



Caché MultiValue Commands Reference

Version 2008.2
10 October 2008

Caché MultiValue Commands Reference
Caché Version 2008.2 10 October 2008
Copyright © 2008 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.



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



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.

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.

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

1 MVIMPORT	1
1.1 Arguments	1
1.2 Description	2
1.2.1 Letter Code Options	2
1.3 Determining the Account Name	3
1.4 Errors and Log Files	3
2 MultiValue Commands	5
2.1 ; (semicolon)	5
2.2 [(left square bracket)	6
2.3 .X	7
2.4 ABORT	7
2.5 ASSIGN	7
2.6 AUTOLOGOUT	8
2.7 BASIC	8
2.8 BLOCK.PRINT	9
2.9 BLOCK.TERM	9
2.10 BREAK	10
2.11 BUILD.INDEX	10
2.12 CATALOG	11
2.13 CEMU	11
2.14 CENTURY.PIVOT	12
2.15 CHOOSE.TERM	12
2.16 CLEAR.FILE	12
2.17 CLEAR.LOCKS	13
2.18 CLEARDATA	13
2.19 CLR	13
2.20 COMO	14
2.21 COMPILE.DICT	14
2.22 COMPILE.TERM	14
2.23 COPY	15
2.24 COPY.LIST	15
2.25 COS	15
2.26 COUNT	16
2.27 CREATE.ACCOUNT	16
2.28 CREATE.FILE	17
2.29 CREATE.INDEX	18

2.30 CREATE.TRIGGER	18
2.31 CS	19
2.32 DATE	19
2.33 DATE.FORMAT	19
2.34 DECATALOG	20
2.35 DELETE	20
2.36 DELETE.ACCOUNT	21
2.37 DELETE.FILE	21
2.38 DELETE.INDEX	22
2.39 DELETE.LIST	22
2.40 DELETE.TRIGGER	22
2.41 DISPLAY	23
2.42 ED	23
2.43 EDIT	25
2.44 EDIT.LIST	25
2.45 ENABLE.BREAK.KEY	25
2.46 FORM.LIST	26
2.47 GET.LIST	26
2.48 HUSH	26
2.49 ICOMP	27
2.50 ICOMP.ALL	27
2.51 JOBS	28
2.52 KEYS	28
2.53 LIST	28
2.54 LIST.INDEX	29
2.55 LIST.ITEM	29
2.56 LIST.JOB	29
2.57 LIST.LOCKS	30
2.58 LISTF	30
2.59 LISTME	30
2.60 LISTPA	31
2.61 LISTPEQS	31
2.62 LISTPH	32
2.63 LISTPTR	32
2.64 LISTS	32
2.65 LISTU	33
2.66 LOGOFF	33
2.67 LOGOUT	34
2.68 LOGTO	34
2.69 MVI	35

2.70 MVIMPORT	35
2.71 NSELECT	35
2.72 P	36
2.73 PHANTOM	36
2.74 PQ.SELECT	37
2.75 PRINT.CATALOG	37
2.76 PTERM	37
2.77 Q	38
2.78 QSELECT	38
2.79 QUIT	38
2.80 RUN	39
2.81 SAVE.LIST	39
2.82 SEARCH	39
2.83 SELECT	40
2.84 SET.FILE	40
2.85 SETPTR	40
2.86 SH	42
2.87 SLEEP	42
2.88 SORT.LIST	42
2.89 SP.x Commands	43
2.90 SPOOL	43
2.91 STACKCOMMON	44
2.92 STATUS	44
2.93 TABS	45
2.94 TANDEM	45
2.95 TERM	46
2.96 TERM.TYPE	47
2.97 TIME	48
2.98 TRACE	48
2.99 UNASSIGN	49
2.100 WHERE	49
2.101 WHO	50
2.102 Z	50
2.103 ZH	51

1

MVIMPORT

Imports accounts from other MultiValue implementations to Caché MultiValue.

```
MVIMPORT backupfile [dirpath] [(code)]
```

1.1 Arguments

<i>backupfile</i>	The file containing the information to restore. This is assumed to be in the native “save” format of the originating system (for example, a UniVerse BACKUP). It can contain data for one or more accounts (namespaces).
<i>dirpath</i>	The location in which to restore directory files from a jBASE or UniVerse backup. If this option is omitted, the default is the current working directory.
<i>code</i>	A letter code governing the operation of MVIMPORT. Available values are A, C, L, M, O, and R.

MVIMPORT ignores any command line arguments it does not understand and is not prepared to handle.

1.2 Description

The **MVIMPORT** command used to move accounts from other MultiValue implementations to Caché. It does this by taking a backup file created by another MultiValue implementation and restoring it to Caché MultiValue. **MVIMPORT** supports backups created by UniVerse, jBASE, D3, Reality, MVBase, and Power95. The features of **MVIMPORT** vary according to the type of backup being restored.

MVIMPORT is designed to be a restoration utility for migrating legacy MultiValue accounts to Caché. Users should become familiar with Caché administrator facilities for backup, restore, and other administrative functions.

Because **MVIMPORT** is designed as a migration aid, it only recognizes programs and data when performing its task. Any indices present in *backupfile* are ignored. Native Caché facilities for indexing need to be used to recreate indices after the import. Caché classes are not created. It is the responsibility of the application developer to create and use Caché classes as needed.

Note: MultiValue account manipulation requires %Admin_Manage privileges. This is normally associated with the SYSPROG account. For information on the Caché security model, see the [Caché Security Administration Guide](#). For specific information on roles and privileges, please consult the chapter on [Roles](#) and the chapter on [Privileges and Permissions](#) in the *Caché Security Administration Guide*.

1.2.1 Letter Code Options

The following *code* values are supported:

- **A:** Prompts for an alternate account name for restoration instead of using the account name that was originally backed up. This takes precedence over the **R** option.
- **C:** Forces the account name to upper case. For D3 backups, file names are also forced to upper case.
- **Lnn:** Specifies the tape label size as *nn* bytes. This may be needed when the original backup was to tape which was then copied to an operating system file. Usually this value is "L80".
- **M:** (Merge) If the account exists in Caché, and the file exists in the account, **MVIMPORT** overwrites existing records if they are in *backupfile*, and leaves alone existing records that are not in *backupfile*. Any new records and files in *backupfile* are also imported.
- **O:** (Overwrite) If the account exists in Caché, **MVIMPORT** replaces it. The old account will be deleted, then recreated, and its records imported from *backupfile*.
- **R:** (Rename) If the account exists in Caché, **MVIMPORT** does not replace it. Instead, **MVIMPORT** imports a new account with a new name. The new name will be ExistingName_nn where *nn* is a sequence counter.

1.3 Determining the Account Name

The account name is found on the backup from D3, Reality, MVBase and Power95. For a UniVerse backup, the account name is the last directory name in the file path. jBASE does not provide the account name, so the user will be prompted for an account name at the start of the import.

If the account exists, and the M, O, or R options were not specified on the command line, **MVIMPORT** prompts for a response. You can respond with the M, O, or R letter code, which have the same effects as the (M), (O), or (R) command line *code* options. For example:

```
Account 'DBIHELPDESK' already exists.  
Enter 'M'erge, 'O'verwrite, 'R'ename or 'Q'uit:?
```

MVIMPORT creates a new account, namespace, and database for each account in the backup that does not already exist in Caché. All hash files (for example, database files) are restored to Caché globals in the created account. All non-hash files (for example, OS files in a directory) are restored to the home directory of the account, or as specified by the *dirpath* argument.

If **MVIMPORT** is merging accounts using the M option, when duplicate record keys exist in Caché and on the import data stream, the Caché values are overwritten.

MVIMPORT always creates a new account in the default location, as a sub-directory of the Intersystems\CacheMgr directory. If you want to create the account in any other location (for example, on another disk), then the procedure is as follows:

1. Prior to the **MVIMPORT**, create a blank MultiValue account with **CREATE-ACCOUNT**, specifying an alternate location. For example:

```
SYSPROG: CREATE-ACCOUNT MYACCOUNT UV E:\data\MYACCOUNT
```

2. Run **MVIMPORT** with the (M) option. This enables you to merge the account on the backup with the already created account.

1.4 Errors and Log Files

At the conclusion of a **MVIMPORT** operation (for all backups except UniVerse):

- Caché displays a summary of the number of errors and warnings. **MVIMPORT** saves the details of these errors and warnings and general restore statistics to an external file. It displays the filename of this error log file to the user.
- When **MVIMPORT** fails to create a namespace and/or a database, it displays an informative error message.

- All items found in *backupfile* are restored to the VOC, unless they already existed in the VOC.
- The file IMPORTED.VOC contains all the original VOC contents, regardless of whether or not **MVIMPORT** restored the items to the VOC. This allows you to see what the VOC looked like before **MVIMPORT** made decisions on whether or not to apply items to the VOC.
- **MVIMPORT** looks for object code in D3 backups. While this is discarded on Caché because it cannot be used, its presence indicates which files contain compiled code. This fact is used to mark the file as BASIC source for use with Caché Studio.

2

MultiValue Commands

This chapter provides an alphabetical listing of the command line commands supported by the Caché MultiValue Shell. In MultiValue database systems, these commands are also known as “verbs”. Several of these commands have an alternate name that includes a **.VERB** suffix. These suffix forms are not listed here.

