Compute (Bridgend) Ltd

CBLi 1.50 New Features
for IBM Mainframe z/OS, VSE & VM/CMS Systems

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

Contents

CBLi 1.50 New Features

Documentation Notes

Information in this New Feature List reflects differences between CBLi 1.40 and CBLi 1.50, the interactive components of SELCOPY and CBLVCAT.

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

The **CBL Products Installation Guide** and **New Features** documents are available in Adobe Acrobat PDF format at CBL web page http://www.cbl.com/selcdoc.html.

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.

The following generic terms are used throughout this document to indicate all available versions and releases of IBM mainframe operating systems:

MVS - z/OS, OS/390, MVS/ESA, MVS/XA, MVS/SP, OS.

VSE - z/VSE, VSE/ESA, VSE/SP, DOS.

CMS - z/VM, VM/ESA, VM/XA, VM/SP.

All - All IBM mainframe operating systems and releases.

Section 01: Important Changes

All: List Catalog/Dataset - Trailing *.** Wildcards

Generating a list of data sets (via the Catalog List or Dataset List windows) requires provision of a data set file mask.

A change introduced by CBLi 1.40 zaps (Ref: I140z02 and I140z09) alters the implementation of the trailing wildcard string on a file mask specified via the CBLi CLI commands, LISTCAT or LISTDATASET, or via the Entry field of the Catalog List or Dataset List windows.

In previous releases, the wildcard string "*.**" is appended to the file mask only when the file mask already contains no "*" (asterisk) or "**" (double asterisk) wildcard characters.

In order to perform equivalent operation to MVS ISPF 3.4, this basic specification has been refined so that the wild card string of "*.**" is always appended to the file mask regardless of the presence of imbedded wildcards. e.g.

```
LC  DEV.OEM.CBL202        becomes:   LC  DEV.OEM.CBL202*.**
LC  SYS1.ZOS              becomes:   LC  SYS1.ZOS*.**
```

Exceptions to this are as follow:

1. If the supplied filemask is a single qualifier or the last qualifier is length 8, a wildcard string of ".**" (not "*.**") is appended. e.g.

   ```
   LC  DEV                   becomes:   LC  DEV.**
   LC  DEV.OEM.TRSPAN00      becomes:   LC  DEV.OEM.TRSPAN00.**
   LC  DEV.*.TRSPAN00        becomes:   LC  DEV.*.TRSPAN00.**
   ```

2. If "*" (asterisk) is the **last** character of the file mask, then no wild card string is appended. e.g.

   ```
   LC  DEV.OEM.CBL202.**     becomes:   LC  DEV.OEM.CBL202.**
   LC  DEV.OEM.CBL202.*      becomes:   LC  DEV.OEM.CBL202.*
   LC  DEV.OEM.*             becomes:   LC  DEV.OEM.*
   ```

3. If "." (dot/period) is the **last** character of the file mask, then the preceding qualifier is the last qualifier of the file mask. The trailing "." is stripped and no wild card string is appended. e.g.

   ```
   LC  DEV.OEM.CBL202.       becomes:   LC  DEV.OEM.CBL202
   LC  DEV.OEM.*.EXE.        becomes:   LC  DEV.OEM.*.EXE
   ```

In addition to this, if the supplied filemask has asterisk as the first qualifier, a warning is issued that ALL catalogs will be searched. (For large systems, this may take a great deal of time and resource.) e.g.

```
LC  *.OEM.CBL.CBLI
```

Section 02: New Facilities

MVS: Structured Data Environment

Many production data sets are created and maintained so that each record conforms to a template, often defined by a COBOL or PL/1 copybook. In CBLi, these types of data sets are considered to be structured.

CBLi 1.50 includes a Structured Data Environment (SDE) which includes a structured file editor with support for the following:

1. Generate CBLi SDE format structure objects (structures) from an existing COBOL copybook or using CBLi SDE native syntax. Each SDE structure may contain any number of individual record type definitions used to map different records within structured data sets, where each record type definition consists of a number of concatenated fields with defined position, length and data type.

