



CBLi 1.20 New Features

8 Merthyr Mawr Road, Bridgend, Wales UK CF31 3NH

Tel: +44 (1656) 65 2222
Fax: +44 (1656) 65 2227

CBL Web Site - <http://www.cbl.com>

This document may be downloaded from **<http://www.cbl.com/cblidoc.html>**

Contents

CBLi 1.20 New Features	1
Documentation Notes.....	1
Section 01: 3270 Emulators	2
Section 02: Standard 3270 (Model 2/3/4/5) terminals	3
Section 03: Disaster Recovery - CBLi VTAM Execution	4
Section 04: CBLi CLI Commands	5
EDIT fileid (options).....	5
ERASE Command.....	5
RENAME Command.....	5
Window Resizing.....	6
Section 05: CBLi Text Editor	7
VSAM Data Sets.....	7
Temporarily Ignore Line End Character.....	7
Section 06: CBLi CLI Commands	8
ALLOCATE Command.....	8
COMMAND Command.....	9
CREATE BUTTON Command.....	9
DESTROY BUTTON Command.....	11
EDIT Command Initialisation options.....	11
EDITV Command.....	12
EXTRACT Command Options Added.....	13
FREE Command.....	16
LESS & MORE Commands Parameter TAG Added.....	17
LIST Command.....	18
OVERLAY Command.....	20
QUERY Command Options Added.....	20
REFRESH Command.....	21
REPLACE Command.....	22
SHIFT Command.....	22
SOS Command Parameter REFRESH.....	23
SYNEX Command.....	23
WINDOW Command Parameters Added.....	24
Section 07: CBLi SET Commands	26
SET COLOUR Command Parameters Added.....	26
SET DEFPROFILE Command.....	26
SET DSORG Command.....	27
SET HSCROLLCURSOR Command.....	27
SET ISPFMODE Command.....	28
SET KEY Command.....	28
SET LCOLOUR Command.....	29
SET LINEFLAG Command.....	30
SET MDILIST Command.....	31
SET RANGE Command.....	31
SET SCOLOUR Command.....	32
SET SYNONYM Command.....	33
SET THIGHLIGHT Command.....	34
SET WINNAME Command.....	34
SET WINPOS Command.....	35
SET WINSIZE Command.....	35
Section 08: CBLi Edit Macros	37
BOX Macro.....	37
BOXSEQ Macro.....	37
CMXAMS Macro.....	38
DELALL Macro.....	38
DELNOT Macro.....	39
GETAVRL Macro.....	39
IALL Macro.....	40
ISPFMAC Macro.....	40
JCLCMX Macro.....	41
LDIFF Macro.....	42
LM Macro.....	42

Contents

<u>Section 08: CBLe Edit Macros</u>	
<u>NAM Macro</u>	43
<u>PROFIRST Macro</u>	43
<u>QX Macro</u>	43
<u>SDBPOPOP Macro</u>	44
<u>SDBTRACK Macro</u>	45
<u>SDBWINX Macro</u>	45
<u>TRB Macro</u>	46
<u>TSOC Macro</u>	47
<u>WINX Macro</u>	47
<u>Section 09: CBL CMX (Command) Files</u>	49
<u>CBLIDEMO</u>	49
<u>CBLIINST</u>	49
<u>Section 10: SELCOPY Interactive</u>	50
<u>CBLe text edit support</u>	50
<u>MDI (Multiple Document Interface) Support</u>	50
<u>Popup Menu and Window Configuration</u>	50
<u>TRACE window</u>	50
<u>Messages</u>	50
<u>Section 11: CBLiINI Options</u>	51
<u>Section 12: Define VSAM Dialogs</u>	52
<u>Section 13: Operating System Window</u>	53
<u>Section 14: List and CBLVCAT window Prefix Commands</u>	54
<u>Glossary</u>	55

CBLi 1.20 New Features

Documentation Notes

Information in this New Feature List reflects differences between CBLi 1.10 and CBLi 1.20.

The **CBLi Reference and User Guide**, **CBLi Editor Manual**, **CBLi Installation Guide** and **CBLi New Features** are available in Adobe Acrobat PDF format at CBL web page <http://www.cbl.com/cblidoc.html>.

Copyright in the whole and every part of this document and of the CBLi and CBLi system and programs, is owned by Compute (Bridgend) Ltd, whose registered office is located at 8 Merthyr Mawr Road, Bridgend, Wales, UK, CF31 3NH, and who reserve the right to alter, at their convenience, the whole or any part of this document or the CBLi and CBLi system and programs.

No reproduction of the whole or any part of the CBLi or CBLi system and programs, or of this document, is to be made without prior written authority from Compute (Bridgend) Ltd.

At the time of publication, this document is believed to be correct. CBL do not warrant that upward compatibility will be maintained for any use made of this program product to perform any operation in a manner not documented within the user manual.

Section 01: 3270 Emulators

A selection of currently available 3270 emulator packages have been installed at CBL to determine whether support is included for those additional features considered useful for CBLi execution (e.g. Keyboard macros.) The following link opens a page on the CBL web site that contains the results for each emulator tested to date.

<http://www.cbl.com/cbli3270.html>

Emulator configuration files and macros that are applicable to individual 3270 emulator packages, have been generated and published on the CBL web pages for use with CBLi. Please see the following:

<http://www.cbl.com/cblidl.html>

Section 02: Standard 3270 (Model 2/3/4/5) terminals

Operation of CBLi when used with these types of 3270 terminals has been improved. In particular, the CBLi editor and SELCOPY Interactive parent windows have been amended to facilitate use on smaller terminals. Problems encountered when running CBLi in smaller 3270 terminal sizes, are primarily related to the quantity of data that can be displayed in a window and navigation between the open windows.

These changes include:

1. MDI support for SELCOPY Interactive

In earlier releases of CBLi, each window associated with SELCOPY Interactive, was treated as being a child window of the CBLi main window so that PF9 (assigned, by default, to CBLi command "NextWindow") would pass focus to windows not associated with the SELCOPY Interactive application.

This problem is overcome by making SELCOPY Interactive an MDI (Multiple Document Interface) application. An MDI application comprises a parent window with a client area within which one or more MDI child windows are displayed. All MDI child windows are confined to the parent window's client area.

Making SELCOPY Interactive an MDI application allows us to assign PF9 to command "MDINext", thus restricting focus to only those child windows that belong to the MDI application.

Note: The CBLi text editor is already an MDI application.

2. MDI List window support

Introduction of the CBLi command "SET MDILIST ON|OFF" allows the user to control how a CBLi List window is displayed when opened from any MDI child window in the current MDI application.

ON List window is opened as an MDI child of the current MDI application.

OFF List window is opened as a child of the CBLi main window. (Default action in CBLi 1.10.)

SELCOPY Interactive now also supports CBLi text edit MDI child windows in the same way as the CBLi text editor itself. This means that any CBLi edit command or macro may be issued from a CBLi child window within the SELCOPY Interactive application. Therefore, MDILIST may be set in the CBLi and/or in the SELCOPY Interactive MDI applications.

Setting MDILIST ON has the benefit of allowing the user to place focus on a list window within the MDI application (using PF9) i.e. without having to leave the application. This is particularly useful when running the MDI application in a maximised state.

MDILIST ON is the default.

3. Maximised MDI windows

MDI parent windows now open in a maximised state if the width of the 3270 terminal is equal to 80 bytes. This will display the maximum amount of data possible.

SELCOPY Interactive contains, as standard, 4 MDI child windows of different formats (CBLi and Storage) that are opened in fixed positions within the client area. Because these fixed positions are arranged so that each window is visible when used with large terminal sizes, the window positions are not optimum when used with smaller terminals. Therefore, SELCOPY Interactive child windows are also opened in a maximised state where the terminal display is of width 80 bytes.

4. MDI Edit View Configuration Macros

CBLi edit macros WINX and SDBWINX allow users to save and restore the configuration (size and position) of edit view child windows in the MDI application.

In CBLi, where the CBL supplied PROFILE macro is used as the default edit profile macro, user buttons **wS** and **wR** are added to the menu bar. These buttons invoke the WINX macro to save and restore the current edit view configuration, respectively.

By default, in SELCOPY Interactive, the PF4 key is assigned to macro SDBPOPUP. This macro opens a popup menu containing items that allow the user to perform SELCOPY Interactive functions on the focus token. It also contains a **Window Layout** item which invokes the SDBWINX macro to save, restore or set back to default the size and position of all edit views in the application.

Section 03: Disaster Recovery - CBLi VTAM Execution

This is a recommendation rather than a new feature since CBLi release 1.10 has always been able to execute as a VTAM application.

CBLi has the same functionality when executing as a VTAM application as it does when running under TSO. Therefore, the CBLIVTAM module and VTAM applid may be installed on a recovery volume to perform data editing, data set allocation and system navigation if ISPF is unavailable.

Couple this with SELCOPY, which is installed in the same load library as CBLi, the systems programmer is provided with a powerful set of tools to aid the data recovery process.

1. fileid1 is the data set and member name of the member to be renamed and fileid2 is the new member name only.
2. fileid1 is the fully quoted data set and member name of the member to be renamed and fileid2 is the fully quoted data set and new member name.

For this second method and all other types of MVS data set, RENAME executes the following IDCAMS command:

```
ALTER fileid1 NEWNAME(fileid2)
```

Therefore, types of file that may be renamed and the supported format of fileid1/2 is governed by the IDCAMS ALTER command. (See "DFSMS Access Method Services for Catalogs".)

Note that RENAME does not utilise the TSO prefix.

Parameters:

fileid1	Current fileid of the file.
fileid2	New fileid to be assigned to the file.

Examples:

```
rename cbl.ssc.ctl(ssstest) sstest01
rename 'cbl.ssc.ctl(ssstest)' 'cbl.ssc.ctl(ssstest2) '
ren cbl.cbli.test.file nbj.test.data
ren oem.cbl.cbliuser.ini oem.cbl.nbj.ini
```

Window Resizing

New window resizing commands intended to be assigned to PFKeys at the "Border" level (see CLI command KEYS).

```
DRAGBORDERMINUS
```

Depending on which window border the cursor is positioned when the PFKey is hit i.e. horizontal, vertical or corner, the DRAGBORDERMINUS command positions the window's border one line, column or both closer to the top left corner of the 3270 display area.

By default, DRAGBORDERMINUS is assigned to Border PFKeys F7 and F10.

```
DRAGBORDERPLUS
```

Same as DRAGBORDERMINUS except that the window's border is positioned one line, column or both away from the top left corner of the 3270 display area.

By default, DRAGBORDERPLUS is assigned to Border PFKeys F8 and F11.