Most command names that contain punctuation exist in two variant forms: the hyphen form and the dot form. For example, **CREATE-ACCOUNT** and **CREATE.ACCOUNT**. InterSystems supports these two forms for compatibility with different vendor versions of MultiValue code. In most cases, these two forms are synonymous. When this is the case, only one of the forms is listed in this chapter. The most notable exception is **SP-EDIT** and **SP.EDIT**, which provide different functionality.

In most cases, only one MultiValue command can be specified on a MVShell command line. You can, however, specify multiple **;** commands on the same command line.

2.1 ; (semicolon)

The **;** command causes the statement following it on the command line to be interpreted as an MVBasic statement.

```
; basicstatement
```

This MVBasic statement is immediately executed and the results returned to the terminal.

The **;** can be immediately followed by an MVBasic statement, or one or more spaces can be placed between them. Unlike most MultiValue shell commands, you can specify multiple **;** commands on the same command line. Each MVBasic statement requires its own **;** command.

Thus, the following is a valid use of the **;** command:

```
USER: ; PRINT "hello" ;PRINT "world"
```

You can issue the `;` command either from the MultiValue shell prompt, as shown above, or from the MultiValue debug prompt following an MVBASIC `DEBUG` statement, as in the following example:

```
USER:;myvar="ABC"  
USER:;DEBUG  
  
<BREAK>+1^MVBASIC1048.mvi  
Source Id: File: Line:0  
USER 7d1>;CRT "my variable",myvar  
my variable      ABC  
USER 7d1>
```

See Also: BASIC, RUN

2.2 [(left square bracket)

The `[` command causes the statement following it on the command line to be interpreted as a Caché ObjectScript command line.

```
[ commandline
```

A Caché ObjectScript command line can consist of one or more Caché ObjectScript statements, separated by blank spaces. This Caché ObjectScript command line is immediately executed and the results returned to the MultiValue prompt at the Caché Terminal.

The `[` can be immediately followed by a Caché ObjectScript command line, or one or more spaces can be placed between them.

If *commandline* changes the current namespace (for example, by issuing a `ZNSPACE` command), the Caché MultiValue Shell restores the initial namespace upon exiting ObjectScript.

The following is an example of the `[` command:

```
USER:[ SET x="Hello World!" WRITE !,x
```

The `[` command is functionally identical to the `COS` command.

See Also: COS

2.3 .X

The **.X** command performs command line recall.

```
.X[n]
```

The command **.X** (or **.X1**) retrieves the previous command line. The command **.Xn** retrieves a command line earlier in the command stack, with the integer *n* specifying how many previous line to count back. Thus, **.X4** retrieves the command line issued four lines ago. Note that issuing **.X** or **.X1** does not increment the command line count, whereas issuing **.Xn** (where *n* is ≥ 2) does increment the command line count by 1.

2.4 ABORT

The **ABORT** command terminates the current process and returns to either the MV Shell or the EXECUTE command that invoked the process.

```
ABORT (expr ( , expr ) )
```

2.5 ASSIGN

The **ASSIGN** command assigns a form queue spool device to a LPTR device. The **-WAIT** keyword is a no-op.

```
ASSIGN form-queue TO LPTR n [-WAIT]
```

The *form-queue* can be specified either by name or by number. The default form queue has the name STANDARD, and a form queue number of 0. It can be specified as “STANDARD”, “0”, “F0”, “FN0”, or “FQ0”. The *n* device number variable can take an integer value between 0 and 255 (inclusive).

See Also: UNASSIGN

2.6 AUTOLOGOUT

The **AUTOLOGOUT** command sets and displays the time setting for automatic logout.

```
AUTOLOGOUT [minutes]
```

Specifying **AUTOLOGOUT** with no operand returns the current autologout setting: either “Automatic logout is set for *x* minutes” or “Automatic logout is disabled”. The default is “disabled”.

AUTOLOGOUT with no operand returns the most recent setting; it does not return the number of minutes remaining.

Use the optional *minutes* argument to set an automatic logout time. Automatic logout can be set to a positive integer number of minutes. A fractional number of minutes is truncated to its integer portion. Setting *minutes* to 0, a fraction less than 1, or a negative number disables automatic logout.

2.7 BASIC

The **BASIC** command compiles an MVBasic program stored in a MVBasic file.

```
BASIC filename[,datasection] [programs | *] [options]
```

The *filename* argument specifies a file created using **CREATE.FILE** and edited using **ED**. Specify *datasection* if the program to be compiled is stored in the data section of the file. The *programs* argument specifies the record ID of the program to be compiled. A * specifies that all programs in the file should be compiled. The optional *options* argument can take the values: S (create and spool a program listing), X (create a cross-reference listing), or Z.

After compiling a program, you can catalog it using the **CATALOG** command. To run an MVBasic program, use the **RUN** command.

See Also: ; (semicolon), CATALOG, MVI, RUN

2.8 BLOCK.PRINT

The **BLOCK.PRINT** command prints a text as large-format letters.

```
BLOCK.PRINT text
```

You can specify any printable character(s) as *text*, including a text containing blank spaces.

BLOCK.PRINT uses multiple “X” characters to represent each character of *text* as a large block character, inserting line breaks on blank spaces when needed. An error message is returned if a string of characters (excluding blanks) is too long to print on one line.

A *text* does not require delimiters. It can optionally be delimited by either single quotes (') or double quotes ("). These delimiter characters do not print. To include one of these delimiter characters as part of the printed string, enclose the string with the other type of delimiter character. For example:

```
"won't".
```

The **BLOCK.PRINT** command outputs to the current printer. The **BLOCK.TERM** command outputs to the current terminal. These commands are otherwise identical.

See Also: **BLOCK.TERM**

2.9 BLOCK.TERM

The **BLOCK.TERM** command displays a text as large-format letters.

```
BLOCK.TERM text
```

You can specify any printable character(s) as *text*, including a text containing blank spaces.

BLOCK.TERM uses multiple “X” characters to represent each character of *text* as a large block character, inserting line breaks on blank spaces when needed. An error message is returned if a string of characters (excluding blanks) is too long to display on one line; by default this maximum string length is 9 characters.

A *text* does not require delimiters. It can optionally be delimited by either single quotes (') or double quotes ("). These delimiter characters do not display. To include one of these delimiter characters as part of the displayed string, enclose the string with the other type of delimiter character. For example:

```
"won't".
```

The **BLOCK.TERM** command outputs to the current terminal. The **BLOCK.PRINT** command outputs to the current printer. These commands are otherwise identical.

See Also: **BLOCK.PRINT**

2.10 BREAK

The **BREAK** command (and its variants) enable or disable terminal keys that can pause program execution.

```
BREAK [ON | OFF]
BREAK.KEY.ON
BREAK.KEY.OFF
BREAK.KEY.ENABLE
ENABLE.BREAK.KEY
```

When **BREAK** is enabled (ON), the Interrupt, Suspend, and Quit keys will cause program execution to be suspended. When **BREAK** is disabled (OFF) these keys have no effect on program execution. **BREAK** is enabled by default. The same operation is performed by the MVBasic **BREAK** statement.

Issuing any of these statements increments or decrements a counter. Thus multiple **BREAK OFF** statements (of any type) must be reversed by an equal number of **BREAK ON** statements.

In jBASE emulation, these statements simply enable or disable (toggle) without maintaining a counter.

See Also: MVBasic **BREAK** statement.

2.11 BUILD.INDEX

The **BUILD.INDEX** command builds (populates) either an index for a specified file attribute, or indices for all of the attributes of the file.

```
BUILD.INDEX filename attribute | ALL | *
```

You can specify a single *attribute*, or a series of *attributes*, separated by blank spaces. You can use either the ALL keyword or the asterisk (*) to build all indices.

If the *attribute* is not specified, **BUILD.INDEX** returns a [211] message. If *attribute* is invalid, or there are no indices defined for this *filename*, **BUILD.INDEX** returns a [842] message. If the index has already been built, **BUILD.INDEX** overwrites the previous index data.

Before you can build an index, you must create the index for *filename* using **CREATE.INDEX**.

See Also: **CREATE.INDEX**, **LIST.INDEX**

2.12 CATALOG

The **CATALOG** command catalogs a program and stores it in the VOC as a verb.

```
CATALOG [filename[,datasection]] [catalog [program] ] [options]
```

CATALOG stores *program* in the &ROUTINES& file of the VOC. Before cataloging a program, compile it using the **BASIC** command.

filename is the file to search for *program*. Specify *datasection* if the program is stored in a data section of the file. *catalog* is the external program name (the record ID in the VOC). *program* is the name of the program (subroutine). *options* is a letter code: L = local catalog, N = normal catalog.

See Also: BASIC, CREATE.FILE, DECATALOG

2.13 CEMU

The **CEMU** command changes the emulation of the current account.

```
CEMU [emulation]
```

Specifying **CEMU** with no operand returns the current emulation. For example, “Emulation for account 'USER' is 'CACHE'.”

Specifying **CEMU** with an operand sets the current emulation, and returns a message. The available *emulation* values are: Cache, D3, IN2, INFORMATION, JBASE, MVBase, PICK, PIOpen, Prime, R83, R95, Reality, Ultimate, UniData, and UniVerse. The *emulation* argument is not case-sensitive. Both “Prime” and “Information” *emulation* values set an emulation of “INFORMATION.” An *emulation* value of “Default” sets an emulation of “CACHE”. An invalid *emulation* value returns an 812 error.

