER namespace. So you can use this command to wait until Caché Terminal is ready for more input.

You can use a timer to break out of a **wait for** command.

See the **case match** command to turn exact case matching on or off.

6

Connecting to Remote Hosts

To connect the current Terminal session to a database on a remote host, use the **Connect** menu.

- To connect to a remote host, click **Connect > Host** or press **Alt+o**. This brings up a dialog box where you can enter the address of the desired host.

  Or click the name of the host from the **Connect** menu.
- To send a break over the communications channel, click **Connect > Send Break** or press **Alt+b**.
- To disconnect or to cancel a connection attempt, **Connect > Disconnect**.

Note: If you start the Terminal with either the /console or the /server control argument, the **Connect** option is not shown.

6.1 Example

This example starts an instance of Terminal and then manually connects it to the TELNET port on the local host to enable a console session. The example assumes that the default user ID and password are available.

1. Select **Connect > Host**.
2. In the dialog box that appears, enter `127.0.0.1` for the **Remote System address** and `23` for the **Port Number**. Select **OK**.

   The Terminal application then attempts to connect to your local host via the TELNET port.
3. At the **Username:** prompt, enter SYS and press **Enter**.
4. Then, at the **Password:** prompt, enter a password and press **Enter**.

You then see a prompt (%SYS>) that indicates that you are connected and have been placed in the **%SYS** namespace.

5. Enter Caché commands as needed.
6. To terminate the session, select **Connect >Disconnect**. Or, to terminate this instance of Terminal, click the Close box in the upper right of the window.

7

Using the Terminal in Batch Mode

For some operating systems, you can run the Caché Terminal from the command line (for example, the DOS window). This chapter is organized as follows:

- Operating system differences
- Control arguments
- Examples of running a script:
  - Example 1: getting process information (batch)
  - Example 2: getting process information (interactive)

7.1 Operating System Differences

The various versions of Microsoft Windows interpret the caret character (^) and percent character (%) differently from one another. A specific line of text may also be interpreted differently depending on whether you type it into a "batch" window (DOS prompt) or include it as input in a batch script.

The following table gives the different input sequences required to represent the following line on different operating systems:

```
cterm /console=cn_ap:cache[USER] ^%D
```

Operating System Differences

Operating System	Environment	Rule	Input Sequence
Windows NT, Windows 2000, Windows XP	DOS prompt	Double caret (^) characters	cterm /console=cn_ap:cache[USER] ^^%D
Windows NT, Windows 2000, Windows XP	Batch file	Double caret (^) characters Double percent (%) characters	cterm /console=cn_ap:cache[USER] ^^%%D
Windows 9x, Windows ME	DOS prompt or batch file	Double percent (%) characters	cterm /console=cn_ap:cache[USER] ^%%D

For more current information on representing these characters, consult the documentation for the relevant operating system.

7.2 The Batch Command Line

You can invoke Terminal from the DOS command line (cmd.exe, to be precise). The general form of the command line is:

```
cterm <Arg1> <Arg2> ... <ArgN> <ScriptFilePath>
```

where:

Invoking Terminal in Batch Mode

Item	Meaning
cterm	The invocation of the Terminal application. If the Windows environment variable *PATH* includes the location of the Caché binaries, then use the command name `cterm` or `cterm.exe`. Otherwise, you must use a full or partial path. For a default installation of Caché, the binaries are in the directory *install-dir*\Bin
<Arg*M*>	One of the control arguments for Terminal (described separately).
<ScriptFilePath>	The location of the script file.

7.3 Control Arguments

Several arguments modify the starting environment for the Terminal session. Some of these are reserved for internal use and/or debugging; they are not described here. The most useful arguments are as follows.

Note: If you start the Terminal with either the /console or the /server control argument, the **Connect** option is not shown.

7.3.1 /console=<ConnectString>

This argument specifies both the type of connection and the additional data needed to make the connection. There are two types: TELNET connections and connections to local console applications.

Note: The arguments `/console` and `/server` are mutually exclusive.