Section 05: CBLe Text Editor

VSAM Data Sets

Previous releases of CBLi support READ ONLY edit of VSAM data sets using the CBLe text editor.

Support has been introduced to allow READ/WRITE edit of VSAM data sets that have a high-used RBA of zero or are defined with the **REUSE** parameter.

1. Data sets defined with parameter REUSE.
When a save is actioned (SAVE, FILE, etc.), CBLe re-opens the data set with the RESET attribute so that the high-used RBA is reset to zero.
2. Empty Data Sets.
Where no data has ever been written to a data set, the high-used RBA is zero. Beware that, having saved data to an empty data set that is defined with NOREUSE, the high-used RBA is no longer set to zero and subsequent attempts to save the file will fail with the following error message:

```
EDT085E VSAM PUT error for file dataset : return code x'8' reason code x'8'.
```

When editing KSDS data sets, care must be taken to preserve the primary key sequence. If an attempt is made to save a KSDS data set that is not in key sequence, then one of the following error messages is returned:

```
EDT086E Duplicate keys - records m and n have the same key.  
EDT087E Sequence error - record m has a higher key than record n.
```

Temporarily Ignore Line End Character

CBLe supports a line end special character which allows the user to issue multiple commands from either the command line or via the CMDTEXT interface, simply by separating each command with the line end character.

The line end character definition is controlled using the SET LINEND command and switched on and set to "!" (exclamation mark) by default.

If LINEND is ON and the user wants to temporarily switch it off in order to execute a command that contains the linend character as text, then this may be achieved by prefixing the command with the line end character.

Examples:

```
!c/Hello there!/Goodbye/ 10 1  
Change command.
```

```
!/==!/   
Locate line-target.
```

```
! nomsg all "!!READ THIS!!"   
ALL command.
```

Section 06: CBLi CLI Commands

ALLOCATE Command

Syntax:

```
>>-- ALLOCate --+-----+---- allocparms ----><
                |         |
                +- -Cat ---+
                |         |
                +- -FREE +-
```

Description:

The ALLOCate command may be used to:

1. Dynamically define and/or allocate a data set.
2. Concatenate a list of data sets.
3. Concatenate a data set to an existing list of data sets.
4. Free (unallocate) a data set or override DISP/CLASS. (Same as FREE command.)

ALLOCATE allows users to allocate files whether or not a TSO environment is available. The syntax of the command closely matches that of the TSO ALLOCATE command so most ALLOCATE commands, executed without TSO as a prefix, will give the same results.

ALLOCATE is supported for MVS only.

Parameters:

- CAT The data set being allocated is concatenated to an existing data set, or list of data sets, allocated to the specified ddname.
- FREE Unallocates the data set(s) allocated to the specified ddname.
- allocparms Parameters supported by the TSO ALLOCATE command as follow:

```
DDname(ddname) or File(ddname), DATaset('dsn'..) or DSName('dsn'..) or DUMMY,
MOD, NEW, OLD, SHR, CATALOG, DELETE, KEEP, UNCATALOG, TRACKS, CYLinders,
BLOCK, DIR(directory-blocks), SPACE(n[,m]), VOL(volser[,volser...]),
MAXVOL(volumes), UNIT(unit), SYSOUT[(class)], WRITER(external-writer-name),
FORMS(forms), DEST(dest or node[.user]), COPIES(copies), BLKSIZE(blocksize),
LRECL(record-length), DSORG(PS|PO|DA), RECFM(format[,format...]),
BUFNO(buffers), OUTDES(output-descriptor-name), STORCLAS(storage-class),
MGMTCLAS(management-class), DATACLAS(data-class), RECOG(LS),
DSNTYPE(LIBRARY|PDS|HFS), SPIN(UNALLOC) PATH(pathname),
PATHOPTS(path-options-list), PATHMODE(path-mode-list),
PATHDISP(KEEP|DELETE[,KEEP|DELETE]), FILEDATA(TEXT|BINARY), REUSE,
LIKE('model-dsn')
```

In addition to these parameters, CBLi ALLOCATE supports the following:

```
SUBSYS(subsys-name[,subsys-parm]...)
```

Directs the allocation request to the specified subsystem name with optional parameters. Null parameters can be specified by leaving a subsys-parm empty.

Examples:

```
alloc f(sysin) dsn('cbl.ssc.ctl(ssdemo01)') shr reuse
Allocate an existing data set.
```

```
alloc f(sysudump) da('nbj.sysudump') cyl space(100,20) lrecl(133) blksize(0)
recfm(v,b,a) new catalog
Allocate a new data set. Note that DCB information may be omitted.
```

```
alloc f(multdsn) da('cbl.ssc.ctl(ssdemom1)' 'cbl.ssc.ctl(ssdemom2)') shr
Allocate a new data set list.
```


Where more than one command is to be executed, they should be invoked via a macro name (in storage or saved to disk) or via the IMMEDIATE command. The LINEND character is not respected in a CREATE BUTTON command.

Examples:

```
create button 0 80 /HW/msg Hello World/
crea but col yell uscore press yell rev 1 1 /HD/macro HD/
crea but col gr non 1 5 /Ring/imm 'ext ring';do i=1 to ring.0;'msg' ring.i;end/
crea but col bl non press wh rev cursor 1 10 /ColN/imm 'ext cursor';'msg' cursor.2/
crea but press turq rev pass 0 85 #CL#cmsg imm 'stream on';'cl/xxx/';'stream off'#
```

DESTROY BUTTON Command

Syntax:

```
>>-- DEStroy -- BUTton ---/name/---><
      |           |
      +--- * -----+
```

Description:

Destroy a specified button or all buttons generated by a previous CREATE BUTTON command.

As for CREATE BUTTON, any supported delimiter character may be used to encompass the name string so long as the delimiter character used does not appear in the name string.

Parameters:

name	The name of the button to be destroyed.
*	(asterisk) indicates that all buttons are to be destroyed.

Examples:

```
destroy button /HW/
destroy button *
```

EDIT Command Initialisation options

Syntax:

```
>>---+--- Edit ---+-----+-----+-----+-----+-----+-----+-----><
      |           |           |           |           |           |           |
      +--- Kedit ---+   +--- fileid ---+   +- ( ---+--- option ---+---+
      |           |           |           |           |           |           |
      +--- Xedit ---+           |           |           |           |           |
                                   +-----+
```

Description:

CBLi EDIT command initialisation options are supported as for CBLi EDIT. See CBLi CLI commands above.

LIST will list global edit variables.
LISTF will list file edit variables.

varname Edit variable name.

value Edit variable string value.
Must be a single token for SET/SETF.

Examples:

```
editv set tmp nbj.tmp work nbj.work.txt user guest
editv set1 docs All of this is assigned to "docs"
editv list
```

EXTRACT Command Options Added

Syntax:

```
>>-- EXTRACT --+-- option --+--><
      ^                |
      |                |
      +-----+-----+
```

The EXTRACT command allows the user to extract information about the current edit environment for use in CBLi macros. The values for each EXTRACT option are assigned to REXX stem variables.

Where the definition of an option may be set by the user, a more detailed description of that option may be found under the relevant SET command.

The following EXTRACT options have been introduced in CBLi 1.20.

DEFPROFile	defprofile.0	1	
	defprofile.1		Name of default profile macro.
DSORG	dsorg.0	1	
	dsorg.1		PDS SEQ KSDS ESDS RRDS LDS CMS LIBR
FLSCReen	flscreen.0	2	
	flscreen.1		File line number of first text line visible in the window view.
	flscreen.2		File line number of last text line visible in the window view.
HSCROLLCursor	hscrollcursor.0	1	
	hscrollcursor.1		ON OFF
ISPFMODE	ispfmode.0	1	
	ispfmode.1		ON OFF
KEY	key.0	2	
	key.1		Start column of the primary key in the edited KSDS data set.
	key.2		End column of the primary key in the edited KSDS data set.
LCOLor	lcolor.0		Number of SET LCOLOR commands in effect for the current file.

	<code>lcolor.i</code>	All SET LCOLOR definitions and their associated parameters in the order in which they were entered (i=1 to lcolor.0).
		Each lcolor.i variable contains the following information: <ol style="list-style-type: none"> 1. The string line-target (as entered on SET LCOLOR.) Note that the line-target may contain more than one token. e.g. "word /to/", "/string containing blanks/", etc. 2. Colour in lower case. 3. Extended highlighting in lower case. 4. Name associated with the SET LCOLOR command in upper case. If no name was specified, the target string is used. 5. "case" in lower case. 6. Either "respect" or "ignore" in lower case. (The prevailing SET CASE value for string target search when SET LCOLOR was issued.) 7. "zone" in lower case. 8. Left zone value. (The prevailing SET ZONE left zone value when SET LCOLOR was issued.) 9. Right zone value. (The prevailing SET ZONE right zone value when SET LCOLOR was issued.)
		e.g. <pre> /=AB/ pink default /=AB/ case respect zone 1 1 /xYZ/ red revvideo sll case ignore zone 8 20 </pre>
LCOLour	<code>lcolour.0</code>	Same as LCOLOR.
	<code>lcolour.i</code>	
LRECL	<code>lrecl.0</code>	2
	<code>lrecl.1</code>	Defined LRECL value of the current file.
	<code>lrecl.2</code>	Length of longest record on load of the current file.
MDIList	<code>mdilist.0</code>	1
	<code>mdilist.1</code>	ON OFF
NBWindow	<code>nbwindow.0</code>	3
	<code>nbwindow.1</code>	Number of MDI child window edit views.
	<code>nbwindow.2</code>	Number of edit views of the current file.
	<code>nbwindow.3</code>	Edit view number of the current file.
RANGE	<code>range.0</code>	2
	<code>range.1</code>	Line number of first line in range
	<code>range.2</code>	Line number of last line in range. (Equal to one less than RANGE.1 if no lines in current range)
RING	<code>ring.0</code>	Number of files being edited in the ring.
	<code>ring.i</code>	Information on each file being edited in the ring (i=1 to ring.0).

The tokens returned in ring.i are:

1. Fileid.
2. Line number of current line.
3. Column number of current column.
4. File Size (number of lines.)
5. Alteration count.

e.g.

```
CBL.CMX(NBJ) Line=23 Col=1 Size=429
Alt=0,0
```

```
NBJ.TEST Line=0 Col=67 Size=1 Alt=7,7
```

SCOLor

scolor.0

Number of SET SCOLOR commands in effect for the current file.

scolor.i

All SET SCOLOR definitions and their associated parameters in the order in which they were entered (i=1 to scolor.0).