You can also specify emulation within MVBasic by using the **\$OPTIONS** command, as described in the *Caché MultiValue Basic Reference*.

For further details, refer to the **CEMU** section of the *Operational Differences between MultiValue and Caché* manual.

See Also: CREATE.ACCOUNT

2.14 CENTURY.PIVOT

The **CENTURY.PIVOT** command specifies how two-digit year values are interpreted. By default, a two-digit year is interpreted as being in the range 1900 to 1999. You can use **CENTURY.PIVOT** to set any hundred-year range for two-digit years.

```
CENTURY . PIVOT [year]
```

Specifying **CENTURY.PIVOT** with no operand returns the current date range for two-digit years. To set the date range, specify a four-digit *year* as the beginning year of the century range.

2.15 CHOOSE.TERM

The **CHOOSE.TERM** command allows you to choose a terminal type. It displays a list of supported terminal types, then prompts you to specify the desired terminal type.

```
CHOOSE . TERM
```

For further details, refer to the [Terminal Output](#) chapter of the *Caché MV Terminal Independence* manual, and the **CHOOSE.TERM** section of the *Operational Differences between MultiValue and Caché* manual.

See Also: **COMPILE.TERM**, **TERM**

2.16 CLEAR.FILE

The **CLEAR.FILE** command clears all data from a file.

```
CLEAR.FILE [DATA | DICT] filename
```

The optional **DICT** and **DATA** keywords enable you to specify a dictionary file or a data file. The default is a data file.

If the *filename* is not specified, **CLEAR.FILE** returns a [200] message. If the *filename* is not valid, **CLEAR.FILE** returns a [201] message. If the *filename* is valid, **CLEAR.FILE** returns a [433] message indicating that the file has been cleared, even if there was no data in the file.

See Also: **DELETE**

2.17 CLEAR.LOCKS

The **CLEAR.LOCKS** command clears locks that are held by the current process.

```
CLEAR . LOCK [ nnn ]
```

The optional *nnn* argument enables you to specify a specific lock. The default is to release all locks held by the process. **CLEAR.LOCKS** does not release system locks.

Locks are established using the MVBasic **LOCK** command or by opening a sequential file.

See Also: LIST.LOCKS

2.18 CLEARDATA

The **CLEARDATA** command clears the data stack.

```
CLEARDATA
```

2.19 CLR

The **CLR** command clears the screen and sets the cursor to the first line.

```
CLR
```

The **CLR** and **CS** commands are synonyms.

See Also: CS

2.20 COMO

The **COMO** command copies terminal output to a record in the **&COMO&** file.

```
COMO ON record [HUSH]
COMO OFF COMO LIST
COMO DELETE record | *
COMO SPOOL record
```

If you specify **COMO** with no arguments, it returns a series of prompts requesting an option keyword and the record name.

COMO ON creates the specified record in the **&COMO&** file. Caché stores the **&COMO&** file using the **^COMO** global. The optional **HUSH** keyword suppresses terminal display. **COMO DELETE** deletes the specified **&COMO&** file record. **COMO DELETE *** deletes all the **&COMO&** file records.

2.21 COMPILE.DICT

The **COMPILE.DICT** command compiles I-descriptors in dictionary records.

```
COMPILE.DICT filename [attributes]
```

2.22 COMPILE.TERM

The **COMPILE.TERM** command compiles terminal definitions.

```
COMPILE.TERM [filename [item-list | *] [(VT)]]
```

COMPILE.TERM, with no arguments, compiles all the record definitions in the **%MV.TERMDEFS** file. **COMPILE.TERM filename** compiles all records in the *filename* file. The "T" option performs a tree search on *filename*; this assumes that *filename* is a directory in Unix/UniVerse format. For example: **COMPILE.TERM //C:/terminfo (T)**. Note the use of the **//** prefix to directly reference a directory.

For further details, refer to the [Terminal Definition](#) chapter of the *Caché MV Terminal Independence* manual.

See Also: **CHOOSE.TERM**, **TERM**

2.23 COPY

The **COPY** command copies items from one file to another file.

```
COPY [FROM] sourcefile [TO targetfile] [item1 [item2] [item-n] | ALL]
```

If you omit the TO clause, **COPY** prompts you for the destination file. If you omit the *item* list (or the ALL keyword), **COPY** prompts you for the Item Id.

If the specified *sourcefile* or *targetfile* is not valid, **COPY** returns a [201] message.

2.24 COPY.LIST

The **COPY.LIST** command copies a save select list from the &SAVEDLISTS& file.

```
COPY.LIST [filename [listname] ]
```

The optional *filename* is the destination file where the select list is to be copied. If you omit *filename*, **COPY.LIST** prompts you for the destination file name. The optional *listname* is the name of an existing select list; the default is select list 0. The *listname* select list is saved in the &SAVEDLISTS& file. Caché stores this file using the ^SAVEDLISTS global.

See Also: DELETE.LIST

2.25 COS

The **COS** command issues a Caché ObjectScript command.

```
COS commandline
```

COS issues a Caché ObjectScript command without exiting the MultiValue Shell. The *commandline* can be any valid Caché ObjectScript command line. A *commandline* cannot be specified as a variable, nor can it be enclosed in quotation marks.

If *commandline* changes the current namespace (for example, by issuing a ZNSPACE command), the Caché MultiValue Shell restores the initial namespace upon exiting ObjectScript.

You can use **SH** to issue an operating system command without exiting the MultiValue Shell.

The COS command is functionally identical to the [command.

See Also: [, SH

2.26 COUNT

The **COUNT** command counts the number of items in a file and returns the number.

```
COUNT filename
```

Specifying an invalid *filename* returns a [200] message. Specifying a *filename* that contains no items returns a [401] message.

See Also: LIST

2.27 CREATE.ACCOUNT

The **CREATE.ACCOUNT** command creates an account (namespace).

```
CREATE.ACCOUNT account [emulation] [directory]
```

The *account* is the name to assign to the namespace and the account. An account name can consist of letters, numbers, and the percent (%), hyphen (-), and underbar (_) characters. Percent (%) can only be used as the first character; hyphen (-) and underbar (_) cannot be used as the first character. Caché creates a corresponding namespace, as follows:

- Caché strips out internal punctuation characters.
- Caché namespace names cannot begin with a number as the first character. If the account name begins with a number, Caché appends a % character to the *account* name to create the corresponding namespace name.
- Caché converts the resulting name to all uppercase characters. Namespace names are not case sensitive, but account names are case sensitive.

If you have specified an account name that differs from an existing account name only in capitalization or punctuation, Caché creates a unique namespace name by appending an underbar and sequential number, starting with the “_1” suffix, for the second account.

Note that the SYSPROG account corresponds to the %SYS namespace.

If the specified account already exists, **CREATE.ACCOUNT** returns an [810] message. Upon successful completion, **CREATE.ACCOUNT** returns an [814] message.

The optional *emulation* argument specifies the MultiValue emulation type. The default is Cache.

The optional *directory* argument specifies the location in which to create the account database. Specify a fully-qualified pathname. The default is \Mgr\accountname in the Caché installation location.

Note: Account creation and deletion requires **%Admin_Manage** privileges. This is normally associated with the SYSPROG account, which is the %SYS namespace. For information on the Caché security model, see the [Caché Security Administration Guide](#). For specific information on roles and privileges, please consult the chapters on [Roles](#) and [Privileges and Permissions](#).

See Also: CEMU, DELETE.ACCOUNT, LOGTO

2.28 CREATE.FILE

The **CREATE.FILE** command creates a file.

```
CREATE.FILE [DATA | DICT] filename[,datasection] [ANODE]
```

The *filename* is the name of the file, which is created as a Caché global (^filename). The optional DICT and DATA keywords enable you to specify a dictionary file or a data file. The default is a data file. Specify *datasection* if you are creating a file with a data section. The *datasection* argument can only be specified for a data file; attempting to specify it for a dictionary file returns a [424] message.

You can create an &SAVELISTS& or &HOLD& file as an ANODE type file (attribute of an item node), as follows:

```
USER:DELETE.FILE &SAVEDLISTS&
USER:CREATE.FILE &SAVEDLISTS& ANODE
```

By default, **CREATE.FILE** creates a file of type INODE (item node).

You can create an &SAVELISTS& file as a DIR type file in an account, as follows:

```
USER:DELETE.FILE &SAVEDLISTS&
USER:CREATE.FILE &SAVEDLISTS& DIR SAVEDLISTS
```

Caché stores the &SAVEDLISTS& file using the ^SAVEDLISTS global. If the specified *filename* already exists, **CREATE.FILE** returns a [413] message.

See Also: CATALOG, CLEAR.FILE, DELETE.FILE

2.29 CREATE.INDEX

The **CREATE.INDEX** command creates an index on a file.

```
CREATE.INDEX [USING dictname | DICT] filename
```

The *filename* is the name of the file, which is created as a Caché global (^filename). The optional USING *dictname* clause enables you to specify a dictionary file. The default is the DICT dictionary file.

After creating an index, you can populate it using **BUILD.INDEX**.

For further details, refer to the [CREATE.INDEX](#) section of the *Operational Differences between MultiValue and Caché* manual.