7.3.1.1 /console=cn_iptcp:<HostAddr>

This specifies the target system with which the Terminal is to interact over a TELNET connection. One of the more common uses is to run a script on the local machine. In this case, you specify the local machine IP address and port as *HostAddr*. For example:

```
cterm /console=cn_iptcp:127.0.0.1[23]
```

7.3.1.2 /console=cn_ap:<Configuration>[<NameSpace>]:<Routine>

You can run user applications in the Terminal by specifying the configuration for the application. For example, assume that the following appears as a line in a batch (.bat) file executed by a Windows 2000 system:

```
cterm /console=cn_ap:cache[USER]:^^%%D
```

This line starts a Terminal session and, if needed, a version of Caché from the `cache` configuration in the USER namespace. Then the Terminal runs the **^%D** routine which prints out the current date. When **^%D** returns, the Terminal session is closed.

A *configuration* corresponds to the name of a Caché parameter file in the system manager's directory without its suffix. The default parameter file is cache.cpf.

The namespace name is optional. If it is not supplied, the default namespace (%SYS) is used.

The colon after `cn_ap` and the brackets enclosing the namespace name are required. The colon after the closing bracket is required only if a routine name is given; otherwise, it is omitted.

7.3.2 /server=<ServerName>

This argument specifies the name of the server to use for a secure connection between this Terminal session and the given server. The server name must be one of those defined in the **Preferred Server** list available as one of the choices on the Caché cube.

Note: The arguments `/console` and `/server` are mutually exclusive.

7.3.3 /size=RowsxCols

This argument gives the initial size of the Terminal screen, in terms of rows and columns. Each of *Rows* and *Cols* must be an unsigned integer. The `x` that separates them is required as shown. No spaces are permitted in the control argument.

The allowed ranges for *Rows* and *Cols* are:

- 5 <= *Rows* <= 64
- 20 <= *Cols* <= 132

7.3.4 /pos=(X,Y)

This argument gives the initial origin of the Terminal screen in the display device window in terms of device pixels. Both *X* and *Y* must be unsigned integers. The parentheses around the pair and the comma separator are required. No spaces are permitted in the control argument.

Note: It is possible to place a window outside the displayed area by using values for *X* and *Y* that are larger than the size of the display device.

7.3.5 /ppos=(Xpct,Ypct)

This argument gives the initial origin of the Terminal screen in the display device window in terms of a percentage of the display area. Both *Xpct* and *Ypct* must be unsigned integers. The parentheses around the pair and the comma separator are required. No spaces are permitted in the control argument.

The allowed ranges for *XPct* and *Ypct* are:

- 0 <= *Xpct* <= 40
- 0 <= *Ypct* <= 40

That is, the window origin cannot be placed above and to the left of the device origin. Nor can it be placed more than 40% down or across the display device.

7.4 Examples

This section uses the example script shown earlier in this document, and shows two ways to run it.

7.4.1 Running a Script in Batch Mode

This example uses the basic debugging routine **^%STACK** to display information about the current user and Terminal process. Assuming that the script commands are stored in C:\TestScript.scr, you can run the example by typing into a DOS command window:

```
C:\CacheSys\Bin\cterm.exe /console=cn_iptcp:127.0.0.1[23] C:\TestScript.scr
```

7.4.2 Running a Script Interactively

You can invoke the previous routine in a Terminal session as follows:

```
C:\CacheSys\Bin\cterm.exe /console=cn_ap:cache[USER]:^^%STACK
```

You can manually provide the responses that the script supplied in that example. When you reply to the prompt `Stack Display Action:` by pressing the **Enter** key, the Terminal window closes.