Each scolor.i variable contains the following information:

1. The string line-target (as entered on SET SCOLOR.)
Note that the line-target may contain more than one token. e.g. "word /to/", "/string containing blanks/", etc.
2. Colour in lower case.
3. Extended highlighting in lower case.
4. Name associated with the SET SCOLOR command in upper case. If no name was specified, the target string is used.
5. "case" in lower case.
6. Either "respect" or "ignore" in lower case. (The prevailing SET CASE value for string target search when SET SCOLOR was issued.)
7. "zone" in lower case.
8. Left zone value. (The prevailing SET ZONE left zone value when SET SCOLOR was issued.)
9. Right zone value. (The prevailing SET ZONE right zone value when SET SCOLOR was issued.)

e.g.

```
/</ red default /</ case respect zone 1 *
'IQ00' yellow uscore Q1 case ignore zone
20 40
```

SCOLour

scolour.0

Same as LCOLOR.

scolour.i

SCReen

screen.0

2

screen.1

Depth of 3270 terminal screen.

screen.2

Width of 3270 terminal screen.

SYNonym

synonym.0

1

synonym.1

ON|OFF

SYNonym *

synonym.0

Number of synonyms defined.

synonym.i

Extracts all defined synonyms in the order in which they were entered. (i=1 to synonym.0).

The format of the tokens returned in synonym.i are:

1. New name.
2. Synonym definition.

THIGHLight	thighlight.0	2
	thighlight.1	ON OFF
	thighlight.2	ALL FIRST
WINNAME	winname.0	4
	winname.1	CBLi window name for the current MDI child edit view.
	winname.2	Title bar contents for the current MDI child edit view.
	winname.3	CBLi window name for the current MDI parent.
	winname.4	Title bar contents for the current MDI parent.
WINPOS	winpos.0	4
	winpos.1	Row number in the client area of the current edit view.
	winpos.2	Column number in the client area of the current edit view.
	winpos.3	Row number in the screen display of the current frame window.
	winpos.4	Column number in the screen display of the current frame window.
WINSIZE	winsize.0	10
	winsize.1	Number of rows in the current edit view.
	winsize.2	Number of columns in the current edit view.
	winsize.3	Number of rows in the current edit view's display area.
	winsize.4	Number of columns in the current edit view's display area.
	winsize.5	Number of rows in the parent window.
	winsize.6	Number of columns in the parent window.
	winsize.7	Number of rows in the parent window's display area.
	winsize.8	Number of columns in the parent window's display area.
	winsize.9	Number of rows in the 3270 terminal.
	winsize.10	Number of columns in the 3270 terminal.

FREE Command

Syntax:

```
>>-- FREE -- freeparms ---->>
```

Description:

The FREE command may be used to:

1. Dynamically unallocate a single data set.
2. Override the disposition of allocated data sets.
3. Override the output class.

FREE allows users to unallocate files whether or not a TSO environment is available. The syntax of the command closely matches that of the TSO FREE command and so most FREE commands, executed without TSO as a prefix, will give the same results.

FREE is identical to the ALLOCATE -FREE command and is supported for MVS only.

Parameters:

`freeparms` Parameters supported by the TSO FREE command as follow:

```
DDname(ddname) or File(ddname), DSName('dsn') or DATaset('dsn'), KEEP,
DELETE, CATALOG, UNCATALOG, SYSOUT(class) and SPIN(UNALLOC)
```

Examples:

```
free f(indd)
Unallocate an existing data set.
```

```
free f(ixnbj) keep
Unallocate a ddname overriding the disposition with KEEP.
```

LESS & MORE Commands Parameter TAG Added

Syntax:

```
>>-- MORE ---+-----+--- line-target --><
          |           |
          +--- TAG ---+

>>-- LESS ---+-----+--- line-target --><
          |           |
          +--- TAG ---+
```

The MORE and LESS commands may be used to further display and hide lines that contain the specified line-target, following an ALL command.

The TAG command is used to set the tag bit on for specific lines in the display.

The TAG parameter may be inserted before the line-target of a MORE and LESS command to further add or remove the tag bit from lines in the display that satisfy the specified line-target.

The MORE TAG command does not affect lines that already have the tag bit on.

Lines in the display that have the tag bit on are automatically highlighted.

Examples:

```
more tag /##/
Tag (highlight) lines that contain the string "##". (Lines that are already have the tag bit on are unaffected.)
```

```
less tag /###/
Remove the tag bit and highlight from lines containing the string "###".
```

LIST Command

Syntax:

```
>>-- LIst -- listtype -- /listparms/ ----->
      |                               | | |
      +- STEM stemname +- +- STRIP +-
      |                               |
      +- FILE filename +-

  +- Lines ---+
  >-----><
  |           |
  +- Columns +-   +- SUBset /selectclause/ +-<
```

Description:

Extract the data returned by a CBLi list function and place it either in a file to be edited in the current ring, or, if executed in a macro, in an array of REXX variables.

Parameters:

- listtype** The CBLi list type function to extract. This must be one of the following:
- AMS List IDCAMS command output.
 - APE List Assembler Program Environment. This is the list of modules in the current version of CBLi.
 - FS List records that match a search string.
 - FSEARCH
 - LA List allocated datasets.
 - ALLOcated
 - LC List catalog entries.
 - CATalog
 - LD List dataset attributes.
 - DATasets
 - DSNs
 - FILEs
 - LJQ List MVS enqueues for a given job.
 - JOBEQueuees
 - LL List library members.
 - LIBrary
 - LQ List MVS enqueues.
 - ENQueuees
 - LV List VTOC entries.
 - VTOC
 - LVOL List DASD volumes.
 - VOLumes
 - LX List VTOC extents.
 - EXTents
 - SWA List Attributes of all open windows.
- listparm** A parameter string passed to the list function. This string must always be present. If it contains no blanks or special characters it can be blank-delimited, otherwise it must be delimited by a pair of special characters. If a null string is required it can be given as // for example.
- FILE** A keyword parameter which requests that the LIst command places the listed data in a file for the user to edit. This keyword may be followed by a file name. This is the name used for the edit view but the data is not written to disk. The user may file the data if required.
- This option is the default if the LIst command is issued from the command line or via a function key.
- STEM** A keyword parameter which requests that the LIst command places the listed data in an array of REXX variables with this stem name.
- This option is the default if the LIst command is issued from a macro.

If the stem name is not given then the list type name is used.

If the stem name does not end with a period then one is added.

STRIP A keyword parameter which requests that the data is stripped of leading and trailing blanks before being placed in REXX variables.

Lines A keyword parameter which requests that the Llist command returns complete list lines.

If the FILE option is used then the list column header and list lines are placed in an edit window in the current ring with the file name specified (but not actually written to disk).

If the STEM option is used then the following variables are set:

stem.0	The number of list lines.
stem.i	The ith list line.
stem.columns	The list column header line.

Columns A keyword parameter which requests that the Llist command returns the list data in column variables.

If the FILE option is used, or the Llist command is not issued from a macro, this option is invalid. An error message is issued and no data is returned.

If the STEM option is used then the following variables are set:

stem.0	The number of list columns.
stem.i.colname	The value of column colname for the ith list line.
stem.columns.0	The number of columns.
stem.columns.j	The name of the jth column.

SUBset A keyword parameter which must be followed by a list subset command delimited by a special character.

The subset command can contain one or more of the following (in any order):

select clause	Use the select clause to define the list of column names to return. The default is * which means all columns.
where clause	Use the where clause to apply a filter which restricts the lines returned based on a condition defined in terms of the
sort (order by) clause	values in the columns of the list. Use the sort or order by clause to determine the order in which the list lines are returned based on column values.

See *Selecting, Sorting and Filtering* for details of the syntax of these clauses.

Examples:

```
list volumes/CBL*/ file lac.vollist subset/select unit,vol,freecyl/
li alloc /SYSEXEC/
li alloc SYSPROC
li library/cbl.cbli120.asm(edt*)/ stem ll.
li library/cbl.cbli120.asm(edt*)/ file lac.liblist
li catalog/cbl.cbli*.**/ file lac.catlist
li dataset/cbl.cbli*.**/ file lac.dsnlist
li vtoc /cblmct/ file lac.vtoclist subset / select vol,dsn,org /
li extent /cblmct/ file lac.extlist
li enqueue/sysdsn LAC./ file lac.enqlist
li enqueue/spfedit LAC./ file lac.enqlist
```

OVERLAY Command

Syntax:

```
>>-- Overlay --- string -----><
```

Description:

Overlay text in the focus line with the specified text string starting at column 1. The text string begins immediately after the single separating blank that follows the OVERLAY command verb.

Characters in the focus line that correspond to blanks in the supplied overlay string are unchanged. In order to overlay a character with a blank, the "_" (underscore) character should be specified in the corresponding position of the overlay string.

All other characters supplied in the overlay string replace characters in corresponding positions in the focus line.

Parameters:

`string` Overlay text string.
Where more than 1 blank separates the text string from the command verb, the additional separating blanks are treated as part of the text string.

Examples:

```
overlay abc
Overlay text at column 1 of the focus line with "abc".
```

```
O ABC_X YZ
Where original text at column 1 of the focus line is: 0123456789
After the OVERLAY command text is: 0ABC X6YZ9
```

QUERY Command Options Added

Syntax:

```
>>-- Query ----- option -----><
```

The QUERY command allows the user to query information about the current edit environment. The information is displayed on the message line.

Where the definition of an option may be set by the user, a more detailed description of that option may be found under the relevant SET command.

The following QUERY options have been introduced in CBLi 1.20.

DEFPROFile	Displays the name of default profile macro.
DSORG	Displays the Data Set Organisation of the current file. (PDS, SEQ, KSDS, ESDS, RRDS, LDS, CMS or LIBR)
FLSCReen	Displays the file line number of the first and last text line visible in the window view.
HSCROLLCursor	Displays the current setting for HSCROLLCURSOR (ON or OFF).
ISPFMODE	Displays the current setting for ISPFMODE (ON or OFF).
KEY	Displays the start and end columns containing the key in the current KSDS file.
LCOLor LCOLour	Displays the active LCOLOR definitions in the order that they were entered. The output also displays the SET CASE and SET ZONE options that were active when each LCOLOR command was issued.
LRECL	Displays the defined LRECL value for the current file and the longest record encountered when the file was loaded for edit.