See Also: BUILD.INDEX, DELETE.INDEX, LIST.INDEX

2.30 CREATE.TRIGGER

The **CREATE.TRIGGER** command creates a trigger that calls a subroutine when a file event of a specified type occurs.

```
CREATE.TRIGGER filename event subroutine
```

The *filename* is the name of an existing file. The trigger is specific to events occurring to this file. The *event* is the operation that invokes the trigger. It can be one of the following: *, POSTOPEN, PREREAD, PREINSERT, PREUPDATE, PREWRITE, PREDELETE, PRECLEAR (or PRECLEARFILE), POSTREAD, POSTINSERT, POSTUPDATE, POSTWRITE, POSTDELETE, POSTCLEAR (or POSTCLEARFILE). To invoke the trigger upon any of the event types, specify an asterisk (*) as the *event* value. The trigger code that is executed when the trigger is pulled is located in *subroutine*, which is an MVBasic subroutine.

Specifying an invalid *filename* returns a [201] message. Specifying a *subroutine* that has not yet been cataloged returns a [825] message.

See Also: DELETE.TRIGGER

2.31 CS

The **CS** command clears the screen and sets the cursor to the first line.

```
CS
```

The **CS** and **CLR** commands are synonyms.

See Also: CLR

2.32 DATE

The **DATE** command returns the current date and time.

```
DATE
```

For example: “Tuesday, October 23, 2007 02:55pm”, The actual date and time format is governed by the **DATE.FORMAT** command.

See Also: DATE.FORMAT

2.33 DATE.FORMAT

The **DATE.FORMAT** command specifies the format used for displaying dates.

```
DATE.FORMAT [ON | OFF] [(I) | (D)]
DATE.FORMAT INFORM
```

The optional **ON** keyword specifies international date order; for example: “Tuesday, 23 October 2007 02:55pm”, The optional **OFF** keyword specifies USA date order; for example: “Tuesday, October 23, 2007 02:55pm”, The default is **ON**. The optional **(I)** and **(D)** arguments are equivalent to the **ON** and **OFF** arguments; the parentheses are mandatory. Specify either **ON/OFF** or **(I)/(D)**, not both.

The **INFORM** option sets `@SYSTEM.RETURN.CODE` to the current date format setting: 0 (USA date order) or 1 (international date order).

For further details, refer to the [DATE.FORMAT](#) section of the *Operational Differences between MultiValue and Caché* manual.

See Also: DATE

2.34 DECATALOG

The **DECATALOG** command removes a cataloged program.

```
DECATALOG [filename[,datasection]] [program | * ] [option]
```

DECATALOG may be run against the VOC or against a file of source code. When run against the VOC, it deletes the specified VOC item, as well as any copies generated for normal catalog. If run against a source file, it deletes any catalog pointers for the specified file, as well as normal and global catalog copies. If you specify the *A option*, it also deletes the compilation object code and removes it from the &ROUTINES& file.

filename is the file to search for *program*. Specify *datasection* if the cataloged program is stored in the data section of the file. *program* is the name of the program (subroutine) to decatalog. *option* is a letter code: A = delete all, g = global catalog, L = local catalog, N = normal catalog, V = verbose mode.

See Also: CATALOG

2.35 DELETE

The **DELETE** command deletes a record from a file.

```
DELETE [DATA | DICT] filename recordname [recordname]
```

The *filename* is the name of an existing file. The optional **DICT** and **DATA** keywords enable you to specify a dictionary file or a data file. The default is a data file. You can specify one or more than one *recordname*, separated by spaces.

If the specified *filename* is not valid, **DELETE** returns a [201] message. If *recordname* is not present in the file, **DELETE** returns a [202] message. If no items are deleted [430] is returned. If one item is deleted [431] is returned. If more than one item is deleted [432] is returned.

See Also: CLEAR.FILE, DELETE.FILE

2.36 DELETE.ACCOUNT

The **DELETE.ACCOUNT** command deletes an account (namespace) and all of the MultiValue files within it.

```
DELETE.ACCOUNT account
```

account is an existing MultiValue account name, which has been assigned to a corresponding Caché namespace. **DELETE.ACCOUNT** deletes all of the MultiValue globals and routines found in *account*, and, if the namespace is empty, deletes *account*. **DELETE.ACCOUNT** *does not* delete globals or routines in the namespace that were not created by MultiValue operations. If the account is not associated with a namespace (for example, a synonym account) an error message is returned.

This is a restricted command. You must be logged in to the SYSPROG account to delete an account; use the **LOGTO** command to log in to SYSPROG. You cannot delete the SYSPROG account, which is the %SYS namespace. Attempting to do so results in an error message.

See Also: CREATE.ACCOUNT, LOGTO

Note: Account creation and deletion requires **%Admin_Manage** privileges. This is normally associated with the SYSPROG account (the %SYS namespace). For information on the Caché security model, see the [Caché Security Administration Guide](#). For specific information on roles and privileges, please consult the chapters on [Roles](#) and [Privileges and Permissions](#).

2.37 DELETE.FILE

The **DELETE.FILE** command deletes a file and its VOC entry.

```
DELETE.FILE [DATA | DICT] filename
```

The *filename* is the name of an existing file, which was created as a Caché global (^filename). The optional DICT and DATA keywords enable you to specify a dictionary file or a data file. The default is a data file.

See Also: CLEAR.FILE, CREATE.FILE, DELETE

2.38 DELETE.INDEX

The **DELETE.INDEX** command deletes an index on a file.

```
DELETE.INDEX filename
```

The *filename* is the name of the file, which is created as a Caché global (^filename).

For further details, refer to the [CREATE.INDEX](#) section of the *Operational Differences between MultiValue and Caché* manual.

See Also: CREATE.INDEX, LIST.INDEX

2.39 DELETE.LIST

The **DELETE.LIST** command deletes a select list from the &SAVEDLISTS& file.

```
DELETE.LIST listname | *
```

The *listname* is the name of an existing select list in &SAVEDLISTS&. The asterisk (*) argument deletes all select lists in &SAVEDLISTS&. Caché stores the &SAVEDLISTS& file using the ^SAVEDLISTS global.

See Also: COPY.LIST

2.40 DELETE.TRIGGER

The **DELETE.TRIGGER** command deletes a specified type of trigger from a file. It removes the trigger specification, not the trigger subroutine itself.

```
DELETE.TRIGGER filename event
```

The *filename* is the name of an existing file. The *event* is one of the following: *, POSTOPEN, PRE-READ, PREINSERT, PREUPDATE, PREWRITE, PREDELETE, PRECLEAR, POSTREAD, POSTINSERT, POSTUPDATE, POSTWRITE, POSTDELETE, POSTCLEAR. To delete all triggers of any type, specify an asterisk (*) as the *event* value.

Specifying a valid *filename* and *event* returns a [828] message, indicating that the trigger has been deleted, even if the specified trigger does not exist. Specifying an invalid *filename* returns a [201] message. Specifying an invalid *event* returns a [824] message.

See Also: CREATE.TRIGGER

2.41 DISPLAY

The **DISPLAY** command displays a line of text on the terminal screen.

```
DISPLAY [text]
```

If *text* is omitted, this command displays the empty string (a blank line).

2.42 ED

The **ED** command allows you to edit a record in a file.

```
ED filename[,datasection] record
```

ED is the MultiValue line editor. The *filename* is the name of the file to edit, which is created as a Caché global (^filename). If the specified *filename* is not valid, **ED** returns a [201] message. Specify *datasection* if the program to be edited is stored in the data section of the file. If *record* is an existing record, **ED** positions the current line pointer to the beginning of this record. If *record* is not an existing record, **ED** creates the specified record.

The **ED** command provides a prompt for editing the current record. You can issue a variety of subcommands at the **ED** prompt. For a list of these subcommands, specify **HELP** at the **ED** prompt. For a list of the attributes of the specified *record*, specify **?** at the **ED** prompt.

The following is a list of the **ED** subcommands supported by Caché:

```
-[nnn] Decrement current line pointer.
+[nnn] Increment current line pointer.
? Display information about the record.
! Execute an external MV command from within ED.
< Marks the beginning of a block of text for COPY, DROP, and MOVE.
> Marks the end of a block of text for COPY, DROP, and MOVE.
nnn Set current line pointer.
A text Append text to end of line.
B Set current line pointer to Bottom of record.
B [string] Break line at position of string.
C Synonym for R subcommand.
COPY Copy a block of text to another location in the record.
D[nn] Delete one or more lines of text.
DE[nn] Delete one or more lines of text.
```

MultiValue Commands

DELETE Delete the record from the file.
DROP Delete a block of text from the current record.
EX[KO] Exit editing this record.
FD[KO] Delete the record from the file and release lock.
FI[K] File record and exit editing this record.
FILE File record and exit editing this record.
FORMAT Indents code for ease of reading.
FS File record and continue editing this record, maintain lock.
HELP Display help screen.
HEX Display characters in hexadecimal, insert hexadecimal values.
I [string] Insert mode or insert string at current line pointer.
IB [string] Insert before mode or insert string before current line.
L Repeat locate (or list one line if no previous locate).
L string Locate next occurrence of string.
Lnnn List the next nnn lines.
Lnnn string Locate all strings in the next nnn lines.
LOAD [filename] record Load lines from another record into the current record.
MOVE Move a block of text from its current location to a new location in the current record.
OOPS Undo the previous edit.
P[n[Command]] Prestore command, set or execute.
Q Exit editing this record.
QUIT Exit editing this record.
R Repeat replace.
R newtext Replace entire line with 'newtext'.
R/from/to/[Gnn] Replace string 'from' with 'to' for the next nn lines.
SIZE Display information about size of record.
T Set current line pointer to Top of record.
TB n[,n] Set one or more tab stops.
UNDO Undo the previous edit.
UNLOAD [filename] record Load lines from the current record into another record.
XEQ Execute an external MV command from within ED.

