



Caché Terminal

Version 5.2

01 September 2006

Caché Terminal

Caché Version 5.2 01 September 2006
Copyright © 2006 InterSystems Corporation.
All rights reserved.

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.



The Caché product and its logos are registered trademarks of InterSystems Corporation.



The Ensemble product and its logos are registered trademarks of InterSystems Corporation.



The InterSystems name and logo are trademarks of InterSystems Corporation.

This document contains trade secret and confidential information which is the property of InterSystems Corporation, One Memorial Drive, Cambridge, MA 02142, or its affiliates, and is furnished for the sole purpose of the operation and maintenance of the products of InterSystems Corporation. No part of this publication is to be used for any other purpose, and this publication is not to be reproduced, copied, disclosed, transmitted, stored in a retrieval system or translated into any human or computer language, in any form, by any means, in whole or in part, without the express prior written consent of InterSystems Corporation.

The copying, use and disposition of this document and the software programs described herein is prohibited except to the limited extent set forth in the standard software license agreement(s) of InterSystems Corporation covering such programs and related documentation. InterSystems Corporation makes no representations and warranties concerning such software programs other than those set forth in such standard software license agreement(s). In addition, the liability of InterSystems Corporation for any losses or damages relating to or arising out of the use of such software programs is limited in the manner set forth in such standard software license agreement(s).

THE FOREGOING IS A GENERAL SUMMARY OF THE RESTRICTIONS AND LIMITATIONS IMPOSED BY INTERSYSTEMS CORPORATION ON THE USE OF, AND LIABILITY ARISING FROM, ITS COMPUTER SOFTWARE. FOR COMPLETE INFORMATION REFERENCE SHOULD BE MADE TO THE STANDARD SOFTWARE LICENSE AGREEMENT(S) OF INTERSYSTEMS CORPORATION, COPIES OF WHICH WILL BE MADE AVAILABLE UPON REQUEST.

InterSystems Corporation disclaims responsibility for errors which may appear in this document, and it reserves the right, in its sole discretion and without notice, to make substitutions and modifications in the products and practices described in this document.

Caché, InterSystems Caché, Caché SQL, Caché ObjectScript, Caché Object, Ensemble, InterSystems Ensemble, Ensemble Object, and Ensemble Production 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

Caché Terminal	1
1 General Characteristics	2
2 Caché Terminal Menus	3
2.1 File Menu	3
2.2 Edit Menu	5
2.3 Connect Menu	10
2.4 Help	10
3 Script Files	11
3.1 Starting Script Files	11
3.2 Stopping Script Files	11
3.3 Script File Format	11
3.4 Script Command Table	12
3.5 Script Command Arguments	13
3.6 Individual Script Command Descriptions	14
4 Extended Keyboard Features	23
4.1 Key Mappings	23
4.2 Keyboard Functions	24
5 DDE Overview	25
5.1 DDE Layout Connections	26
5.2 DDE Screen Connections	26
5.3 DDE Message Connections	27
6 An Example Script	27
6.1 The Batch Command Line	28
6.2 Control Arguments	29
6.3 The Sample Script	30
6.4 Running the Script, Example 1: Get Process Info (Batch)	32
6.5 Running the Script, Example 2: Get Process Info (Interactive)	32
6.6 Running the Script, Example 3: Manually Connecting To A Host	32
Index	35

Caché Terminal

This article is intended to give a description of the features and functions of the Caché Terminal as well as offer basic information on its use. It includes the following topics:

- [General Characteristics](#)
- [Caché Terminal Menus](#)
 - [File Menu](#)
 - [Edit Menu](#)
 - [Connect Menu](#)
 - [Help Menu](#)
- [Script File Information](#)
 - [Starting Scripts](#)
 - [Stopping Scripts](#)
 - [File Format](#)
 - [Command Table](#)
 - [Command Arguments](#)
 - [Individual Command Descriptions](#)
 - [Break](#), [Call Script](#), [Case Match](#), [Closelog](#), [Connect](#), [Debug](#), [Disconnect](#), [Display](#), [Echo](#), [Execute](#), [Exit](#), [GoTo](#), [If Empty](#), [Key_Starttime](#), [Key_Stoptime](#), [Key_Timer](#), [Logfile](#), [Multiwait For](#), [Notify](#), [On Error](#), [Pause](#), [Return](#), [Send](#), [Subroutine](#), [Terminate](#), [Test](#), [Timer](#), [Title](#), [Wait For](#)
- [Keyboard Features](#)
 - [Mappings](#)
 - [Functions](#)
- [Dynamic Data Exchange \(DDE\) Overview](#)
 - [Layout](#)

- [Screen](#)
- [Message](#)
- [An Example Script](#)
 - [The Command Line](#)
 - [Control Arguments](#)
 - [The Sample Script](#)
 - [Running the Script, Example 1: Get Process Info \(Batch\)](#)
 - [Running the Script, Example 2: Get Process Info \(Interactive\)](#)
 - [Running the Script, Example 3: Manually Connecting To A Host](#)

1 General Characteristics

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 communication mode is shown in the title bar of the Caché Terminal window.

- Local communication is used when a Caché Terminal communicates with the Caché server with which it was installed. When local communication is in use, 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. When local communication is in use, the title bar displays “<Server> NT — Caché Telnet” where <Server> is the host name of the remote server.

Caché uses Winsock as its communications stack. 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 hostname may be followed by an optional #nnn to specify a non-standard port number.

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

Note: 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.

When sized, the main window attempts to keep the most current line (from the host) visible on the bottom. Of course, the user may scroll back to previous lines. Whenever active text arrives, Caché Terminal moves the window back to the newly arrived text. This version supports a scroll back buffer of approximately 375 lines in a 24 x 80 configuration.

2 Caché Terminal Menus

As a Windows application, Caché Terminal has the usual menu items at the top of its windows. They are described here together with their submenus.

2.1 File Menu

File commands are used to control log files and script files as well as printing text files.

2.1.1 Logging (Alt+L)

This command 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 created anew. The user can select to overwrite the log file by clicking on the Overwrite button. The default log file name and location is taken from the registry key:

```
HKEY_CURRENT_USER
  Software
    Microsoft
      Windows
        CurrentVersion
          Explorer
            ComDlg32
              OpenSaveMRU
                <xxx>
```

where `<xxx>` is *log* unless you are learning a script; in that case, the information will be taken from *scr*.

2.1.2 Script (Alt+S)

This command starts the execution of a script file. If a script file is currently running, use this command to stop it. See the description of [script commands](#) for more information on script files.

2.1.3 Print Log

This command prints log files from the working directory, but can also print any ASCII file. It does no special processing except to try to be reasonable in processing formfeed 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.

2.1.4 Printer Setup

This command permits the selection and setup of a printer for Caché Terminal.

2.1.5 Print Screen

This command send the contents of the Caché Terminal screen to the default printer.

2.1.6 Exit

This command causes this copy of Caché Terminal to exit, closing any open files. Exiting Terminal can also be done with Alt+F4.

If this copy of Caché Terminal was connected to a server at startup, it exits on its own when the communications channel is closed. If it was started using the “Remote System Access” -> “Cache Telnet” from the Caché Cube, then it does not exit automatically when the communications channel is closed, but remains active to permit another connection to be attempted via the “Connect” menu.

2.2 Edit Menu

Edit commands support Clipboard control, reinitialization of the window display, as well as selection of items customized by the individual user.

2.2.1 Copy (Ctrl+Ins)

This command copies selected text to the Windows clipboard according to the standard Windows paradigm.

If copied text includes a line boundary, it is saved on the clipboard as a carriage return and a linefeed. You may not want to paste linefeeds. See the Edit User Settings commands.