MACRO <macroname>	Displays the name and contents of each macro loaded into storage by a DEFINE command. If <macroname> is specified, then output is restricted to the specified macro. The following message marks the start of the defined macro: "MACRO Name=<macroname> Text=Loaded from <location> ..."
MDIList	Displays the current setting for MDILIST (ON or OFF).
NBWindow	Displays the total number of open edit views, the number of edit views displaying the current file and the current edit view number for the current file.
RANGE	Displays the line number of the first and last lines in the current range.
REDO REDO *	Same as UNDO.
RING	Displays the number of files edited in the ring of edit views, followed by information on each edited file. (i.e. Alteration counts, number of records and fileid.)
SCOLour SCOLor	Displays the active SCOLOR definitions in the order that they were entered. The output also displays the SET CASE and SET ZONE options that were active when each SCOLOR command was issued.
SCReen	Displays the number of rows and columns available in the 3270 terminal screen.
SYNonym	Displays the current setting for SYNONYM (ON or OFF).
SYNonym *	Displays all defined synonyms in the order in which they were entered. (i.e. synonym name and definition.)
THIGHlight	Displays the current setting for THIGHLIGHT (ON or OFF followed by ALL or FIRST).
UNDO *	Displays change levels (Edit TRansactions - ETR) and a description of the functions performed for each change level (Edit Transaction Element - ETE) that exist in UNDO storage and may be undone using the UNDO command.
WINNAME	Displays the internal name for current edit view and the frame window (MDI parent window).
WINPOS	Displays the respective row, column positions of the current edit view within the client area and the current frame window within the screen display.
WINSIZE	Displays the rows x columns size of the current edit view, the current edit view's display area, the MDI parent window, the MDI parent window's display area and the 3270 terminal screen.

REFRESH Command

Syntax:

```
>>-- REFRESH -----<<
```

Description:

For use in a CBL REXX macro, the REFRESH command will refresh the edit display during execution of the macro. The display is not normally updated until a macro is completed or unless keyboard input is required.

Note: Frequent refresh of the display within a macro can increase the macro's run time.

Examples:

```
imm do 10; 'addl'; 'refresh'; end
```

Add 10 blank lines, one at a time, and refresh the view after each line added.

REPLACE Command

Syntax:

```
>>-- Replace --+-----+----->>
          |           |
          +-- string --+
```

Description:

Replace the focus line with the specified text string. The text string begins immediately after the single separating blank that follows the REPLACE command verb.

Parameters:

`string` Replace text string.
Where more than 1 blank separates the text string from the command verb, the additional separating blanks are treated as part of the text string.

If `string` is omitted, the focus line is replaced with a blank line.

Examples:

```
r Hello World
```

Focus line is replaced with "Hello World starting in column 1.

```
replace
```

Focus line is replaced with a blank line.

SHIFT Command

Syntax:

```
>>-- SHift --+--- Left ---+----- 1 -----+>>
          |           |           |           |
          +-- Right ---+-- n --+ +-- group-target ---+
```

Description:

Shift text to the LEFT or RIGHT by the specified number of columns. Text is moved on the focus line and on lines up to, but not including, the line containing the first match for `group-target`.

The SHIFT command affects text from the left zone column or the left boundary of a marked box block, through to the LRECL column. Text that is shifted left beyond the left zone column, the left box block boundary or right beyond the LRECL column, is truncated.

Parameters:

`Left`
`Right` Direction in which to shift text.

`n` Integer specifying the number of columns by which the text is to be shifted.

`group-target` Group-target condition defining the end of the target area for the command. If the `group-target` is not satisfied then the SHIFT command will fail.

Examples:

```
shift right
```

Starting at the left zone column, text at the focus line is shifted 1 column to the right.

```
sh r 6 all
```

Starting at the left zone column, text in all lines of the edit view is shifted 6 columns to the right.

```
sh L 10 /abc/
```

Starting at the left zone column, text at the focus line and in all lines up to, but not including, the first line following the focus line to contain string "abc", is shifted 10 columns to the left.

```
sh L 1 block
```

Starting at the left boundary of the marked block, text at the focus line and in all lines within the marked block, is shifted 1 column to the left.

SOS Command Parameter REFRESH

Syntax:

```
>>-- SOS -- REFResh -----><
```

The SOS command performs Screen Operation Simulation to action cursor placement and text editing in CBL e macros.

Action REFRESH is supported to refresh the entire 3270 display. This should be used when some external function has taken control of the 3270 display and updated its contents.

SOS REFRESH performs the function usually assigned to PA2 in TSO and ISPF.

SYNEX Command

Syntax:

```
>>-- SYNEX -- command -----><
```

The SYNEX (SYNONYM EXecute) command is for use in CBL e REXX macros.

When SYNONYM ON is in effect, a command entered from a CBL e command line is automatically checked to determine whether it is the name of a defined synonym. If so, the action taken will be that specified by the synonym definition.

By default, CBL e macros do not perform any synonym processing. Prefixing the command with SYNEX when SYNONYM ON is in effect, will force CBL e to check whether the command to be executed has had its behaviour defined via a SET SYNONYM command. If not, the command is executed as normal.

Parameters:

`command` Any synonym name, CBL e command or macro name followed by its parameters.

Examples:

```
synex CC red
```

Check existence of "CC" as a synonym name. Failing that, treat it as a CBL e command or macro name.

WINDOW Command Parameters Added

Syntax:

```

>>-- WINDow  --+----- NEXTwindow  -----+----->>
|
|+----- PREVwindow  -----+
|
|+----- CAScade  -----+
|
|               +-- HOR  -----+
|+----- TILE  -----+
|               +-- Vert  -----+
|
|+----- ARRANGEIcons  -----+
|
|+----- NEWwindow  -----+
|
|+----- HEX  -----+
|
|               +-- DOCument  --+
|+----- CLOse  -----+
|               +-- FRame  -----+
|               +-- FILE  -----+
|
|+----- MENUmode  -----+
|               +-- DOCument  --+
|               +-- FRame  -----+
|               +-- FILE  -----+
|               +-- Edit  -----+
|               +-- ACTions  ----+
|               +-- OPTions  ----+
|               +-- WINDow  -----+
|               +-- Help  -----+
|               +-- DOCument  --+
|
|+----- RESTore  ---+
|               +-- FRame  -----+
|+----- MINimize  ---+
|+----- MINimize  ---+
|               +-- MAXimize  ---+
|+----- MAXimize  ---+

```

Description:

Perform window focusing, positioning and sizing operations on the current edit (document) view or MDI parent (frame) window.

Parameters:

NEXTWINDOW	Place focus on the next edit view in the ring.
PREVWINDOW	Place focus on the previous edit view in the ring.
CASCADE	Cascade all MDI child windows within the MDI parent window.
TILE	Tile all MDI child windows within the MDI parent window.
HOR	horizontally (default.)
VERT	vertically.
ARRANGEICONS	Arrange all minimised MDI child windows so that they are lined up along the bottom of the MDI parent display area.

NEWWINDOW	Open an new edit view for the data in the current edit view.
HEX	Open a hex view of the focus line. A hex view is a storage display window that contains a hexadecimal and character representation of the line of edited data. The contents of the line may be updated by the user in either of the data representations, simply by overtyping the existing data and hitting <Enter>. PF7 and PF8 scroll up and down respectively through the displayed data, whereas PF10 and PF11 scroll backwards and forwards through the lines of edited data.
CLOSE	Close the specified window type. DOCUMENT Current document window (edit view) (default.) FRAME Frame window (MDI parent) and all its child windows. FILE All edit views that contain the file currently being edited.
MENUMODE	Places the cursor at the specified menu bar item and, where possible, opens the drop down menu. If no item is specified the cursor is simply placed at the first item of the menu bar. DOCUMENT Document window (edit view) system menu. FRAME Frame window (MDI parent) system menu. FILE File drop down menu. EDIT Edit drop down menu. ACTIONS Actions drop down menu. OPTIONS Options drop down menu. WINDOW Window drop down menu. HELP Help item.
MAXIMISE MAXIMIZE	Maximise the current edit view (DOCUMENT), the default, or the MDI parent (FRAME) window.
MINIMISE MINIMIZE	Minimise the current edit view (DOCUMENT), the default, or the MDI parent (FRAME) window.
RESTORE	Restore the current edit view (DOCUMENT), the default, or the MDI parent (FRAME) window back to its original state, prior to being maximised or minimised.

Examples:

```
win max
Maximise the current edit view.

win new
Open a new edit view for the data in the current edit view.

win clo file
Close all edit views containing the file currently being edited.
```

Section 07: CBL e SET Commands

SET COLOUR Command Parameters Added

Syntax:

```
>>--+-----+--+ COLOr  --+--+ Arrow  -----+--+ Blue  -----+--+ NONE  -----+>><
    |         |  |          |  +- PROMpt -----+ |          |  |          |          |
    +- Set  -+  +- COLOur -+  +- COMmand -----+ +- Red  -----+ +- BLInk -----+
                                     +- Cmdline -----+ +- Pink -----+ +- REVvideo -+
                                                         +- Green -----+ +- Uscore -----+
                                                         |
                                                         +- Turquoise -+
                                                         |
                                                         +- Yellow  -----+
                                                         |
                                                         +- White  -----+
                                                         |
                                                         +- Default  -+>><
```

Description:

Set colours for fields in the CBL e window.

In addition to the existing parameters, ARROW (PROMPT) and COMMAND (CMDLINE) are now supported for command SET COLOUR.

Parameters:

ARROW PROMPT	Command prompt "Command>". Default colour is WHITE.
COMMAND CMDLINE	Command line text. Default colour is RED.

Examples:

```
set colour prompt green uscore
col com turq rev
```

SET DEFPROFILE Command

Syntax:

```
>>--+-----+--- DEFPROFile --- macroname ----->><
    |         |
    +- Set  -+>><
```

Description:

Define the name of the default profile macro that gets executed every time a new file is added to the ring of edited files.

SET DEFPROFILE takes effect at the Global level.

Parameters:

macroname	Name of a CBL e REXX macro. This may be the full data set name and member name or simply the name of a member found in the macropath libraries.
-----------	--

Examples:

```
defprof cbleprof
defprof cbl.test.cble(profile)
```

SET DSORG Command

Syntax:

```
>>+-----+----- DSORG ---+ PDS -----+----->>
    |         |         |         |         |
    +- Set  +-         +- SEQ -----+
    |         |         |         |         |
    +- KSDS -- kp1 --- kp2 ---+
    |         |         |         |         |
    +- ESDS -----+
    |         |         |         |         |
    +- RRDS -----+
```

Description:

Change the data set organisation for the current file. DSORG may only be set for data sets that have not yet been allocated or defined.

SET DSORG takes effect at the File level.

When changing the DSORG to PDS, CBL e removes the FNAME (penultimate) qualifier from the DSN and uses it as the member name. If only 2 qualifiers exist in the original DSN, then the command will fail.