Additional notes on subcommands:

- **DE**: When you delete a line of code, the current line pointer moves upward. Following a line delete, ED displays the current line. As this current line pointer behavior is not consistent across all MultiValue systems, exercise caution when performing repeated line deletions.
- **HEX**: A mode toggle to turn on or off display of text in hexadecimal. HEX mode only affects character display and the I (insert) subcommand; it does not affect other subcommands.
- **I**: To insert a blank line, specify the I subcommand followed by a blank space. To insert a hexadecimal value, go into HEX mode. To insert an @VM character (CHAR(253)) specify **Ctrl-J**. To insert an @SM character (CHAR(252)) specify **Ctrl-A**.
- **I** and **R**: You can use the ^ up-arrow code operator in an insert (I) or replace (R) subcommand to specify a single character by its integer code (values 000 through 255). This can be used to specify a MultiValue delimiter, such as ^253 for a value mark. However, you cannot specify a ^254, because this specifies a field mark.

An Escape character typed using **ED** is echoed (displayed) as "[".

To abort **ED** and release all locks, use **Ctrl-C**.

ED is a simple line editor, a subset of the line editors supplied with other MultiValue systems. For more complex editing, the user should use the Caché Studio.

The **ED** and **EDIT** commands are synonyms.

See Also: CREATE.FILE, EDIT, EDIT.LIST

2.43 EDIT

The **EDIT** command allows you to edit a record in a file.

```
EDIT filename record
```

The **EDIT** and **ED** commands are synonyms.

See Also: ED

2.44 EDIT.LIST

The **EDIT.LIST** command allows you to edit a select list in **&SAVEDLISTS&**.

```
EDIT.LIST [listname]
```

EDIT.LIST allows you to create or modify a select list, using the Caché MultiValue command line editor (**ED**) prompts. **EDIT.LIST** is the same as **ED &SAVEDLISTS& recID**, where *recID* is the record ID of a saved list. If you do not specify a *listname*, you are prompted to supply one. Caché stores the **&SAVEDLISTS&** file using the **^SAVEDLISTS** global.

See Also: CREATE.LIST, ED

2.45 ENABLE.BREAK.KEY

The **ENABLE.BREAK.KEY** and **BREAK ON** commands are synonyms.

2.46 FORM.LIST

The **FORM.LIST** command allows you to create a select list from elements stored in a record.

```
FORM.LIST [filename] [recID] TO [listnum]
```

FORM.LIST take the record identified by *recID* from *filename*, and uses it to create a numbered select list *listnum*. If you omit *filename* or *recID* you are prompted to supply one. Select lists are numbered 0 through 10; if the **TO listnum** argument is omitted, it defaults to select list 0.

See Also: **CREATE.LIST**, **GET.LIST**

2.47 GET.LIST

The **GET.LIST** command copies the specified named select list into a numbered select list.

```
GET.LIST listname [TO n]
```

The *listname* argument specifies a named select list. The optional **TO n** argument is a select list number in the range 0 through 10 (inclusive); if omitted, select list 0 is the default.

If *n* is out of range, **GET.LIST** returns a [209] message.

GET.LIST is the inverse of **SAVE.LIST**. To copy Select List 0 to another numbered select list, use the **PQ.SELECT** command.

See Also: **PQ.SELECT**, **SAVE.LIST**

2.48 HUSH

The **HUSH** command suppresses terminal screen display. It suppresses all terminal display, including displaying the terminal prompt.

```
HUSH [ON | OFF]
```

Specifying **HUSH** with no operand toggles display suppression. **HUSH ON** suppresses display. **HUSH OFF** re-enables display.

The **KEYS** command temporarily overrides the **HUSH**. However, when **KEYS** times out, it returns to the prior **HUSH** mode. This may be mistaken for a hang state.

The **HUSH** and **P** commands are synonyms.

See Also: DISPLAY, P

2.49 ICOMP

The **ICOMP** command compiles the I-type dictionary definitions in the specified file.

```
ICOMP filename [* | item[,item[,...]]]
```

ICOMP compiles the I-type dictionary definitions in *filename*. If you specify no item argument, **ICOMP** prompts you for the I-type item names. If you specify *, **ICOMP** compiles all I-type items in the file. If you wish to compile specific I-type items, specify the item names as a comma-separated list.

See Also: ICOMP.ALL

2.50 ICOMP.ALL

The **ICOMP.ALL** command compiles all the I-type dictionary definitions in one or more accounts.

```
ICOMP.ALL [accountname] [( [A][V] )]
```

ICOMP.ALL with no arguments compiles all the I-type dictionary definitions in the current account (namespace). **ICOMP.ALL** with the optional *accountname* argument compiles all the I-type dictionary definitions in the specified account (namespace). **ICOMP.ALL** with the optional (A) option compiles all the I-type dictionary definitions in all accounts (namespaces). You cannot specify both *accountname* and the (A) option. The (V) option returns verbose output while the command executes.

See Also: ICOMP

2.51 JOBS

The **JOBS** command list all running phantom processes initiated by the current process.

```
JOBS
```

JOBS lists all running processes initiated by the current process; **LIST.JOB** lists all running processes. You can use the **PHANTOM** command to initiate a phantom process.

See Also: LIST.JOB, PHANTOM

2.52 KEYS

The **KEYS** command sets the terminal into a mode in which each keyboard input character is displayed, along with its hexadecimal and ASCII base-10 equivalents. Compound keys (for example, the F1 key) return their component characters, one line per character. The **KEYS** command is designed to display all characters, including those that would normally terminate input. For this reason, the only way to exit **KEYS** mode is by timing out. This mode terminates automatically after 10 seconds of inactivity.

```
KEYS
```

The **KEYS** mode overrides the **HUSH** mode. However, when **KEYS** times out, it returns to the prior **HUSH** mode. This may be mistaken for a hang state.

For further details, refer to the [Terminal Input](#) chapter of the *Caché MV Terminal Independence* manual.

2.53 LIST

The **LIST** command lists the items in the specified file.

```
LIST filename
```

After listing each full page of items, **LIST** issues a prompt to the user to display the next page. To terminate listing, specify Q at the display prompt.

LIST begins each page of its listing with the current date and time, and ends the listing with the total number of items listed.

If the specified *filename* is not valid, **LIST** returns a [200] message. If no items are present in the file, **LIST** returns a [401] message.

See Also: COUNT, LIST.ITEM, LISTF

2.54 LIST.INDEX

The **LIST.INDEX** command lists the indices defined for the specified file.

```
LIST.INDEX filename [index | ALL]
```

See Also: CREATE.INDEX

2.55 LIST.ITEM

The **LIST.ITEM** command lists the items with their attributes in the specified file.

```
LIST.ITEM filename
```

The attributes of an item are presented as numbered lines. After listing each full page of items, **LIST.ITEM** issues a prompt to the user to display the next page. To terminate listing, specify Q at the display prompt.

LIST.ITEM begins each page of its listing with the current date and time.

If the specified *filename* is not valid, **LIST.ITEM** returns a [200] message.

See Also: LIST, LISTF

2.56 LIST.JOB

The **LIST.JOB** command displays a table listing all running phantom processes.

```
LIST.JOB
```

LIST.JOB lists all running processes; **JOBS** lists all running processes initiated by the current process.

See Also: JOBS

2.57 LIST.LOCKS

The **LIST.LOCKS** command displays a table listing the current locks. It lists both system locks and locks established by the current process.

```
LIST.LOCKS
```

For all locks, **LIST.LOCKS** lists the process ID of the process holding the lock and the lock type (X (exclusive) or S (shared)). Locks established using the MVBASIC **LOCK** command are displayed as **LOCK *nnn***. Locks established by opening a sequential file are displayed as **FILE *filename***. Other locks display the lock global variable and its complete pathname.

See Also: **CLEAR.LOCKS**

2.58 LISTF

The **LISTF** command displays a table listing the current MultiValue files.

```
LISTF
```

For each file, **LISTF** lists the file name, the file type (F or Q), the corresponding data file global, and the corresponding dictionary file global. **LISTF** begins its listing with the current date and time, and ends its listing with the total number of files listed.

You can use **SET.FILE** to create a Q-type file.

See Also: **LIST**

2.59 LISTME

The **LISTME** command displays a table listing the current MultiValue user processes.

```
LISTME
```

For each user process, **LISTME** lists the process ID (pid), the port number, the date and time of initialization of the MV Shell, and the user name. The current process is indicated by an asterisk preceding the pid number.

This listing is initiated by displaying a header, and concludes with a count of the number of items listed. After listing each full page of items, **LISTME** issues a prompt to the user to display the next page. To terminate listing, specify **Q** at the display prompt.

The **LISTME** command returns the same information as the **LISTU** and **STATUS** commands. However, **LISTME** can only list processes when invoked from the **USER** account. **LISTU** and **STATUS** list all active processes when invoked from any account.