If the host has a mouse request outstanding and you wish to do a local cut and paste, depress the Control key while selecting the region, the mouse action will not be reported to the host.

2.2.2 Paste (Shift+Ins)

This command transmits the current text from the clipboard to the host. The text becomes visible in the Terminal window unless echoing has been disabled. When you press the right mouse button, a menu appears with the following options for paste:

- Copy
- Paste
- Copy + Paste

Caché Terminal can often paste data faster than a host can accept it. See the [Edit User Settings](#) for settings to control the speed of pasting. Also, linefeeds can be discarded during a paste command.

2.2.3 Reset

This command resets the margins, scroll region and other processing on the current page and causes Caché Terminal to repaint the window. It is not normally needed.

2.2.4 Erase (Ctrl+Del)

This command reinitializes the Caché Terminal window, erasing all data, and resetting the scroll-back region to zero.

2.2.5 Font

This command permits you to select a font style appropriate to your monitor and resolution. Whenever Caché Terminal is switched to a different size screen, it attempts to use the preselected font.

Note: If you choose a setting for 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.

2.2.6 Colors

This command permits selection of default foreground and background colors for Caché Terminal. Selecting "Apply" changes ONLY the current copy. "Save" does not change the current copy but saves the color information for new instances of Caché Terminal.

Adjusting the colors allows changes of any color from the expected ANSI-named colors to whatever the display board can deliver. These colors are saved along with the foreground and background choices. The default colors can be selected with the "Default" button.

2.2.7 User Keys

User Keys are selected via 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.

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

Further, `<P1>`, `<P2>`, etc. name command line parameters.

Note: There are known problems with the User Keys facility in this version of Caché. For more complete, up-to-date information, please contact the [InterSystems Worldwide Response Center](#).

2.2.8 User Settings

User Settings control both the current setup and initial values of various parameters used by Caché Terminal. The simple settings are:

Setting	Description
Wrap	Automatic wrapping at right hand column

Setting	Description
<- 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
Disable Mouse Reports	Do not sent mouse reports
Enable Fast Paint	Enable fast paint mode
Paste Keeps Linefeed	Set to send LF from clipboard (along with CR)
Paste Size	Number of bytes to send from clipboard at once
Paste Wait	Number of milliseconds to wait between bursts
Pass XOFF Through	Let remote host handle XOFF/XON.
Paste burst size (in bytes)	The number of characters that will be pasted 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 (in bytes): determines how much to send at once and the Paste Pause Time (in msec): determines the pause time between transmissions. If Paste Burst Size or Paste Pause Time is less than 1, the entire clipboard is sent at once.

2.2.9 Window Size

The window size can be selected. The maximum number of columns is 132 and the maximum number of rows is 64. There are buttons to rapidly select commonly used sizes for both rows and columns.

As changes are made, the number of scrollbar lines and "pages" available are instantly updated. Selecting "Apply" updates the current instance and "Save" saves the values for future instances of Caché Terminal.

Changing the window size erases all current data in both the current display page and all back pages. Further, if there is a font selected for that size, it is also selected.

2.2.10 Network Encoding

The Caché Terminal stores characters in its display memory in Unicode, but it receives keyboard input in single or multi-byte character streams that depend on the Windows input code page for interpretation and translation to Unicode. Communication with the peer server also requires the internal Unicode characters to be translated into a network-encoded character stream and characters received from the peer to be translated into Unicode.

Characters received from the keyboard are translated from the single or multi-byte 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, you should select UTF8 as your network encoding and your Caché **\$ZMODE** I/O translation table.

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. This translation is governed by the network encoding chosen with the Network Encoding menu. The network encoding selected is sensitive to the connection method used. A local encoding is established by the Network Encoding menu when the Caché Terminal is connected locally and a remote encoding is established when a telnet connection is active. There are 4 choices: UTF8, Windows, ISO, and EUC. These encodings are not all relevant to every input locale, so only the relevant choices are enabled on the menu.