Coversely, when changing DSORG from PDS to any other supported organisation, the member name is inserted into the DSN as the penultimate qualifier.

Note: SET FNAME is synonymous with SET MBR.

Parameters:

PDS Supported data set organisations.
 SEQ
 KSDS
 ESDS
 RRDS

kp1 Start and end positions of the key field. These are required by CBL e to edit KSDS data sets.
 kp2

Examples:

```
dsorg ksds 1 16
dsorg pds
```

SET HSCROLLCURSOR Command

Syntax:

```
>>+-----+----- HSCROLLCursor ---+-- ON ---+----->>
    |         |         |         |         |
    +- Set  +-         +- OFF ---+
    |         |         |         |         |
```

Description:

Controls horizontal scrolling when the target of a successful CLOCATE command is outside the displayed width of the edit view.

When HSCROLLCURSOR is in effect and the target column is outside the edit view, the view is scrolled horizontally so that the target (new focus) column becomes the first column displayed in the edit view.

SET HSCROLLCURSOR takes effect at the View level.

Parameters:

ON
OFF

Set HSCROLLCURSOR ON or OFF.
Default is OFF.

Examples:

```
hscrollc on
```

SET ISPFMODE Command

Syntax:

```
>>--+-----+--- ISPFMode ---+--- ON ---+-----><
      |         |           |         |
      +- Set  -+           +--- OFF ---+
```

Description:

When running in an ISPF environment, SET ISPFMODE controls whether 3270 screen I/O is managed by ISPF (ON) or CBL3270 (OFF). This is the same as executing the CBLi command ISPF with no parameters which toggles ISPF screen management on and off.

If SET ISPFMODE is issued in a non-ISPF environment, the following error message is returned:

```
EDT095E ISPF is not available in the current environment.
```

With ISPFMODE ON, the menu bar item **Swap** is added to the CBLi Main Menu which, if selected, will execute **ISPF SWAP** to display an ISPF split screen.

SET ISPFMODE takes effect at the Global level.

Parameters:

ON
OFF

Set ISPFMODE ON or OFF.
In an ISPF environment, the default is ON. Otherwise ISPF is OFF.

SET KEY Command

Syntax:

```
>>--+-----+--- KEY --- kp1 -- kp2 -----><
      |         |
      +- Set  -+
```

Description:

Change the key position and key length for the current KSDS file. KEY may only be set for KSDS data sets that have not yet been defined.

SET KEY takes effect at the File level.

Parameters:

kp1
kp2

Start and end positions of the key field. Both are mandatory for SET KEY.

Examples:

```
key 101 116
```


Examples:

```
set lcolour /==/ green uscore equal2
lcol /alloc/ yel rev "tso allocs"
lcol word level blue blink
```

SET LINEFLAG Command**Syntax:**

```

>>+-----+ LINEFLAG +-----+ +----- 1 -----+
      |         |         |         |         |         |
      +- Set -+         | CHAnge |         |         |         |
      |         |         |         |         |         |
      +- NOCHAnge -+    |         |         |         |         |
      |         |         |         |         |         |
      +- NEW -----+   |         |         |         |         |
      |         |         |         |         |         |
      +- NONEW -----+  |         |         |         |         |
      |         |         |         |         |         |
      +- TAG -----+   |         |         |         |         |
      |         |         |         |         |         |
      +- NOTAG -----+  |         |         |         |         |

```

Description:

Each line of a file in an edit view has three associated flag bits, namely the CHANGE bit, NEW bit and the TAG bit.

By default, when CBLe loads a file for edit, all three flag bits are set off. The flag bits are set automatically by CBLe as follow:

CHANGE Set ON when a line is changed, added, moved or sorted.

NEW Set ON when a line is added.

TAG Set on by TAG and MORE TAG and set off by LESS TAG.

SET LINEFLAG allows the user to manually set the flag bits on and off, for lines contained in the specified target area.

SET LINEFLAG takes effect at the File level.

Parameters:

CHANGE	Set the change flag bit on/off.
NOCHANGE	Default is NOCHANGE.
NEW	Set the new flag bit on/off.
NONEW	Default is NONEW.
TAG	Set the tag flag bit on/off.
NOTAG	Default is NOTAG.
group-target	Group-target condition defining the end of the source target area. If the group-target is not satisfied then the DELETE command will fail.

Examples:

```
set lineflag nonew
Set the new flag bit off for the focus line only.
```

```
set lineflag new nochange /##/
Set the new flag bit on and the change flag bit off for the focus line and all lines up to, but not including, the first line to contain the string "##".
```

```
set lineflag tag 20
Set the tag flag bit on for the focus line and the 19 lines that follow.
```


* (Asterisk) references all SET SCOLLOUR entries.
 OFF Remove a SET SCOLLOUR entry and undo any colouring caused by that entry.

Examples:

```
set colour /*/ green none asterisk
scol help yel uscore "help items"
scol pre 'sup' pink blink
```

SET SYNONYM Command**Syntax:**

```
>>--+-----+-- SYNONYM  +--+ name  +-----+-- action  +-----+<<
      | Set  |          |          |         |         |
      +-- Set -+          +--- n ---+         |         |
                        +-----+-- ON  +-----+--
                        |         |         |         |
                        +--- OFF ---+         |         |
```

Description:

SET SYNONYM has the following functions:

1. Controls CBLLe synonym processing.

With SYNONYM ON, CBLLe checks any command verb to be executed as being a defined synonym name. The exception to this is when the command issued is done so via a CBLLe macro. In this case, the SYNEX command must prefix the command to be executed in order to force CBLLe's synonym processing.

With SYNONYM OFF, no synonym checking occurs.

CBLLe's synonym checking may be bypassed by prefixing the command with the COMMAND command.

2. Defines new synonym names and associated actions.

The name token is assigned the specified action. The name may be defined with a minimal truncation of n characters. Thereafter, only the first n characters of the name need be entered to execute the associated action.

Each name/action definition is stored until the MDI application is closed.

SET SYNONYM ON|OFF takes effect at the View level.

SET SYNONYM name takes effect at the Global level.

Parameters:

ON	Set SYNONYM processing ON or OFF.
OFF	Default is ON.
name	Synonym name which may be equal to an existing CBLLe command or macro name.
n	Number of characters that CBLLe will accept as the minimal truncation for name.
action	CBLLe command stream.

Examples:

```
set synonym off
synonym find 1 locate
syn delblank imm 'ext/flscr/';'nomsg all blank';'del *';'all';':'flscreen.1
```


SET WINPOS Command

Syntax:

```

      +- View  +-
      |         |
>>---+-----+--- WINPOS ---+-----+--- row -- col --><
      |         |         |         |
      +- Set  +-         +- FRame +-
  
```

Description:

Position the current edit view at the specified row and column within the frame (MDI parent) client area, or position the current frame (MDI parent) window at the specified row and column of the screen.

A window's positional coordinates refer to the row x column position of the window's system menu button (found on the extreme left of the title bar.) It is this coordinate that is positioned at the new row x column position.

The lowest position within an MDI client area is row 1, column 1, which corresponds to the top left position below the window's menu bar.

The lowest position within the screen is row 1, column 2, which corresponds to the top left position occupied by the CBLi main window's system menu button.

A window cannot be positioned entirely outside its parent (MDI or CBLi main) window. If large row x column values are used, CBLi will ensure that at least 5 characters of the window's title bar remains in view.

Parameters:

FRAME	Position the frame or edit view window.
VIEW	Default is VIEW.
row	A positive, non-zero number specifying the row number at which the window is to be positioned.
col	A positive, non-zero number specifying the column number at which the window is to be positioned.

Examples:

```
winpos view 5 10
```

Position the edit view at row 5, column 10 of the current frame's client area.

```
winpos 10 15
```

Position the edit view at row 10, column 15 of the current frame's client area.

```
winpos frame 1 2
```

Position the frame window at row 1, column 2 of the screen.

SET WINSIZE Command

Syntax:

```

      +- View  +-
      |         |
>>---+-----+--- WINSIZE ---+-----+--- depth -- width --><
      |         |         |         |
      +- Set  +-         +- FRame +-
  
```

Description:

Resize the current edit view or frame (MDI parent) window to the specified depth and width.

CBLi does not allow an edit view display to be resized to a window less than 5 rows deep by 10 columns wide or to a window size greater than that of the MDI parent window. Similarly, CBLi does not allow an MDI parent window display to be resized to a window of less than 10 rows deep by 10 columns wide or to a window size greater than that of the CBLi main window.

If WINPOS depth/width values are used that exceed these limits, the window is resized to the allowable limit.

Parameters:

FRAME Resize the frame or edit view window.
VIEW

depth A positive, non-zero number specifying the number of rows to which the window is to be resized.

width A positive, non-zero number specifying the number of columns to which the window is to be resized.

Examples:

```
winsiz view 28 80
```

```
winsize fra 40 100
```

Section 08: CBLLe Edit Macros

CBL has included the following new or updated CBLLe REXX macros with CBLi 1.20.

The detailed description for each of the following macros is documented in the macro member itself. Because macros are subject to change by users, only a brief description of macros supplied by CBL may be found in the CBLi Help documentation.

BOX Macro

Syntax:

```
>>-- BOX ---+-----+--- command -----><
          |         |
          +- # -+
          |         |
          +- ; -+
```

Description:

The BOX macro allows the user to restrict the area affected by the specified command stream to be the area defined by a marked block. This relieves the user from having to manually set and subsequently reset the ZONE and RANGE values.

The specified command stream is executed after the macro temporarily sets the RANGE and ZONE to be the limits defined by the currently marked block. When the macro completes the RANGE and ZONE is reset.

The CBLLe command may actually be another macro or a number of commands separated by the specified separator character '#' or ';'.

Parameters:

#	Separator character to be used to separate commands that follow.
;	
command	CBLLe command stream.

Examples:

```
box #change /hello/goodbye/* * #c /fat/thin/* *
```

Change only occurrences of "hello" and "fat" that exist within the marked block.

Unlike the CHANGE command where a target of BLOCK is used, this method will truncate or pad text at the block right boundary if the new string is longer or shorter than the original.

```
box delall /abc/
```

Execute the DELALL macro to delete all lines that contain the string "abc" within the marked block.

```
box mi set scol /abc/ green rev
```

Execute the MI macro (temporarily set CASE MIXED IGNORE) with a SET SCOLLOUR command as its parameter. All occurrences of the string "abc" (any case) within the marked block will be coloured.

BOXSEQ Macro

Syntax:

```
>>-- BOXSEQ ---+-----+-----+-----+-----+-----><
          |         |         |         |         |
          +- HEX -+ +- startnum -+ +- incval -+
          |         |         |         |         |
```

Description:

Inserts a sequential stream of numbers, one number right adjusted in each line of a marked box block.