See Also: **LISTU**, **LOGOFF**, **STATUS**, **WHERE**

2.60 LISTPA

The **LISTPA** command displays a table listing the paragraphs in the **VOC** file.

```
LISTPA
```

For each paragraph, **LISTPA** lists the **VOC**, the **F1**, and the **F2**. **F1** is always "PA". If no paragraphs exist, it returns a [401] message.

See Also: **LISTPH**, **LISTS**

2.61 LISTPEQS

The **LISTPEQS** command displays a table listing the printer queue elements and their status.

```
LISTPEQS [ "account" ] [ jobno[-jobno] ] [ (ACF) ]
```

The optional *account* argument allows you limit display to only the elements assigned to a specific account. *account* is case-sensitive, and must be specified as a quoted string. The optional *jobno* argument allows you to specify a single print job number, or a range of print jobs.

The optional (ACF) code consists of one or more letter codes in any order, enclosed in parentheses. The letter codes have the following meaning: **A**=display only jobs created by the current account. **C**=do not display detailed information, just return the total number of queue elements and pages in use. **F**=sort the display by form queue number.

See Also: **LISTPTR**, **SPOOLER**, **SP.DELETE**, **SP.EDIT**

2.62 LISTPH

The **LISTPH** command displays a table listing the phrases in the VOC file.

```
LISTPA
```

For each paragraph, **LISTPH** lists the VOC, the F1, and the F2. F1 is always “PH”.

See Also: LISTPA, LISTS

2.63 LISTPTR

The **LISTPTR** command displays a table listing the current printer assignments.

```
LISTPTR [ jobno[-jobno] ]
```

The optional *jobno* argument allows you to specify a single print job number, or a range of print jobs.

See Also: LISTPEQS

2.64 LISTS

The **LISTS** command displays a table listing the sentences in the VOC file.

```
LISTS
```

For each paragraph, **LISTS** lists the VOC, the F1, and the F2. F1 is always “S”.

See Also: LISTPA, LISTPH

2.65 LISTU

The **LISTU** command displays a table listing the current MultiValue user processes.

```
LISTU
```

For each user process, **LISTU** lists the process ID (pid), the port number, the date and time of initialization of the MV Shell, and the user name. The current process is indicated by an asterisk preceding the pid number.

This listing is initiated by displaying a header, and concludes with a count of the number of items listed. After listing each full page of items, **LISTU** issues a prompt to the user to display the next page. To terminate listing, specify Q at the display prompt.

The **LISTU** and **STATUS** commands are synonyms.

See Also: LISTME, LOGOFF, STATUS, WHERE

2.66 LOGOFF

The **LOGOFF** command logs off a current MultiValue user process. You can log off your own current process, or another active process.

```
LOGOFF portno
```

Specifying **LOGOFF** with a port number logs off the MV Shell on the specified user terminal process. When you issue a log off, the terminal process immediately exits the MV Shell and returns to the Caché mode prompt. If you specify an invalid *portno*, the **LOGOFF** command completes successfully, performing no operation.

To determine the *portno* of current user processes, use the **WHO** command, or the **@PORTNO** MVBasic system variable.

The **LOGOFF** and **LOGOUT** commands are similar.

See Also: LOGOUT, Q, QUIT, WHO

2.67 LOGOUT

The **LOGOUT** command logs off a current MultiValue user process. You can log off your own current process, or another active process.

```
LOGOUT [pid]
```

Specifying **LOGOUT** with no operand logs off the MV Shell on the current terminal process. (You can use the **Q** or **QUIT** command to perform the same operation.) Specifying **LOGOUT pid** logs off the MV Shell on the specified user terminal process. Specifying **LOGOUT ALL** logs off the MV Shell on all user terminal processes except the current terminal process.

When you issue a log off, the terminal process immediately exits the MV Shell and returns to the Caché mode prompt. If you specify an invalid *pid*, the **LOGOUT** command completes successfully, performing no operation.

To determine the *pid* of current user processes, use the **LISTME** command.

The **LOGOUT** and **LOGOFF** commands are similar.

See Also: LISTME, LOGOFF, Q, QUIT

2.68 LOGTO

The **LOGTO** command changes your MV shell login to another account (namespace). This changes the command line prompt.

```
LOGTO account
```

Note that namespace names are not case sensitive, but account names are case sensitive, and are specified in all uppercase letters. The SYSPROG account corresponds to the %SYS namespace. If the specified *account* is not valid, **LOGTO** returns a [229] message. If a **&HOLD&** file has been created in the old account, **LOGTO** create a **&HOLD&** file in the new account.

See Also: CREATE.ACCOUNT

2.69 MVI

The **MVI** command locates the source code for a MVBasic routine.

```
MVI routine [linenumber]
```

MVI searches for MVB.routine. If the specified *routine* is not located, you may be searching in the wrong account.

See Also: BASIC

2.70 MVIMPORT

The **MVIMPORT** command imports accounts from other MultiValue implementations to Caché MultiValue. This command is described in the **MVIMPORT** chapter of this manual.

2.71 NSELECT

The **NSELECT** command generates a select list of items in the supplied (or default) select list that are *not* in the file.

```
NSELECT [DICT] filename [FROM n][TO n]
```

You can use the FROM clause to specify an existing numeric select list. By combining the FROM and TO clauses you can specify a range of numbered select lists. By default, **NSELECT** uses select list 0.

If the specified *filename* is not valid, **NSELECT** returns a [201] message. If there is no active select list, **NSELECT** returns a [240] message.

See Also: SELECT

2.72 P

The **P** command suppresses terminal screen display. It suppresses all terminal display, including displaying the terminal prompt.

```
P [ON | OFF]
```

Specifying **P** with no operand toggles display suppression. **P ON** suppresses display. **P OFF** re-enables display.

The **KEYS** command temporarily overrides the **P**. However, when **KEYS** times out, it returns to the prior **P** mode. This may be mistaken for a hang state.

The **P** and **HUSH** commands are synonyms.

See Also: DISPLAY, HUSH

2.73 PHANTOM

The **PHANTOM** command starts a background process in which to run the specified command.

```
PHANTOM [BRIEF | SQUAWK] command
```

PHANTOM returns the process ID (pid) assigned to the background process. By default, output records are created in the &PH& file. Caché stores the &PH& file using the ^PH global. In PICK-style emulation (jBASE for example), output records are created in the Spooler file.

In CACHE emulation, the optional SQUAWK keyword causes **PHANTOM** to also display the record number of the record created in the &PH& file. In jBASE emulation, SQUAWK returns the spooler job number.

The optional BRIEF keyword starts a background process and returns the process ID (pid), but causes no output to be generated.

PHANTOM is similar to the **ZH** and **Z** commands, except that those commands always output to the spooler, and **ZH** and **Z** can be specified with no argument, causing them to prompt for the account, password, and command to execute.

See Also: JOBS, LIST.JOB, Z, ZH

2.74 PQ.SELECT

The **PQ.SELECT** command copies the default select list (select list 0) into the specified numbered select list

```
PQ.SELECT n
```

The *n* argument is a select list number in the range 1 through 10 (inclusive). If *n* is out of range, **PQ.SELECT** returns a [819] message. If the default select list is not active, **PQ.SELECT** returns a [240] message.

2.75 PRINT.CATALOG

The **PRINT.CATALOG** command displays a list of catalog pointers and their program references.

```
PRINT.CATALOG filename
```

The *filename* argument can be an MVBasic source file, or the VOC.

If an account or routine is missing, **PRINT.CATALOG** displays an appropriate message.

2.76 PTERM

The **PTERM** command sets and displays terminal options.

```
PTERM [LPTR channel] [DEVICE name] [DISPLAY] [option value]
```

By default, the terminal is the user terminal. The *option* argument is the name of an option; the *value* argument is a keyword setting for that option. Available *option value* pairs are CASE INVERT and CASE NOINVERT, CRMODE INLCR and CRMODE NOINLCR.

2.77 Q

The **Q** command quits the MultiValue shell.

```
Q
```

Specifying **Q** causes the current terminal process to immediately exit the MV Shell and returns to the Caché mode prompt. To perform the same operation on other terminal processes, use the **LOGOFF** or **LOGOUT** commands.

The **Q** and **QUIT** commands are synonyms.

See Also: LOGOFF, LOGOUT, QUIT

2.78 QSELECT

The **QSELECT** command generates a select list of the specified items.

```
QSELECT filename [itemlist] [option]
```

The *option* argument allows you to specify the numeric code for an attribute.

By default, **QSELECT** uses select list 0.

If the specified *filename* is not valid, **QSELECT** returns a [201] message.

See Also: SELECT

2.79 QUIT

The **QUIT** command quits the MultiValue shell.

```
QUIT
```

Specifying **QUIT** causes the current terminal process to immediately exit the MV Shell and returns to the Caché mode prompt. To perform the same operation on other terminal processes, use the **LOGOFF** or **LOGOUT** commands.

The **QUIT** and **Q** commands are synonyms.

See Also: LOGOFF, LOGOUT, Q

2.80 RUN

The **RUN** command runs an MVBasic program.

```
RUN filename[,datasection] program [options]
```

Specify *datasection* if the program to be run is stored in the data section of the file.

Before you can run an MVBasic program, you must first compile it using the **BASIC** command.