When UTF8 is selected, the internal Unicode characters are translated 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 “\” delimited field.

When Windows is selected, the current Windows input code page is used to translate I/O between Caché 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.

When ISO is selected, the following ISO 8859-X code pages are used to translate I/O to and from the peer server. The appropriate ISO code page is selected based on the current Windows input code page. The following mappings are enabled:

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
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, the Caché I/O translation shown in **\$ZMODE** should be set such that it is consistent with the character set represented by the active ISO code page used by Caché Terminal.

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

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.

2.2.11 Display Physical Character Setting

This allows the user to choose the aspect of the characters displayed in the Terminal window. The options are: Logical and Physical. The difference is only apparent when using multi-byte character sets.

2.3 Connect Menu

Connect commands support the operation of the Telnet modules.

Note: If Terminal is started with the `/console=` control argument, this menu does not appear.

2.3.1 Host (Alt+O)

This command permits the connection of Caché Terminal to a remote host. Selecting this item brings up a dialog box to obtain the address of the desired host.

2.3.2 Send Break (Alt+B)

This command sends a break over the communications channel.

2.3.3 Disconnect

If the Terminal has been previously connected to a host, selecting this item disconnects the Terminal from the host. If an attempt to connect to a remote host is underway but has not completed, this item terminates the attempt.

2.3.4 List of available hosts

Below the Disconnect menu choice is a numbered list of known hosts for convenience.

2.4 Help

Help allows selection of various types of help (including this help).

2.4.1 Help Menu

Selecting this menu item displays this material.

2.4.2 About Caché Terminal

This command displays the version number of this copy of Caché Terminal.

3 Script Files

Script files are useful for replacing tedious typing tasks. Through the use of scripts, significant control over Caché Terminal is possible.

3.1 Starting Script Files

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

Scripts are invoked either as the first argument of the command line or via the menu function File/Script... (which can also be started by the hotkey 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 there is no switch locking the command mode, or deferred until after a host connection is made if there is a switch.

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

3.2 Stopping Script Files

To stop a script, simply select the File/Script... entry once again and you are prompted to confirm that you want to stop the current script. Answer "Yes" and then Cancel the new script selection.

3.3 Script File Format

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. Blank lines may be liberally used to improve readability. Normally, invalid lines are ignored. Script commands may be preceded by spaces and/or tabs.

The general format for a line containing a script command is:

```
<ScriptCommand>: <ScriptArguments>
```

that is, the *ScriptCommand* is separated from its *ScriptArguments* by a colon. Arguments to script commands are strings or numbers. If the command has no arguments, the colon that would separate the command from its arguments must be omitted.

Labels are used to define points of control transfer. They begin with a dollar-sign (\$), are not case-sensitive, and can have embedded spaces. Labels must appear by themselves on a line.

Note: If a <ScriptCommand> consists of two or more words, the words of the command must be separated from each other by a single space.

If the colon is present after a <ScriptCommand>, it must immediately follow the last word of the command.

3.4 Script Command Table

The following table gives the list of available script commands:

Command	Action
<code>break</code>	Transmit a break for those communications devices that support it
<code>call script</code>	Exit the current script and start another
<code>case match</code>	Indicate if "wait for" string must match in case
<code>closelog</code>	Close a log file
<code>connect</code>	Force a host connection if not connected
<code>debug</code>	Enable/disable debugging for scripts
<code>disconnect</code>	Force a disconnect if connected
<code>display</code>	Send text to the display
<code>echo</code>	Turn on/off echo of incoming characters
<code>execute</code>	Execute a Windows program
<code>exit</code>	Exit the script
<code>goto</code>	Transfer control to another place in the script
<code>if empty</code>	Transfer control if last test string was empty
<code>key_starttime</code>	Simulate Key timing start
<code>key_stoptime</code>	Simulate Key timing stop
<code>key_timer</code>	Turn Key Timing on and off

Command	Action
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
return	Return from a subroutine in the script file
send	Send text to the comm 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

3.5 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 will default 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>, ..., <Pn>

will substitute the *n*-th command line parameter in place of <Pn>.