The sequence begins with the number specified by startnum which gets inserted at the top line of the marked box. Each number in the sequence gets inserted in the line below the last.

HEX may be specified to generate a hexadecimal sequence of numbers.

Note: The hex numbers are in character, not binary, format.

If startnum has a leading zero, then each number in the sequence is padded with leading zeroes up to the left block boundary.

The number sequence may be ascending or descending in increments as governed by incval.

Parameters:

HEX	Generate a hexadecimal sequence.
startnum	First number of the sequence. This may be a positive or negative rational number. Default is 1.
incval	Increment value to be added to the last number in the sequence in order to get the next number. This may be a positive or negative rational number. Default is +1.

Examples:

```
boxseq 10 5
(10, 15, 20, etc.)
```

```
boxseq hex 01E 4
(1E, 22, 26, 2A, 2E, etc.) padded with leading zeroes.
```

```
boxseq -8.53 -1.24
(-8.53, -9.77, -11.01, etc.)
```

CMXAMS Macro

Syntax:

```
>>-- CMXAMS -----<<
```

Description:

Convert an edited IDCAMS DEFINE cluster statement (as generated by a CBLVCAT LISTVCAT DEFINE operation) into a CBLi AMS command suitable for execution using the PF4 (CMDTEXT) key.

Parameters:

CMXAMS does not support any parameters.

DELALL Macro

Syntax:

```
>>-- DELALL -- line-target -----<<
```

Description:

Delete lines from the current file that satisfy the line-target.

Only the focus line, and all displayed lines that follow, are eligible for deletion.

Parameters:

line-target	Line-target condition.
-------------	------------------------

Examples:

```
delall /==/
Delete all lines that contain the string "==" between the focus line and the last line of the file.
```

```
delall 10
Delete every 10th line following the focus line.
```

```
delall word /as/
Delete lines containing the word "as".
```

DELNOT Macro

Syntax:

```
>>-- DELNOT -- string -----<<
```

Description:

Delete lines from the current file that do not contain the specified string.
Only the focus line, and all displayed lines that follow, are eligible for deletion.

Parameters:

`string` String line-target condition.

Examples:

```
delnot /##/
Delete all lines that do not contain the string "##" between the focus line and the last line of the file.
```

GETAVRL Macro

Syntax:

```
>>-- GETAVRL -- fileid -----<<
```

Description:

For CMS and MVS platforms where SELCOPY is installed, GETAVRL returns the average record length of the specified file in a pop-up message window.

Each record of the file is read using SELCOPY and the average LRECL is calculated.

Parameters:

`fileid` Unquoted, full fileid of the required file.
This may be one of the following:

1. CMS fn ft fm.
2. MVS SEQ DSN.
3. MVS VSAM DSN.
4. MVS PDS including member name.

Note: Any TSO prefix should be included in the fileid if required.

Examples:

```
getavrl nbj.iq00123.ksds
getavrl cbl.ssc.ct1(ssdemo01)
```

IALL Macro

Syntax:

```
>>-- IALL ---+-----+--- line-target ---><
      |
      +- linefirst -- linelast -+
      |
      +- colfirst --- collast ---+
```

Description:

ISPF style ALL command performs a case insensitive search for line-target and also supports restriction of the search to a specified range of lines and/or columns. On completion, the cursor is positioned at the first match for line-target.

IALL calls macro ISPFMAC to perform the main processing.

Parameters:

linefirst linelast	Specifies the first and last lines (inclusive) of a range of lines in which the command will operate. linefirst and linelast must each be absolute line number target (e.g. :12, :35) or a named line target (e.g. .nb1, .abc) where linefirst <= linelast.
colfirst collast	Specifies the first and last columns (inclusive) of a range of columns in which the command will operate. colfirst and collast must be positive numeric integers representing absolute column numbers where colfirst <= collast.
line-target	Line-target condition.

Examples:

```
iall 1 10 .a .b /Hello/
iall .a :56 /abc/
iall :10 :34 22 50 /123 456/
```

ISPFMAC Macro

Syntax:

```
>>-- ISPFMAC -- command ---+-----+--- parms ---><
      |
      +- linefirst -- linelast -+
      |
      +- colfirst --- collast ---+
```

Description:

Simulates functionality provided by certain ISPF edit commands whereby the command operates on a restricted range of lines and/or columns, any target search will be treated as case insensitive and wrapping occurs when Top-of-File/End-of-File is encountered.

ISPFMAC may be used to apply the same restrictions when executing CBLed edit commands.

ISPFMAC is called by IALL to simulate ISPF style FIND ALL command.

Parameters:

command	CBLed command verb.
linefirst linelast	Specifies the first and last lines (inclusive) of a range of lines in which the command will operate. linefirst and linelast must each be absolute line number target (e.g. :12, :35) or a named line target (e.g. .nb1, .abc) where linefirst <= linelast.

`colfirst` Specifies the first and last columns (inclusive) of a range of columns in which the command will operate.

`collast`

`colfirst` and `collast` must be positive numeric integers representing absolute column numbers where `colfirst` <= `collast`.

`parms` Parameter and argument string to be passed to the CBLi command.

Examples:

```
ispfmac clocate :15 :25 21 50 /selc/
ispfmac scol :22 .section2 1 15 /</ red none
```

JCLCMX Macro**Syntax:**

```
>>-- JCLCMX -----+-----><
                |         |
                +-- -SAVE fileid --+
```

Description:

For use when running CBLi under ISPF, TSO or as an MVS VTAM application (CBLiVTAM.)

The prime purpose of JCLCMX is to assist use of SELCOPY Interactive where the SELCOPY controls statements are invoked via an MVS batch job stream.

Since SELCOPY Interactive does not yet interpret JCL, JCLCMX may be used to create a temporary, unsaved CMX file that contains ALLOC command equivalents for each DD card in the current JCL file, starting at the current line. Each command generated in the CMX file may be executed using PF4 (CMDTEXT) in the CBLi text editor.

Where the JCL contains an in-stream data set (//ddname DD *), the user is prompted to save the in-stream data to a data set with DSN 'user.JCLCMX.JobName.StepName.FName'. For each in-stream data set, JCLCMX generates an ALLOC, FREE, ERASE and EDIT command for the temporary file.

JCL DD statements that reference a temporary data set (e.g. DSN=&&X.Y.Z) will generate an ALLOC statement with a DSN where the "&&" is replaced by a qualifier equal to the user's logon id (e.g. user.X.Y.Z). JCL defined temporary data sets and data sets allocated with DISP=,DELETE will also generate a FREE and ERASE command.

Where DD statement has ddname SYSIN and belongs to a SELCOPY execution job step, JCLCMX also generates a CBLi SELCOPY command to execute SELCOPY Interactive. e.g.

```
<Selcopy -ctl 'user.JCLCMX.JobName.StepName.FName'
```

If SYSIN is a concatenation of SELCOPY source data sets, then, as for in-stream data sets, the user is prompted to save the concatenated data to a temporary data set with DSN 'user.JCLCMX.JobName.StepName.FName'. Also, FREE, ERASE and EDIT commands are generated for the temporary file.

For concatenated and in-stream SELCOPY SYSIN, a JCLCMX -SAVE macro invocation is also generated to scan for any changes made to the SYSIN temporary data set and, if they exist, apply them to the original source.

Note: The source files appear in the CBLi edit ring with the updates applied but not saved.

Parameters:

`-SAVE fileid` Edit the source data set(s) and apply any changes that have been made to the temporary SELCOPY SYSIN data set. `fileid` is the fully qualified DSN of the temporary SELCOPY SYSIN data set.

The SELCOPY source data set may be the in-stream data in the JCL source.

Examples:

JCL source file CBL.SSC.CTL(NBJ01) contains:

```
//SSDEMO JOB , ,CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1),NOTIFY=NBJ
//STEP1 EXEC PGM=SELCOPY
```


NAM Macro

Syntax:

```
>>-- NAM -----><
```

Description:

Places the current file's fileid on the command line. NAM may be used by all users who find typing long fileids laborious.

Parameters:

None.

PROFIRST Macro

Syntax:

```
>>-- PROFIRST -----><
```

Description:

Called by the PROFILE macro for the first file edited in the ring only. PROFILE is the name of the default profile macro as supplied by CBL.

The default profile macro is executed every time a file is edited, therefore, in order to perform tasks only once when the MDI application (CBLed or SDB) is started, PROFILE macro tests for this condition and calls PROFIRST if it s true.

PROFIRST performs tasks such as creating useful button functions for the edit session.

Parameters:

None.

QX Macro

Syntax:

```
>>-- QX ---- ext_operand -----><
```

Description:

Displays a message reporting the current value of each stem variable generated for the specified EXTRACT operand.

Differences often exist between the way values are reported by QUERY and the way they are assigned to REXX stem variables by EXTRACT. QX allows the user to display the extracted values without having to write an IMMEDIATE macro.

Parameters:

<code>ext_operand</code>	An un-abbreviated CBLed extract operand.
--------------------------	--

Examples:

```
qx curline
qx inivar
```

SDBPOPUP Macro

Syntax:

```
>>-- SDBPOPUP -----><
```

Description:

SELCOPY Interactive popup menu which, by default, is mapped to PF4 in SELCOPY Interactive CBLe edit views. Where the 3270 emulation software allows it, the Mouse Right button should be mapped to PF4 key.

The following is an example of the SDBPOPUP menu:

```
| CmdText  
|-----|  
| Show Pos "@arr"  
|----- "@arr+marre-1"  
|----- "@arr+marre-1" <edit>  
| Track   "@arr"  
|----- "@arr+marre-1"  
|----- "@arr+marre-1" <edit>  
|-----|  
| Track List  
|-----|  
| Break <toggle>  
|-----|  
| Window Layout  
|-----|  
| Debug Keys  
|-----|
```

CmdText

Issue the edit command CMDTEXT for the focus line.

Show Pos "expression"

Open a POS storage display window starting at the position defined by expression. The expressions available are determined by the text pointed to by the cursor at the time of issuing SDBPOPUP. The expressions generated are:

1. The single token that forms part of a larger positional expression.
2. The full positional expression.

Show Pos "expression" <edit>

Do not open a POS storage display window but instead place the generated "WINDOW POS expression" command on the command line ready for edit and submission by the user.

Track "expression"

Invokes the SDBTRACK macro to issue the TRACK command to highlight positions in storage display windows referenced by the positional expression. SDBTRACK also highlights all occurrences of the expression in any open edit window.

The expressions available are determined in the same way as for "Show Pos" above.