See Also: ; (semicolon), BASIC, MVI

2.81 SAVE.LIST

The **SAVE.LIST** command copies the specified numbered select list into a named select list.

```
SAVE.LIST [listname [FROM n]]
```

Use the optional *listname* argument to specify a named select list; if omitted, the select list name is taken from the process ID (pid) or the UDT_SAVEDLIST environment variable. The optional FROM *n* argument is a select list number in the range 0 through 10 (inclusive); if omitted, select list 0 is the default.

If *n* is out of range, **SAVE.LIST** returns a [209] message.

SAVE.LIST is the inverse of **GET.LIST**. To copy Select List 0 to another numbered select list, use the **PQ.SELECT** command.

See Also: GET.LIST, PQ.SELECT

2.82 SEARCH

The **SEARCH** command searches an item for one or more strings.

```
SEARCH filename [item]
```

SEARCH prompts you for strings to search for. If you omit the *item* argument, it prompts you for that as well.

If the specified *filename* is not valid, **SEARCH** returns a [201] message. If the specified *item* is not valid, **SEARCH** returns a [202] message.

2.83 SELECT

The **SELECT** command generates a select list of items in the file that meet the specified criteria.

```
SELECT filename [WITH field = value] [TO listnum]
```

You can use the TO clause to specify a numeric select list. By default, **SELECT** uses select list 0.

If the specified *filename* is not valid, **SELECT** returns a [200] message.

See Also: NSELECT

2.84 SET.FILE

The **SET.FILE** command creates a type Q file.

```
SET.FILE [datafile] [dictfile]
```

SET.FILE with no arguments creates a type Q file with no data file name and VOC as the dictionary file. **SET.FILE datafile** creates a type Q file with the specified data file name and VOC as the dictionary file. **SET.FILE datafile dictfile** creates a type Q file with the specified data file name and dictionary file name. There can only be one type Q file at a time; issuing **SET.FILE** replaces the existing Q file.

You can use **LISTF** to list files with their file type (F or Q), the corresponding data file and dictionary file.

See Also: LISTF

2.85 SETPTR

The **SETPTR** command lists and sets the current printer settings.

```
SETPTR channel,width,lines,top,bottom,mode[,BANNER value | KEEP |  
BRIEF]
```

SETPTR with no arguments lists the current printer settings. To change one or more of these settings, specify the positional parameters with the appropriate leading commas. *channel* is the unit number, the default is 0. *width* is the page width, the default is 132. *lines* is the page depth (number of lines per page), the default is 66. Setting *lines* to 0 disables pagination. *top* is the top of page margin, the default is 3. *bottom* is the bottom of page margin, the default is 3. *mode* is the print mode, the default is 1

(Spooled output). Available *mode* values are 1 (output to the spooler) and 3 (output to the &HOLD& file).

Defaulting of *width*, *lines*, *top*, or *bottom*, is emulation-dependent. For example, **SETPTR ,,,,3** reverts these four settings to the defaults in some MultiValue emulations. In other MultiValue emulations, it retains the previously set values.

If the &HOLD& file does not exist, **SETPTR** with *mode*=3 creates this file in the current mgr/namespace directory. However, you can use **CREATE-FILE &HOLD& ANODE** to create &HOLD& as a MultiValue global. In this case, a subsequent **SETPTR** with *mode*=3 writes to this existing &HOLD& ANODE global file. You must specify ANODE; by default **CREATE-FILE** creates an INODE file. An INODE file cannot be used by **SETPTR**.

BRIEF means that changes to the current printer settings will be made without displaying the changes and prompting you to confirm them. Thus, **SETPTR , , 64** prompts you for confirmation before it sets the page width to 64 lines; **SETPTR , , 64 , , , , BRIEF** sets the page width to 64 lines without prompting for confirmation.

KEEP means that the job will be kept open and not closed when the program terminates. You can use **SP.CLOSE** to close the job. The job will be closed when you exit the MultiValue Shell.

If *mode*=1, **BANNER** means there will be a one page banner printed when the job is despoiled to a printer.

If *mode*=3, **BANNER** has a different meaning, it sets the item ID in &HOLD&: **BANNER name**

- **BANNER name**: item ID is always *name*. Each subsequent job overwrites the previous version of *name*.
- **BANNER NEXT**: item ID is an incremented number, P#0000_#####, where ##### is incremented for each entry to &HOLD&, ##### increments from 0001 through 9999.
- **BANNER NEXT name**: item ID is an incremented number, name_#####, where ##### is incremented for each entry to &HOLD&. ##### increments from 0001 through 9999.
- **BANNER UNIQUE**: item ID is an incremented number, P#0000_#####, where ##### is incremented each time the **SETPTR** command is executed. ##### increments from 0001 through 9999.

After specifying these settings, **SETPTR** prompts you to confirm them with a Y or N, unless you specified the **BRIEF** option.

You can also use **TERM** to change the *width* and *lines* printer settings.

For further details, refer to the Spooler Commands section of the “Spooler Administration” chapter of [The Caché MultiValue Spooler](#).

See Also: **SP.CLOSE**, **TERM**

2.86 SH

The **SH** command issues an operating system command.

```
SH [commandline]
```

SH issues a operating system command without exiting the MultiValue Shell. **SH** with no argument opens an interactive command prompt from which you can issue operating system commands. **SH commandline** issues an operating system command as a background process. The *commandline* can be any valid command line for the current operating system.

You can use **COS** to issue a Caché ObjectScript command without exiting the MultiValue Shell.

See Also: COS

2.87 SLEEP

The **SLEEP** command suspends the process for the specified number of seconds. After the elapsed number of seconds it returns to the MultiValue Shell prompt.

```
SLEEP seconds
```

You can specify *seconds* as an integer or a fraction.

2.88 SORT.LIST

The **SORT.LIST** command sorts a saved select list in the &SAVEDLISTS& file.

```
SORT.LIST [filename [listname] ]
```

The optional *filename* is the destination file where the sorted select list is to be stored. If you omit *filename*, **SORT.LIST** prompts you for the list ID. The optional *listname* is the name of an existing select list in &SAVEDLISTS&; the default is select list 0. Caché stores the &SAVEDLISTS& file using the ^SAVEDLISTS global.

See Also: COPY.LIST, EDIT.LIST

2.89 SP.x Commands

Caché MultiValue supports 34 commands that control the Spooler. The names of these commands begin with either “SP-” or “SP.” Caché supports both variant forms: the hyphen form and the dot form. For example, **SP-ASSIGN** and **SP.ASSIGN** are different names for the same command. In most cases, these two forms are synonymous. The one exception is **SP-EDIT** and **SP.EDIT**, which provide different syntax options.

The following spooler commands are supported: [SP.ASSIGN](#), [SP.AUX](#), [SP.CLEAR](#), [SP.CLOSE](#), [SP.COPY](#), [SP.COPIES](#), [SP.CONDUCT](#), [SP.CREATE](#), [SP.DELETE](#), [SP.DEVICE](#), [SP-EDIT](#), [SP.EDIT](#), [SP.EJECT](#), [SP.FORM](#), [SP.FQDELETE](#), [SP.GLOBAL](#), [SP.JOBS](#), [SP.KILL](#), [SP.LOOK](#), [SP.MOVEQ](#), [SP.NEWTab](#), [SP.OPEN](#), [SP.OPTS](#), [SP.PAGESIZE](#), [SP.POSTAMBLE](#), [SP.PREAMBLE](#), [SP.PURGEQ](#), [SP.RESUME](#), [SP.SKIP](#), [SP.START](#), [SP.STATUS](#), [SP.STOP](#), [SP.SUSPEND](#), [SP.SWITCH](#), [SP.TESTPAGE](#), [SP.VERBOSE](#).

For further information on these commands, refer to the “[Spooler Commands](#)” chapter of *The Caché MultiValue Spooler*.

2.90 SPOOL

The **SPOOL** command controls the spooling of files for printing. It has three forms: send a file to the spooler for printing; list the files pending on the spooler queue; delete a print job from the spooler queue.

```
SPOOL filename itemID SPOOL -LIST [formname]
SPOOL -CANCEL joblist
```

`SPOOL filename itemID` takes a MultiValue item and prints it to the currently assigned printer.

`SPOOL -LIST` lists all the jobs on the spooler table. The optional *formname* argument allows you to filter to a single form queue name. The form queue can be specified either by name or by number. The default form queue has the name `STANDARD`, and a form queue number of 0. It can be specified as “`STANDARD`”, “`0`”, “`F0`”, “`FN0`”, or “`FQ0`”.

`SPOOL -CANCEL` deletes one or more pending print jobs. The *joblist* argument allows you to specify any number of individual print job numbers (separated by blank spaces), as shown in the following example: `SPOOL -CANCEL 66 68 71`. You can also delete a range of print jobs, as shown in the following example: `SPOOL -CANCEL 66-70`.

For further details and examples, refer to [The Caché MV Spooler manual](#).

See Also: `SP.DELETE`

2.91 STACKCOMMON

The **STACKCOMMON** command specifies whether the MVBasic **PERFORM** statement stacks unnamed **COMMON** variable areas.

```
STACKCOMMON  
STACKCOMMON ON  
STACKCOMMON OFF
```

STACKCOMMON with no argument returns the current setting. **STACKCOMMON ON** causes each **PERFORM** to **NEW** the unnamed **COMMON** variables area before calling a routine. **STACKCOMMON OFF** (the default) causes the unnamed **COMMON** variables area to be preserved across multiple **PERFORM** routine calls.