2. In-storage edit or browse of a structured data set using an associated SDE structure.
   The entire data set is loaded into storage before it is displayed in an SDE window view. Records that fail to satisfy the criteria set by any of the record types in the structure, are assigned a default record type for the session.

3. Restrict display to all records of a particular record type or concurrently display multiple record types within the same SDE window view. Each record group of the same record type is preceded by its record headers.

4. Display records in multiple record view or single record (zoomed) view.
   By default, structured records will be formatted into columns of field data. However, records may also be displayed in unformatted character or hexadecimal in either single or multiple record view.

5. Select a subset of field data columns and the order in which they are to be displayed for each record type.

6. FIND or EXCLUDE occurrences of character strings or numeric values within all or selected fields belonging to records of a particular record type. Alternatively, LOCATE or filter (WHERE) only those records that satisfy a supplied *where_clause* expression.

7. INSERT, REPLACE, overtype field data and SAVE changes to the data set.

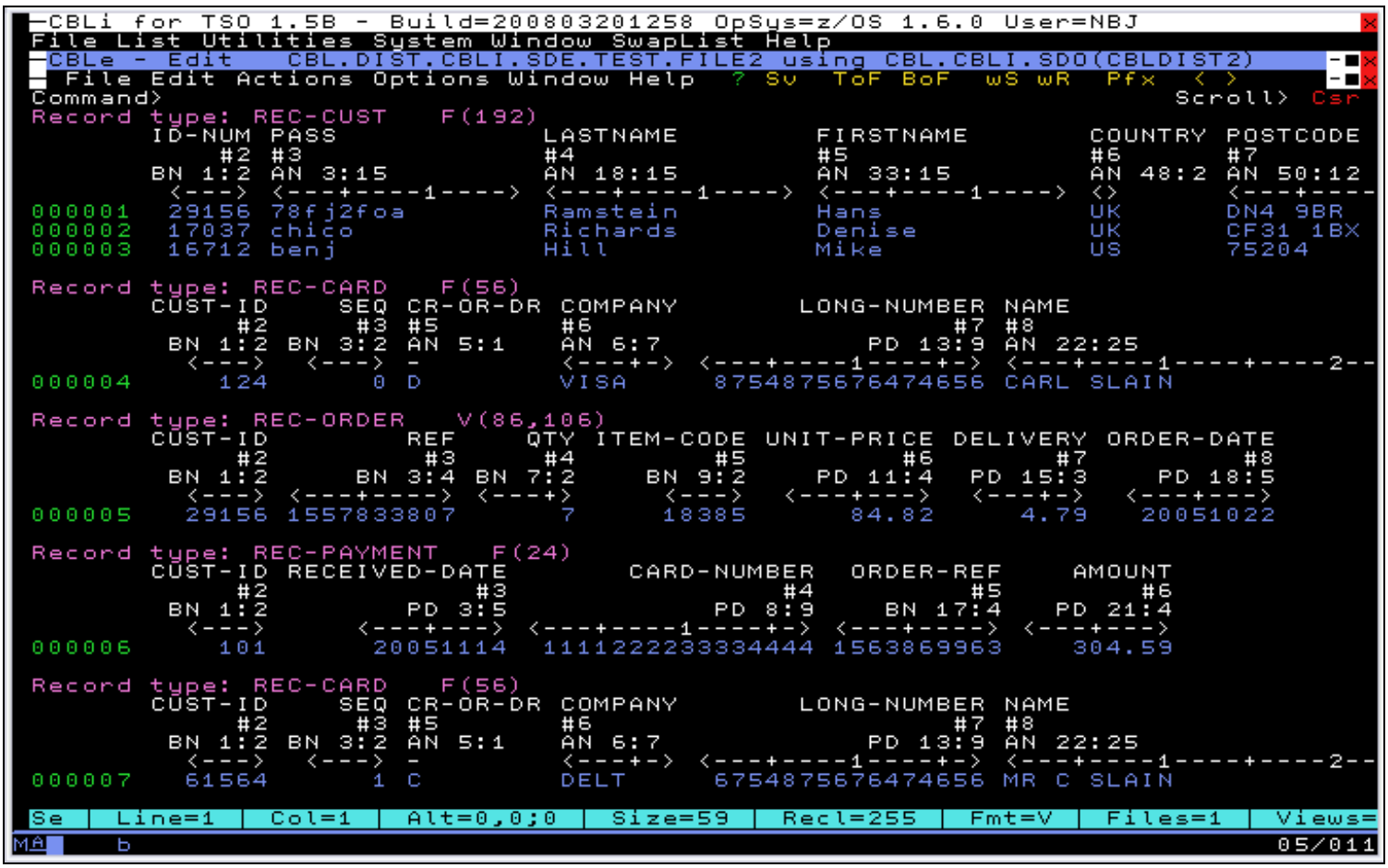


Figure 1. SDE Window View.

SDE runs as a sub-component of the CBLe Text Editor so that SDE window views displaying structured file data are opened as child windows of a CBLe (or SELCOPY Interactive) MDI Frame window.

Although the implementation of SDE in CBLi 1.50 is basic (e.g. significant features such as CHANGE, UNDO and REDO are missing, as are alternative editing techniques more suited to large files or VSAM data sets), it is the prime focus for CBLi development. Updates to the CBLi software will be released frequently containing mostly (or only) new functionality and improvements to the SDE environment.

Please refer to the CBLi Structured Data Environment manual for details of functionality supported in CBLi 1.50.

MVS: Customised List Data Report Windows

The CBLi SDE facility also allows users to generate their own customised CBLi list reports supporting all the standard select, sort and filter list functionality.

The **CREATE LIST** function generates the named list object (containing list title, headers and field data) from a structured data set and the CBLi structure object. **DISPLAY LIST** displays the list object in a CBLi list window.