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.

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

3.6 Individual Script Command Descriptions

This section contains a list of the individual script commands and their use.

3.6.1 Break

This command sends a "break" for those communications nodes that support a break. Otherwise, it acts as a "no-op." It takes no arguments.

Usage example:

```
break
```

3.6.2 Call Script

The **call script** command starts running a script. If a script is executing when this command is executed, the running script is terminated before starting the new script.

Usage example:

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

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

3.6.3 Case Match

The **case match** command enables/disables exact case matching in the wait for command.

Usage example:

```
case match: off
```

This permits matching of strings even if the individual characters differ from one another in case. The default for this switch is *on*.

3.6.4 Closeslog

This command closes a currently open log file. If no log file has been opened, this command has no effect.

Usage example:

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

3.6.5 Connect

This command opens a dialog box to initiate a connection to a remote host.

Usage example:

```
connect
```

3.6.6 Debug

The **debug** command is useful for catching invalid script commands. Normally, Caché Terminal ignores invalid script commands. 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
```

3.6.7 Disconnect

This command is the equivalent of the Disconnect choice on the Connect/Host menu, if Caché Terminal is currently connected to a host. If it is not connected, the command does nothing.

Usage example:

```
disconnect
```

3.6.8 Display

The **display** command outputs 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 command 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:", leaving the cursor after the end of the text.

3.6.9 Echo

The **echo** command enables/disables displayed output to the window and logfile. This is useful if work is to be done and it is desirable that it not appear on the screen or in the logfile.

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 the user clicks "Ok".

3.6.10 Execute

This command launches a Windows program with a `SHOW` attribute for its window.

Usage example:

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

The command executes the Windows Notepad program and open the file, `myfile.not`, inside the 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.

3.6.11 Exit

The **exit** command is used to exit 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
```

3.6.12 GoTo

The **goto** command is used to transfer 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 ...
```

3.6.13 If Empty

The **if empty** command 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 of these two commands determines if the first parameter provided in the command line is missing or empty. The second branches to the label "\$No First Arg" if this is the case.

3.6.14 Key_Starttime

This command is used to start the timing of a "key" sequence. It takes a single numeric argument. If the argument is zero, the statistics are accumulated when ENTER is typed. Otherwise, it is accumulated when F10 is pressed.

Usage example:

```
key_starttime: 0
```

Timing can be stopped with the **key_stoptime** command.

3.6.15 Key_Stoptime

This command stops a timing and accumulates statistics, if timing is currently active.

Usage example:

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

3.6.16 Key_Timer

The **key_timer** command starts/stops the data collection of key timing information. Alternatively, the timer can be started/stopped with the special key, 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 directory that contains a histogram of key timings. Only one such timing sequence can be used because it does not append to the current statistics file but overwrites it.

Note: If timings are to be driven exclusively from script files, you must use <CR> and <F10> in place of <13> and “<27>[21-”, respectively.

3.6.17 Logfile

This command 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 system manager directory.

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

3.6.18 MultiWait For

The **multiwait for** command 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 command 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 equal sign) is treated as the “separator” character which breaks up the argument into substrings. So this command causes the example script to wait until one of the sequences “USER>”, “***ERROR” or “DONE” arrives.

A timer can be used 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>
```

3.6.19 Notify

The **notify** command presents a Windows message box to the user and waits until the "OK" button is depressed before continuing. It can be used to inform the user of important events.

Usage example:

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

Note: This box is "modal", that is, cannot be interrupted except by the user.

3.6.20 On Error

The **on error** command provides a means to specify the label for script commands to be executed if the timer expires (normally while waiting for text to arrive).

Usage example:

- See the **goto** command for an example.

3.6.21 Pause

This command is used to pause a running script for a number of tenths of seconds.