See Also: The MVBasic [COMMON](#) and [PERFORM](#) statements.

2.92 STATUS

The **STATUS** command displays a table listing the current MultiValue user processes.

```
STATUS
```

For each user process, **STATUS** lists the process ID (pid), the port number, the date and time of initialization of the MV Shell, and the user name. The current process is indicated by an asterisk preceding the pid number. If a current terminal process is not running the MultiValue Shell, **STATUS** does not display it.

This listing is initiated by displaying a header, and concludes with a count of the number of items listed. After listing each full page of items, **STATUS** issues a prompt to the user to display the next page. To terminate listing, specify **Q** at the display prompt.

The **STATUS** and **LISTU** commands are synonyms. The **LISTME** command returns identical information, but only lists processes when in the **USER** account. To list the current account name and currently executing command for each terminal process, use the **WHERE** command.

See Also: **LISTME**, **LISTU**, **LOGOFF**, **WHERE**

2.93 TABS

The **TABS** command sets tab stops.

```
TABS n[ ,n[ ,n]]
```

TABS can be used to set any number of tab stops at specified character positions. Multiple *n* arguments can be separated by commas or spaces. **TABS** overwrites any previous tabs settings. To remove all tabs, specify **TABS** with no arguments. If no tabs are set, the tab key advances by a single space.

2.94 TANDEM

The **TANDEM** command allows one MultiValue user to connect to the terminal of another MultiValue user, sharing terminal input and output.

```
TANDEM [ON | OFF] [(N) | (F)]
TANDEM port
```

A tandem session consists of a master and a slave terminal. The slave terminal uses the ON or OFF keyword to specify availability to receive a tandem slave request. (The (N) and (F) keywords are synonyms for ON and OFF.) Once the slave terminal has specified **TANDEM ON** (or **TANDEM (N)**), the master terminal can use the *port* argument to specify the terminal they wish to establish as a tandem slave. To issue **TANDEM port**, the master terminal must be in the %SYS namespace (the SYSPROG account). The tandem session then initiates when the slave terminal next presses the ENTER key.

The master terminal initially enters the tandem session in view-only mode. To quit the session in view-only mode, type “Q”. To change the mode, type one of the following: **Esc-F** puts the master in feed mode; **Esc-V** puts the master in view-only mode; **Esc-M** puts both master and slave in message mode; **Esc-X** causes the master to exit the tandem session.

For further details, refer to the **TANDEM** section of the *Operational Differences between MultiValue and Caché* manual.

2.95 TERM

The **TERM** command displays and sets terminal and printer display characteristics.

```
TERM name  
TERM [#1, #2, #3, #4, #5, #6, #7, #8, #9, #10]
```

TERM name sets the terminal type to *name*; letter case is preserved. You can display the current terminal type using the MVBASIC [SYSTEM \(7\)](#) function. (**TERM name** and **TERM ,,.,.,.,name** are synonymous.)

TERM with no argument returns the current terminal and printer settings.

TERM with one or more numeric arguments sets terminal and printer display characteristics using positional argument values. You may specify any subset of the ten possible settings by including leading commas for unspecified values. Trailing commas are not required. For example, **TERM , , , , , , 100** sets the seventh argument: Printer page width. The unspecified arguments retain their previous values.

The ten positional arguments are as follows:

1	Terminal page width	Maximum line length displayed on the terminal screen. Available values range from 11 to 32767; the default is 79.
2	Terminal page depth	Maximum number of lines displayed on the terminal screen. Available values range from 1 to 32767; the default is 24.
3	Terminal line skip	Number of lines skipped at the bottom of the terminal screen. The default is 0.
4	Line feed delay	Number of seconds to delay following a line feed. Available values range from 0 to 7. Default is no delay.
5	Form feed delay	Number of seconds to delay following a form feed. Available values range from 0 to 7. Default is no delay.
6	Alternate backspace	ASCII decimal code value corresponding to the backspace character. The default is 8.
7	Printer page width	Maximum number of characters per line for printer output. Available values range from 1 to 32767; the default is 132.
8	Printer page depth	Maximum number lines per page for printer output. Available values range from 1 to 32767; the default is 66.
9	Terminal type	The terminal type. The default is CACHE.
10	Printer line skip	Number of lines skipped at the bottom of the printer page. The default is 0.

If you specify an out-of-range value, **TERM** returns a [290] message. If you specify an illegal value, **TERM** returns a [202] message.

For further details, refer to the [Terminal Output](#) chapter of the *Caché MV Terminal Independence* manual.

See Also: CHOOSE.TERM, COMPILE.TERM, SETPTR

2.96 TERM.TYPE

The **TERM.TYPE** and **TERM** commands are synonyms.

2.97 TIME

The **TIME** command returns the current time and date, or returns the elapsed execution time for a MultiValue command.

```
TIME [command]
```

TIME with no argument returns the current time and date. For example: “13:23:26 24 MAR 2008”. This date and time format is *not* affected by the **DATE.FORMAT** command. You can use the **DATE** command to return the current date and time.

TIME command returns the elapsed execution time (in fractional seconds) for the specified MultiValue command.

For further details, refer to the [TIME](#) section of the *Operational Differences between MultiValue and Caché* manual.

See Also: [DATE](#)

2.98 TRACE

The **TRACE** command establishes a trace mode that times the execution of each subsequent command.

```
TRACE [ON | OFF]
```

When **TRACE** is on, the invocation of a MultiValue command or a MVBasic statement returns the **START** date/time (in internal format) before execution and the **END** date/time upon completion, along with the elapsed execution time in fractional seconds.

To activate this debug behavior for subsequent commands, invoke **TRACE ON**. To inactivate this behavior, invoke **TRACE OFF**. **TRACE** with no argument returns the current trace status (on or off).

You can use **TIME command** to return the elapsed execution time (in fractional seconds) for a single specified MultiValue command.

See Also: [TIME](#)

2.99 UNASSIGN

The **UNASSIGN** command deletes the assignment of a form queue spool device to a LPTR device.

```
UNASSIGN form-queue
```

The *form-queue* can be specified either by name or by number. The default form queue has the name STANDARD, and a form queue number of 0. It can be specified as “STANDARD”, “0”, “F0”, “FN0”, or “FQ0”.

See Also: ASSIGN

2.100 WHERE

The **WHERE** command displays a table listing the current MultiValue user processes, with their current account locations, and currently executing command.

```
WHERE [startport [-endport]] [(V)]
```

By default, **WHERE** lists all running MultiValue processes, including phantom processes. By using the optional *startport* and *endport* arguments you can restrict the list of processes returned to a single port number or a specified range of port numbers. The optional (V) argument specifies verbose mode, supplying more information about each process.

For each terminal process running the MultiValue Shell, **WHERE** lists the port number, the device ID (including the pid), the current account name and current namespace name (these are often the same), and the currently executing MultiValue command. The SYSPROG account corresponds to the %SYS namespace.

The active process is indicated by an asterisk preceding the port number. Its current command is always “WHERE”. Other terminal processes indicate the currently executing command (for example, **KEYS** or **SLEEP**); if no MultiValue command is executing, they return “MVShell” as the current command. If a current terminal process is not running the MultiValue Shell, **WHERE** does not displayed it.

See Also: STATUS

2.101 WHO

The **WHO** command displays the terminal port number, the account name, and the user name.

```
WHO
```

You can use **STATUS** to find the process ID (pid) corresponding to the terminal port number. The account name usually identical to the Caché namespace name. However, note that namespace names are not case sensitive, but account names are case sensitive. The **SYSPROG** account corresponds to the **%SYS** namespace. An account is created using the **CREATE.ACCOUNT** command.

2.102 Z

The **Z** command starts a background (phantom) process in which to execute the specified MultiValue *command*. **Z** then returns to the MultiValue Shell with a message specifying the process ID (pid) used for the background process.

```
Z [command]
```

If you specify **Z** with no argument, it prompts you for account name, password, and command to execute. The account name is usually the same as the Caché namespace name. The password is ignored. The command is any valid MultiValue command.

If the MultiValue emulation is **PICK**, the *command* output is directed to the spooler process. If the MultiValue emulation is **UniVerse** (or **Cache**), the *command* output is stored as a record in the **&PH&** file. Caché stores the **&PH&** file using the **^PH** global. You can use the **CEMU** command to change emulation.

The **PHANTOM** and **ZH** commands perform the same operation. **PHANTOM** does not prompt for an omitted argument. **ZH** returns a message specifying where the *command* output has been directed.

See Also: **PHANTOM**, **ZH**

2.103 ZH

The **ZH** command starts a background (phantom) process in which to execute the specified MultiValue *command*. **ZH** then returns to the MultiValue Shell with a message specifying the process ID (pid) used for the background process.

```
ZH [command]
```

If you specify **ZH** with no argument, it prompts you for account name, password, and command to execute. The account name is usually the same as the Caché namespace name. The password is ignored. The command is any valid MultiValue command.

If the MultiValue emulation is PICK, the *command* output is directed to the spooler process. If the MultiValue emulation is UniVerse (or Cache), the *command* output is stored as a record in the &PH& file. Caché stores the &PH& file using the ^PH global. You can use the **CEMU** command to change emulation.

The **PHANTOM** and **Z** commands perform the same operation. **PHANTOM** does not prompt for an omitted argument. **Z** does not return a message specifying where the *command* output has been directed.

See Also: PHANTOM, Z