Track "expression" <edit>

Place the generated "SDBTRACK expression" macro invocation on the command line ready for edit and submission by the user.

Track List

Displays a pop-up menu of all positions actively being tracked. The user can select a menu entry to deactivate tracking for that expression. Alternatively, selecting "All" will deactivate all expression tracking.

Break <toggle>

Toggle a break point on or off for the SELCOPY operation at the cursor.

Window Layout

For use with smaller 3270 terminal sizes where all SELCOPY Interactive MDI child windows are opened in a maximised state.

This menu item invokes the SDBWINX macro which opens a pop-up menu with the following items:

1. "Save" the window position and size of the current window or "All" windows.

2. "Restore" the current window or "All" windows to the saved position and size.
3. "Default" to restore windows to a non-maximised state and then reposition and resize the current window or "All" windows to a standard configuration.

Debug/Edit Keys

Set pkeys for debugging or editing. The following PFKeys are redefined:

PFKey	Edit	Debug
PF1	SOS LINEADD	STEPOVER
PF2	DUPLICATE	STEPINTO
PF13	SOS LINEDEL	GO
PF14	SPLTJOIN	BR ON CURSOR PERM

Parameters:

None.

SDBTRACK Macro

Syntax:

```
>>-- SDBTRACK -- expression ---+-----+---<<
                                |         |
                                +--- colour ---+
                                |         |
                                +--- OFF  -----+
                                |         |
```

Description:

Executes the SELCOPY Interactive TRACK command to begin or end tracking of the specified position in all open storage display windows. SDBTRACK also highlights all occurrences of the POS expression in all open edit views.

Both the tracked position and the POS expression string are highlighted in the nominated colour with REvVideo extended highlighting.

SDBTRACK is called from the SDBPOPUP macro when the Track menu items are selected.

Parameters:

expression Any valid POS expression as supported by the TRACK command.

colour One of the following colours: Default, Blue, Red, Pink, Green, Turquoise, Yellow, White

If colour is not specified, a pop-up menu is opened prompting the user to select a colour or OFF.

OFF Switch off tracking for the specified expression.

Examples:

```
sdbtrack @abc+lrecl-arraylen+1 Green
sdbtrack inrec+@ Red
```

SDBWINX Macro

Syntax:

```
>>-- SDBWINX ---+-----+---<<
                                |         |
                                +--- SAVE  -----+
                                |         |
                                +--- RESTORE ---+ +--- ALL ---+
                                |         |
                                +--- DEFAULT ---+
```

Description:

Save, and subsequently restore, the position and dimensions of MDI child windows in the SELCOPY Interactive application. Alternatively, restore the MDI child windows to a non-maximised state and then reposition/resize one or all of the windows to a default (optimised) configuration.

Users can configure the appearance of the SELCOPY Interactive child windows then use SDBWINX to save the windows' positions and sizes so that they may be resored in subsequent invocations of SELCOPY Interactive.

SDBWINX is ideal for use with smaller 3270 terminal sizes where all SELCOPY Interactive MDI child windows are automatically opened in a maximised state.

SDBWINX is called from the SDBPOPUP macro when the Window Layout menu item is selected.

If no parameters are specified, a pop-up window is opened allowing the user to select one of the valid options.

Parameters:

- SAVE** SAVE the window position and size information to a file with fileid "user.CBLI.WINX" (MVS) or "CBLI WINX" (CMS).
If the file does not already exist, the "Allocate Non-VSAM" dialog is opened to allocate it.
- RESTORE** RESTORE the window position from its saved entry in the WINX configuration file.
- DEFAULT** Position and size the window to a CBL defined optimal state.
- ALL** Perform the SAVE, RESTORE or DEFAULT operation on all child windows in the SELCOPY Interactive display.

Examples:

```
sdbwinx
sdbwinx default
sdbwinx save all
sdbwinx restore all
```

TRB Macro**Syntax:**

```
>>-- TRB -----+-----><
                |         |
                +--- trace_arg ---+
                |         |
                +--- OFF -----+
```

Description:

For use when editing and debugging a REXX macro, TRB inserts TRACE commands around a marked line or box block of text. The TRACE before the block will have parameters as specified on the TRB invocation, while, after the block, TRACE OFF is inserted.

The inserted TRACE commands are suffixed with the comment "/* Line inserted by TRB macro */" so that it may be removed by a later TRB OFF macro invocation.

Parameters:

- trace_arg** Valid REXX TRACE argument.
Default is 'r'.
- OFF** Remove all TRACE lines inserted by TRB.

Examples:

```
trb ?r
```

TSOC Macro

Syntax:

```
>>-- TSOC --- tso_cmd -----><
```

Description:

For use in the TSO (ISPF) environment only.

Executes the specified TSO command, traps any screen output from the command and displays it in an edit view.

Parameters:

tso_cmd	Valid TSO command.
---------	--------------------

Examples:

```
tsoc help alloc
tsoc lista
```

WINX Macro

Syntax:

```
>>-- WINX +- Restore --+
          |             |
          +-----+ +-----+
          +- Save -----+ +- username --+
          |             | |             |
          +- Delete ----+ +- * -----+
          |             | |             |
          +- Ini -----+ +- STD -----+
          |             | |             |
          +- Edit -----+
          |             |
```

Description:

WINX has the following uses:

1. Save the window position and size of the current edit view.
2. Restore the window position and size of the current edit view from a previously saved configuration.
3. Restore the frame window position and size to the InitialSize/InitialPos values set in the CBLiINI file.

WINX SAVE stores the edit view characteristics in a single line entry within the WINX configuration file. Each entry contains the position and size of both the edit view and the CBLe frame window at the time the WINX SAVE was executed, together with an assigned username.

The username is supplied by the user when SAVE WINX is issued, and is subsequently used to reference the required configuration entry on WINX RESTORE/DELETE commands. If no username is specified on WINX SAVE, then the user is prompted to either accept the default username (current file's fileid) or enter a different username.

An edit view may be restored to any of the saved edit view configurations simply by referencing the required entry's username on WINX RESTORE. If no username is specified on WINX RESTORE, then the username defaults to the current file's fileid.

Having identified the configuration entry to be used for RESTORE, if the saved CBLe frame window size does not exactly match that of the current CBLe frame window, then the position and size of the restored edit view are adjusted to reflect that of the saved edit view relative to its original CBLe frame window dimensions. e.g. If an edit view occupies the right half of the CBLe frame window display area when saved, then, after increasing the size of the CBLe frame window, the size and position values used on restore are adjusted so that the edit view still occupies the right half of the CBLe frame window display area.

An entry saved to the configuration file with a username that already exists, will replace the existing entry unless the size of the CBLe frame window is different to that saved in the existing entry. In this case, the new entry is simply added to the configuration file and so the username is no longer unique.

When referencing a non-unique username on WINX RESTORE/DELETE, the configuration entry used will be that containing a CBL e frame window size that matches the current CBL e frame window size. If the current CBL e frame size is not matched by any of the entries that match username, then the first entry matching username is used.

If the CBL supplied PROFILE macro is used as the default profile macro for CBL e, then some useful buttons are added to the CBL e application menu bar. These include **wS**, which executes WINX SAVE, and **wR**, which executes WINX RESTORE.

WINX is called from the SDBWINX macro.

Parameters:

Save	Save position/size information for the current edit view and frame window in the WINX configuration file. This file has fileid "user.CBLI.WINX" (MVS) or "CBLI WINX" (CMS). If the WINX file does not already exist, the "Allocate Non-VSAM" dialog is opened to allocate it.
Restore	Restore the current edit view to the saved position/size referenced by the specified username entry in the WINX configuration file.
Delete	Delete the named entry from the WINX configuration file.
Ini	Restore the window CBL e frame position/size using the CBLiINI variables InitialPos/InitialSize.
Edit	Edit the WINX configuration file.
username	Name used by SAVE/RESTORE/DELETE to reference the window configuration entry in the WINX configuration file. Default is the current fileid.
*	(asterisk) For RESTORE only, use the current fileid as the username. If this username entry does not exist, display a dialog box prompting the user to provide an alternative.
STD	For RESTORE only, restore the current edit view to its standard position/size as if it had just been opened.

Examples:

```
winx save
winx save macview1
winx restore macview1
```

Section 09: CBL CMX (Command) Files

CMX files are non-executable, plain text files containing groups of associated TSO, ISPF, CBLi and CBLc commands and comments that may be used to perform regular tasks and procedures that may ordinarily be issued from a command prompt.

The commands in these files may be executed using CBLc's CMDTEXT facility simply by placing the cursor on the command text and hitting PF4 (assigned to CMDTEXT by default.)

The advantage of using CMX files is that users can maintain them as single points of reference for job management so that they contain a reservoir of tried and tested commands. This can remove the task of having to remember command syntax and having to re-type a command every time it needs to be executed.

Note: Use of CMDTEXT is not restricted to files of type CMX. Commands may be stored, and subsequently executed, from any editable plain text file. (e.g. REXX procedures and SELCOPY, Assembler, C/C++, COBOL source files.)

CBLiDEMO

A starting point for using CBLi including features such as:

- The CBLc text editor.
- SELCOPY Interactive.
- CBLVCAT Interactive.
- List Windows.

CBLiINST

For sites that already have CBLi installed, this CMX file can assist the systems programmer with any subsequent CBLi installation. CBLiINST contains the commands that need to be executed to complete each step of the install process.

Section 10: SELCOPY Interactive

CBLe text edit support

All SELCOPY Interactive child windows, other than list and storage display windows, are CBLe text edit windows. (e.g. SYSIN, SYSPRINT and LOG windows.) This allows the user to edit data in these windows and to issue CBLe commands and macros such as ALL, LOCATE, CHANGE and SAVE.

Similarly, CBLe windows not usually associated with SELCOPY Interactive may also be opened to edit data. e.g. BROWSE the input data sets, etc.

MDI (Multiple Document Interface) Support

The SELCOPY Interactive window has become an MDI parent window in order to properly support CBLe text edit commands and macros.

Also, in previous versions of CBLi, SELCOPY Interactive storage display windows were child windows of the CBLi main window making navigation between related windows awkward. Since, windows within an MDI application may be navigated independantly of other applications, this is no longer a problem.

Note that PF9 is assigned to MDINEXT (place focus on next MDI window) by default.

Popup Menu and Window Configuration

REXX Edit macros SDBPOPUP and SDBWINX have been included in the CBLi 1.20 product bundle to provide additional functionality to the SELCOPY Interactive application. They should be made available in a common macro library and referenced in the user's MacroPath (CBLiINI.)