Usage example:

```
pause: 30
```

The example pauses execution of the script for three seconds. A **pause** with 0 is equivalent to an indefinite pause which will require another Alt+P to resume the script processing.

3.6.22 Return

The **return** command is used with the **subroutine** command to return to the place in the script from which the subroutine was called.

Usage example:

- See the **subroutine** command for an example.

3.6.23 Send

The **send** command simulates typed data that is sent to the current connected host.

Usage example:

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

This command is equivalent to typing:

- the character, 1
- a carriage-return key
- the character, 2
- a carriage-return key
- the string of characters, "A1234"
- the F10 key
- one space character

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

3.6.24 Subroutine

A **subroutine** is useful if repetitive commands are used in a script. It saves both memory and the possible need to invent many different labels. This command should be used 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. An attempt to nest subroutine invocations deeper than that will result in the failure of the script.

3.6.25 Terminate

This command 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
```

3.6.26 Test

The **test** command is used to see if a parameter or window property is non-empty. The command is used in conjunction with the **if empty** command.

Usage example:

- See the **if empty** command for an example.

3.6.27 Timer

The **timer** command sets a timer to be used in conjunction 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
```

Its 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.

3.6.28 Title

The **title** command is used to change the window title to whatever string is required.

Usage example:

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

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

3.6.29 Wait For

The **wait for** command synchronizes the script file with data that arrives from the host.

Usage example:

```
wait for: USER>
```

This command 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 this command can be used to wait until Caché Terminal is ready for more input.

A timer can be used to break out of a **wait for** command.

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

4 Extended Keyboard Features

4.1 Key Mappings

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

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:

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.

4.2 Keyboard Functions

To speed the use of Caché Terminal, several hot keys have been defined to permit quick access to certain menu items. See the individual Menu Commands for information on these keys.

Topic	Summary
Key Timing	This is useful for determination of performance of a host system under various load conditions. The key timing is enabled/disabled via Alt+Shift+T. The output of a timing "run" is a file called KEYTIMER.LOG which can be found in the System manager directory.
Learn Mode	Learn mode is a feature that permits rapid prototyping of script files. Learn mode is enabled/disabled via Alt+Shift+L. When enabled, and logging is active, rather than simply capturing all incoming characters, the "log" file becomes 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.
Spy Mode	Spy mode is useful for DLL applications that are utilizing the communications capabilities of Caché Terminal. It is enabled/disabled via Alt+Shift+S and is mentioned here only as information should that key be typed. It is not for general use.
Pause Mode	Pause mode is used to pause a script file manually. It is enabled/disabled via Alt+P. It can also be activated with the pause command with an argument of 0.

5 DDE Overview

DDE is an acronym for Dynamic Data Exchange, an interprocess communication (IPC) system built into the Macintosh, Windows, and other operating systems. DDE enables two running applications to share the same data. For example, DDE makes it possible to insert a spreadsheet chart into a document created with a word processor. Whenever the spreadsheet data changes, the chart in the document changes accordingly. Although the DDE mechanism is still used by many applications, it is being supplanted by OLE (Object Linking and Embedding, a compound document standard developed by Microsoft Corporation), which provides greater control over shared data.

Caché Terminal supports DDE (Dynamic Data Exchange) links to permit other applications to utilize its communications ability to talk to a remote host.

There are three (non-System) topics supported by Caché Terminal. The discussion of each topic assumes that the user has knowledge of how to use DDE. The topics are:

- **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 has no way to discriminate between multiple instances of Caché Terminal when it comes to using DDE! It is most useful when only one copy of Caché Terminal is running.

5.1 DDE Layout Connections

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

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 VMS 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.

5.2 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.

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 "RnnCmmLpp" 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.

5.3 DDE Message Connections

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

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.

6 An Example Script

This section gives an example of a script session run on the local machine and invoked via the "batch" facility of Windows 2000.

Note: 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 it is typed by the user into a "batch" window (DOS prompt) or read as input in a batch script.