8

Advanced Topics

This chapter discusses the following advanced topics:

- The mapping of the extended keyboard
- Special modes of the Terminal
- Using the Terminal with DDE

8.1 Mapping of the Extended Keyboard

Caché Terminal supports application keyboard mode for the extended keyboard as follows:

Key Mappings

Key	Mapped Value
Num Lock	PF1
Keypad-divide	PF2
Keypad-times	PF3
Keypad-minus	PF4
Keypad-plus	Keypad-comma
Shift+Keypad-plus	Keypad-minus
F1, F2, F3, F4	PF1, PF2, PF3, PF4 (respectively)
Shift+F1 ... Shift+F10	F11 ... F20 (respectively)

The keypad cluster of the extended keyboard is mapped this way:

Keyboard Mappings

Key	Mapped Value
Insert	Insert Here
Home	Find
Page Up	Prev Screen
Delete	Remove
End	Select
Page Down	Next Screen

The Pause key acts as a single XON/XOFF toggle key.

8.2 Special Modes

The Terminal provides a couple of special modes, accessible only by keyboard actions.

8.2.1 Key Timing Mode

To enter or exit *key timing mode*, press **Alt+Shift+t**.

This mode is useful for determination of performance of a host system under various load conditions. The output of a timing run is the file KEYTIMER.LOG in the Caché system manager's directory.

8.2.2 Learning Mode

To enter or exit *learning mode*, press **Alt+Shift+l**. Also enable logging as described earlier in this book.

Learning mode permits rapid prototyping of script files. When this mode is enabled, and logging is active, rather than simply capturing all incoming characters, the log file is a sequence of script **wait for** and **send** commands; this file can almost be played back. The **wait for** commands show up to 16 characters preceding the sent data.

The default file name and location is specified in the HKEY_CURRENT_USER hive of the registry, by the entry Software/Microsoft/Windows/CurrentVersion/Explorer/ComDlg32/OpenSaveMRU/scr

8.3 Using the Terminal with DDE

Caché Terminal supports DDE (Dynamic Data Exchange) links to permit other applications to talk to a remote host. This section assumes that you are familiar with DDE. The topics here are as follows:

- Layout — used to obtain status information. Examples are the row and column size, whether there is a connection, and so on.
- Screen — used to gather data from the Terminal screen.
- Message — used to send data to either the Terminal screen or the host.

Note: A Windows task cannot discriminate between multiple instances of Caché Terminal when it comes to using DDE. Therefore, use DDE only if *one* copy of Caché Terminal is running.

8.3.1 DDE Layout Connections

Caché Terminal supports DDE requests for what could be considered as static information through the Layout topic.

DDE Layout Connections

Item	Meaning of Returned Value
Column	The number of columns of the window.
Row	The number of rows of the window.
hWnd	The decimal equivalent of the main window handle.
Connected	A null string if there is no connection, otherwise the equivalent of the title string "mode: node"
Read	A 1 if the last received character was a CTRL/A. This can be enabled for every read with "set terminal/script" on OpenVMS systems. Its use is detection of the end of screen painting.
Script	A 1 if a script is currently running, otherwise 0.
Title	The title of the window.

8.3.2 DDE Screen Connections

Caché Terminal supports DDE requests for screen data through the Screen topic. Currently, one POKE command is available to select which part of a screen line is desired.

DDE Screen Connections

Item	Meaning of Returned Value
Cursor	The current cursor position in the form row;col.
Line	The current line (without a CR LF).
LeftLine	The left part of the current line up to but not including the character under the cursor.
RightLine	The right part of the current line including the character under the cursor.
All	The entire screen, each line is delimited by CR LF.
Piece	The currently selected piece of a screen line (without a CR LF).

Note: The item "Piece" can be POKEd with a string of the form "R*nn*C*mm*L*pp*" to cause the Piece request to retrieve (at most) *pp* characters on the screen line *nn* beginning with column *mm*. The top left corner of the screen is row 1, column 1.

8.3.3 DDE Message Connections

Caché Terminal supports DDE requests for data communications through the Message topic. These are implemented with DDE POKE commands.

DDE Message Connections

Item	Meaning of returned value
Send	The DDE message value is sent to the host if a connection is active.
Display	The DDE message value is sent to the "screen" as if it were received from the host.

Index