The SELCOPY Interactive Popup Menu is opened on execution of SDBPOPUP (assigned to PF4 by default) within an edit view window. SDBPOPUP is sensitive to the token and SELCOPY operation on which the cursor is positioned when it is executed and reflects this in the menu items.

The menu allows the user to easily set/unset a break point for the operation at the cursor position, open a new POS storage display window, manage existing and set new TRACKed expressions. The expression used on the TRACK function or to open a new POS storage display window, is a variation of the expression (token) at which the cursor is positioned.

Other items include execution of CMDTEXT (traditionally assigned to PF4), toggle between default actions for certain PFKeys used in both SELCOPY Interactive and CBLe (Debug or Edit mode) and Window Layout.

Window Layout invokes the SDBWINX macro to save and restore the size and positions of all the SELCOPY Interactive specific edit view windows. The edit views may be restored to a previously saved configuration or to their default dimensions and location.

SDBPOPUP and SDBWINX macros are documented in more detail in the CBLe Edit Macros section of this New Features document and in the macros themselves.

TRACE window

The trace information, previously displayed in the background of the parent window, is now displayed in a CBLe edit window entitled TRACE. Because of this, the trace output may be saved to a data set if required.

Messages

Message ID prefix included on each SELCOPY Interactive Message.

Section 11: CBLiNI Options

Support for the following SYSTEM or USER CBLi INI file options have been added:

```
[SYSTEM]
  UserINIFile=user.cbliini.file
```

For MVS environments, System.UserINIFile may only be specified in the System CBLiINI file to define the full DSN of the User CBLiINI file. This option will override CBLi's default method of identifying the User CBLiINI file.

The variable **&USER** or **%USER%** may be included in the DSN to denote the user's RACF userid. e.g

```
UserINIFile=SYSGEN.DEV.&USER.CBLI.INI
```

```
[ISPF]
  INITIALSTATE=ON|OFF
```

For TSO ISPF environments only, defines whether 3270 I/O is performed by ISPF (ON) or TSO (OFF). The CBLi CLI ISPF command may subsequently be used to toggle between the two environments.

Default is ON.

```
[CBLVCAT]
  DEFAULTCOMMAND=[<] (CBLi CLI command VCAT input)
```

Specifies the CBLVCAT Interactive input parameters to be used when the CBLVCAT Interactive window is opened with no parameters.

The syntax is the same as that specified on a CBLi CLI VCAT command. i.e. any valid CBLVCAT syntax or input via a control file e.g. "< cbl.ssc.ctl(report1)".

If the input parameters are prefixed by "<" (less than), the command is actioned immediately when the window is opened. Otherwise, the command is simply placed at the VCAT Command prompt for edit.

The default action, when no parameters are specified and no CBLVCAT.DEFAULTCOMMAND is set, is to place the following at the VCAT Command prompt:

```
REPORT VCAT DSN TYPE VOL2          !LISTCAT TYPE=U REF=your.master.catalog
```

This will generate a report, listing user catalogs.

```
[SELCOPY]
  INITIALPOS=row,col
  INITIALSIZE=rows,cols
```

Specifies the initial position and size that is used when the SELCOPY Interactive MDI parent window is opened.

The default position and size of the SELCOPY parent window is based on the dimensions of the 3270 terminal. On smaller terminals, the window is opened in a maximised state.

```
[EDIT]
  DEFPROFILE=macroname
```

Define the name of the macro to be used as the CBLe text edit default profile macro. This macro is executed every time a new file is loaded into the editor.

This may also be set using the CBLe CLI command SET DEFPROFILE.

The name of the default profile may be temporarily overridden using the EDIT command options NOPROF and PROFILE.

The default profile name is PROFILE.

Section 12: Define VSAM Dialogs

Dialog menu bar items have been updated with the following:

- DEFINE Start the VSAM object definition. (Foreground)
- JOB Creates and edits the IDCAMS DEFINE statement including job control ready for submission to batch (See CBLi command SUBMIT). (Background)
- AMS Creates and edits the IDCAMS DEFINE statement only.

Section 13: Operating System Window

The Operating System window menu bar items have been updated with the following:

Storage Drop down menu containing items that open further displays listing allocated and unallocated areas of Private and CSA storage.
Also SQA and LSQA storage. e.g.

```
-Unallocated storage                               +-x
View Back Forward FDB Edit Help

Type -Region- --Size-- Address- -Length-
PVT 00002000 00004000 00002000 00004000
PVT 00006000 008FA000 000F8000 00002000
PVT 00006000 008FA000 000FC000 00795000
PVT 00006000 008FA000 00896000 00005000
PVT 11600000 6EA00000 118FE000 00002000
PVT 11600000 6EA00000 11905000 00003000
PVT 11600000 6EA00000 1190A000 00001000
PVT 11600000 6EA00000 1194D000 00001000
PVT 11600000 6EA00000 11951000 00002000
Command>
Line 1 of 17 | Col 1 of 40 | Views 1 | sel
```

Programs Open the Loaded Programs window.

This is a list window providing information on dynamically loaded programs in the current address. e.g.

```
-Loaded Programs                               +-x
View Back Forward FDB Edit Help

--Name-- --EPA--- Address- ---Size--- SizeHex- -Use- AMode31 Auth Rent
CBLDLL   117D5000 117D5000    663552 000A2000    1 Y   N   Y
CBLI     11791000 11791000   147456 00024000    1 Y   N   Y
CBLNAME  00070D60 00070D60     672 000002A0    1 N   N   N
CBLNAME  00070AC0 00070AC0     672 000002A0    1 N   N   N
CBLNAME  00070820 00070820     672 000002A0    1 N   N   N
CBLXREXX 11626C00 11626C00     512 00000200    1 Y   N   Y
DBXXREXX 11626E00 11626E00     512 00000200    1 Y   N   Y
IEANTCR  1160C050 1160C050     72 00000048    2 Y   N   Y
IEANTDL  116687E8 116687E8     72 00000048    2 Y   N   Y
IEANTRT  1160C008 1160C008     72 00000048    2 Y   N   Y
IGGCSI00 11668290 11668290   1368 00000558    1 Y   N   Y
IGG019BA 00CB7120 00000000     0 00000000    0 N   N   Y
IGG019BB 00E058C0 00000000     0 00000000    0 N   N   Y
ISPLINK  0000A5A8 0000A5A8     440 000001B8   11 N   N   Y
V210     000AF000 000AF000   147456 00024000    1 Y   Y   N

Command>
Line 1 of 15 | Col 1 of 108 | Views 1 | select *
```

Help Help item for the Operating System window.

Section 14: List and CBLVCAT window Prefix Commands

The following table details new and re-assigned functions for LIST and CBLVCAT window prefix area commands in CBLi 1.20.

Description	New Prefix Command	Original Command
Kill (Erase without prompting for verification) the the LIST or CBLVCAT data set or member entry.	K	n/a
(CBLVCAT only) Opens a LIST Dataset window for the entry in the CBLVCAT report. In CBLi 1.20, L is not supported for LIST windows.	L	n/a
Opens an Execute CBLVCAT window and issue a LISTVCAT/LISTVTOC operation for the entry.	V	L
Open the volume statistics window for the volume containing the entry.	?	V

For CBLVCAT windows, the "V" prefix command performs LISTVCAT/VTOC operations based on the contents of the entry fields, as follow:

1. If the entry contains the TYPE field "USERCAT", then a new report is generated for the entire contents of that catalog. Otherwise, only list the catalog entries that match the fileid.
2. If the report entry also contains a "VOLn=volser" field, then generate a LISTVTOC report for entries that match the fileid in the volume's VTOC.

The CBLVCAT window default action on <Enter> also depends on the contents of the report entry, as follows:

1. If the entry contains the TYPE field "USERCAT" or "ALIAS OF", then prefix command "V" is default.
2. If the entry contains the TYPE field "PDS" or "PDSE", then prefix command "M" (Member List) is default.
3. If the entry contains a fileid, then prefix command "E" (Edit) is default.

Glossary

3270 Emulator

Third party software that emulates Mainframe 3270 hardware terminals on PC and UNIX based platforms.

CLI

A Command Line Interface is a text based method by which users can execute functions supported by the application.

CBLe

A powerful text editor that runs as an MDI application under CBLi. CBLe supports its own CLI and has been developed based on specifications found in Mansfield Software's KEDIT for Windows.

CBLi

The Interactive environment developed by CBL and supplied as part of SELCOPY and CBLVCAT licensable software products.

CBLiINI

File containing configuration options for CBLi. The SYSTEM CBLiINI file is processed on startup of CBLi and contains options that apply to all users. The USER CBLiINI file contains options specific to each user that may, where appropriate, override options set in the SYSTEM CBLiINI file.

CBLiVTAM

Name of the multi-user version of the CBLi application that executes under VTAM.

CBLVCAT

CBL licensable product that supports VSAM file tuning and VTOC, ICF/VSAM catalog and VSE LABEL reporting. Executes as a batch facility or interactively as a CBLi application.

Edit View

A CBLe MDI document window that contains a display of edited data. If the same file is displayed in multiple windows, then the user has multiple edit views of the file. Each edit view can have a different current line, ARBCHAR setting, ZONE columns, etc.

List Window

A CBLi window containing rows of associated information. List windows support point-and-shoot column sorting; select, sort and filter CLI commands; and prefix area commands.

MDI

Multiple Document Interface is a Microsoft specification for PC applications that enable the user to work with multiple documents at the same time. Each document is displayed in a separate child window within the client area of the application's main (frame) window. Typical MDI applications on PCs include word-processing and spread sheet applications.

MDI Client Area window

The MDI client area window is the display area within an MDI application's frame window. The MDI client area serves as the background for MDI child windows.

MDI Child/Document Window

An MDI child or document window is opened in an application's client area window each time a document is opened. Each child window has a sizing border, title bar, window menu, minimise, maximise, restore and close buttons. A child window is clipped so that it is confined to the client window and cannot appear outside it.

When a child window is maximized, its client area completely fills the MDI client area window. In addition, the system automatically hides the child window's title bar, and adds the child window's window menu icon and Restore button to the MDI application's menu bar.

MDI Frame Window

An MDI frame window may be considered the main window of an MDI application. It is the parent window of the MDI client area window in which MDI child windows are opened. It has a sizing border, title bar, window menu, minimise, maximise restore and close buttons.

Ring

The set of all **files** being edited within CBLe. It is not the set of all windows opened. e.g. The contents of one file may be displayed in more than one edit view (window.)

SELCOPY Interactive

An Intergrated Development Environment for SELCOPY that runs as an MDI application under CBLi.

Storage Display Window

A CBLi window containing hexadecimal and character display of areas of storage.