The following table gives the different input sequences required to represent the line

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

on the various operating systems.

OS	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

6.1 The Batch Command Line

You can invoke the Terminal application from the DOS command line (the Windows 2000 program, cmd.exe, to be precise). The general form of the command line is:

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

where:

Item	Meaning
cterm	The invocation of the Terminal application. If the Windows environment variable, <i>PATH</i> , has been set to include the location of the Caché binaries, then the command name may be given “cterm” or “cterm.exe”. Otherwise, a full or partial pathname must be used. For a default installation of Caché, it can be found in the directory, C:\CacheSys\Bin.
<ArgM>	One of the control arguments for Terminal (described separately).
<ScriptFilePath>	The location of the script file to be interpreted by the Terminal application.

6.2 Control Arguments

Several arguments modify the starting environment for the Terminal session. Some of these are reserved for internal use and/or debugging; they will not be described here. The ones useful to most invocations are:

6.2.1 /console=<ConnectionString>

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

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

/console=cn_iptcp:<HostAddr>

This specifies the target system that the Terminal is to interact with 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]
```

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

User applications may be run in the Terminal by specifying the configuration for the application. Assuming 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
```

starts a Terminal session and, if needed, a version of Caché from the “cache” configuration in the USER namespace. When this has been done successfully, the Terminal runs the ^%D routine which prints out the current date. When ^%D returns, the Terminal session is closed.

Configurations correspond to the names of the Caché parameter files in the System Manager directory without their 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 “]” is required only if a routine name is given; otherwise, it is omitted.

6.2.2 /server=<ServerName>

This argument specifies the name of the server to be used 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.

The characteristics of the connection are defined when the server entry is added to the list.

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

6.3 The Sample Script

The sample script looks like this:

```
;;;
;;; Script for Cache Terminal Example
;;;
; 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: znospace "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
```

The following sections show three different ways to run this script.

6.4 Running the Script, Example 1: Get Process Info (Batch)

This is an example of using 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`, the user runs 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
```

6.5 Running the Script, Example 2: Get Process Info (Interactive)

The same routine, `^%STACK`, as in the previous example can be invoked in a Terminal session via the Windows 2000 DOS command:

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

The user can manually provide the responses that the script supplied in that example. The Terminal window will be closed when the user replies to the prompt, “Stack Display Action:” by pressing the ENTER key.

6.6 Running the Script, Example 3: Manually Connecting To A Host

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 userid and password are available.

Start Terminal from a DOS window (Windows 2000 assumed).

```
C:\CacheSys\Bin\cterm.exe
```

Then do the following:

- Select Connect from the window menu bar, and then Host.
- In the dialog box which appears, enter “127.0.0.1” for the Remote System: address and “23” for the Port Number. Select OK. The Terminal application will attempt to connect to your local host via the TELNET port.
- You will see the “Username:” prompt from Caché. Enter “SYS” and press ENTER.

- Then, you will see the “Password:” prompt. Enter “XXX” and press ENTER.

If everything is in order, you will then see the prompt, “%SYS>” indicating that you are connected and have been placed in the %SYS namespace. You can replicate what you did in Example 2 by issuing the command,

```
DO ^%STACK
```

in response to the prompt, and then responding with “*S” to the request for a display action. Exit the ^%STACK routine by pressing ENTER in response to its prompt.

You may terminate the session by select Disconnect from the Connect menu. If you wish to terminate this instance of Terminal, select the Close box for the window.

Index

S

script files

- allowed commands, [12](#)
- example, [30](#)
- passing arguments to commands, [13](#)
- running, [11](#)

T

Terminal

- batch command line, [28](#)
- Connect menu, [10](#)
- Edit menu, [5](#)
- example script, [30](#)
- extended keyboard features, [23](#)
- File menu, [3](#)
- hot keys, [24](#)
- network encoding, [8](#)
- running script files, [11](#)
- technical overview, [2](#)
- using DDE, [25](#)