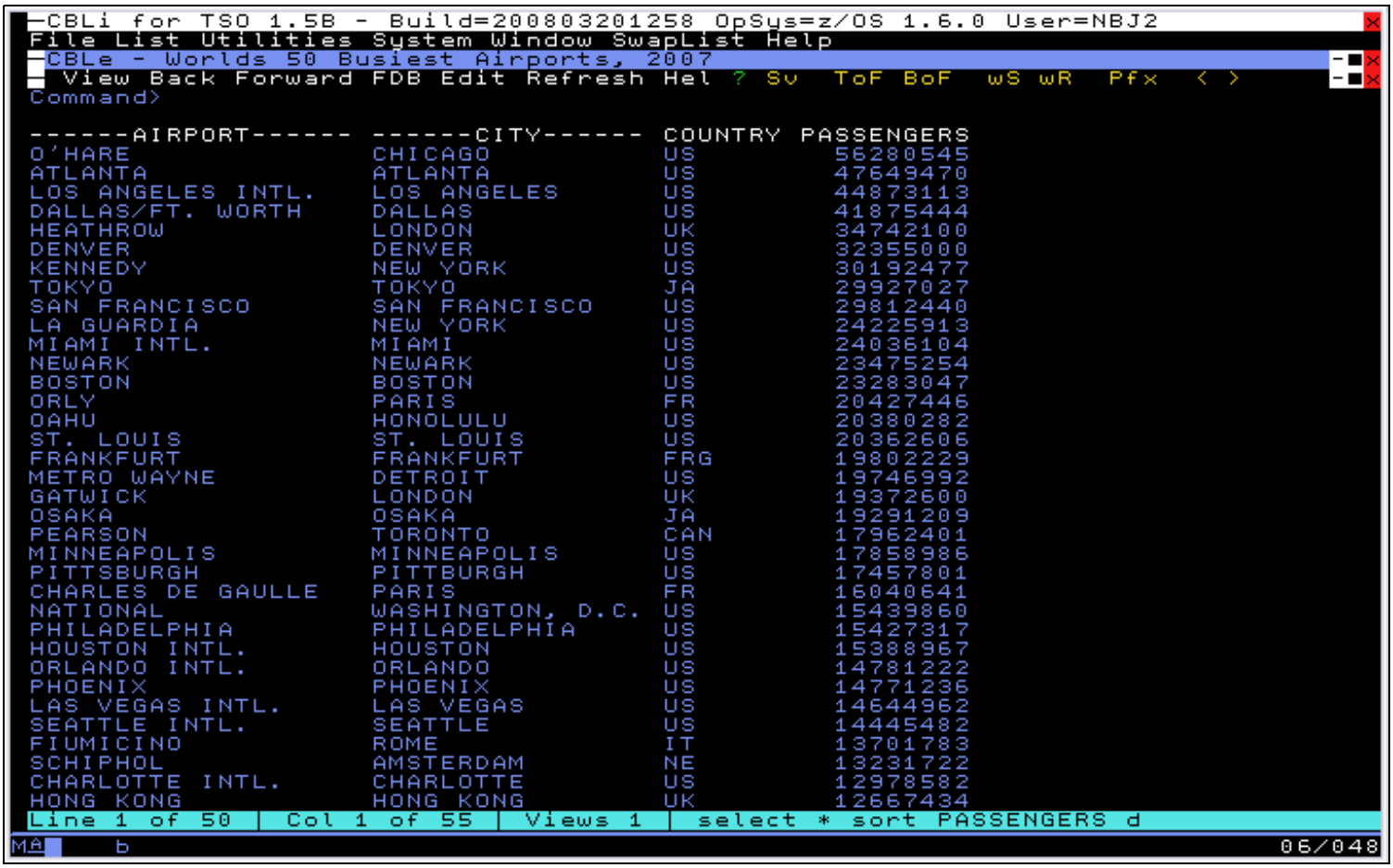


Figure 2. Customised List Report.

VSE: Trusted User Specification

CBLi configuration at VSE installations that have Security Manager software activated, should include the option, System.VSESMLogon=Yes, in the SYSTEM.CBLIINI library member. This ensures that users logging on to CBLi use their usual userid and password and actions performed by CBLi on behalf of the user are subject to security manager authorisation.

In previous releases of CBLi where no VSE Security Management software is in effect, users log on with an arbitrary userid and password and access to VSE resources is unrestricted. The exception to this is the POWER Command output window whereby the contents of an entry in the POWER queue can only be displayed if it is password protected and the password known.

In order to apply a basic level of security on systems without security management software, the concept of trusted users has been introduced in CBLi 1.50 to allow VSE system administrators to restrict access to CBLi. As before, once a trusted user has logged on to CBLi, access to VSE resources is unrestricted with the addition that no restriction is placed on the display of POWER queue entry contents. (Successful display of entries that are password protected would still require specification of the correct password.)

VSE Trusted users are defined by including the following options in the (System) section of the SYSTEM.CBLIINI member:

1. **TrustedUser=YES**
   Activates trusted user logon feature.

2. **Trust-*userid*=*password***
   The Trust-*userid* option assigns a *userid* and *password* for a single trusted user. An instance of this option should be incuded for each trusted userid. Successful logon to CBLi may only be achieved using one of these userids.

Having updated the SYSTEM.CBLIINI member, the CBLiVTAM applid must be restarted to obey the latest changes.

When a user starts CBLi, the CBLe text editor component of CBLi is also started by default. It is during the open of the CBLe application window that the user is prompted to to supply a trusted userid and password.

Note that the user will be prompted for a userid and password any time a new CBLe MDI session is started under CBLi. However, since the CBLe text editor is an integral part of CBLi operation, it is always recommended that the CBLe MDI session is never closed during the CBLi session.

MVS: Generate ALIAS Library Entries

Functionality has been introduced to easily create an ALIAS for a member of PDS/PDSE Library or Load Library. This takes the form of the CBLi CLI command, ALIAS, and the Library List Window prefix command, "A".

Library List Window - Prefix Command "A"

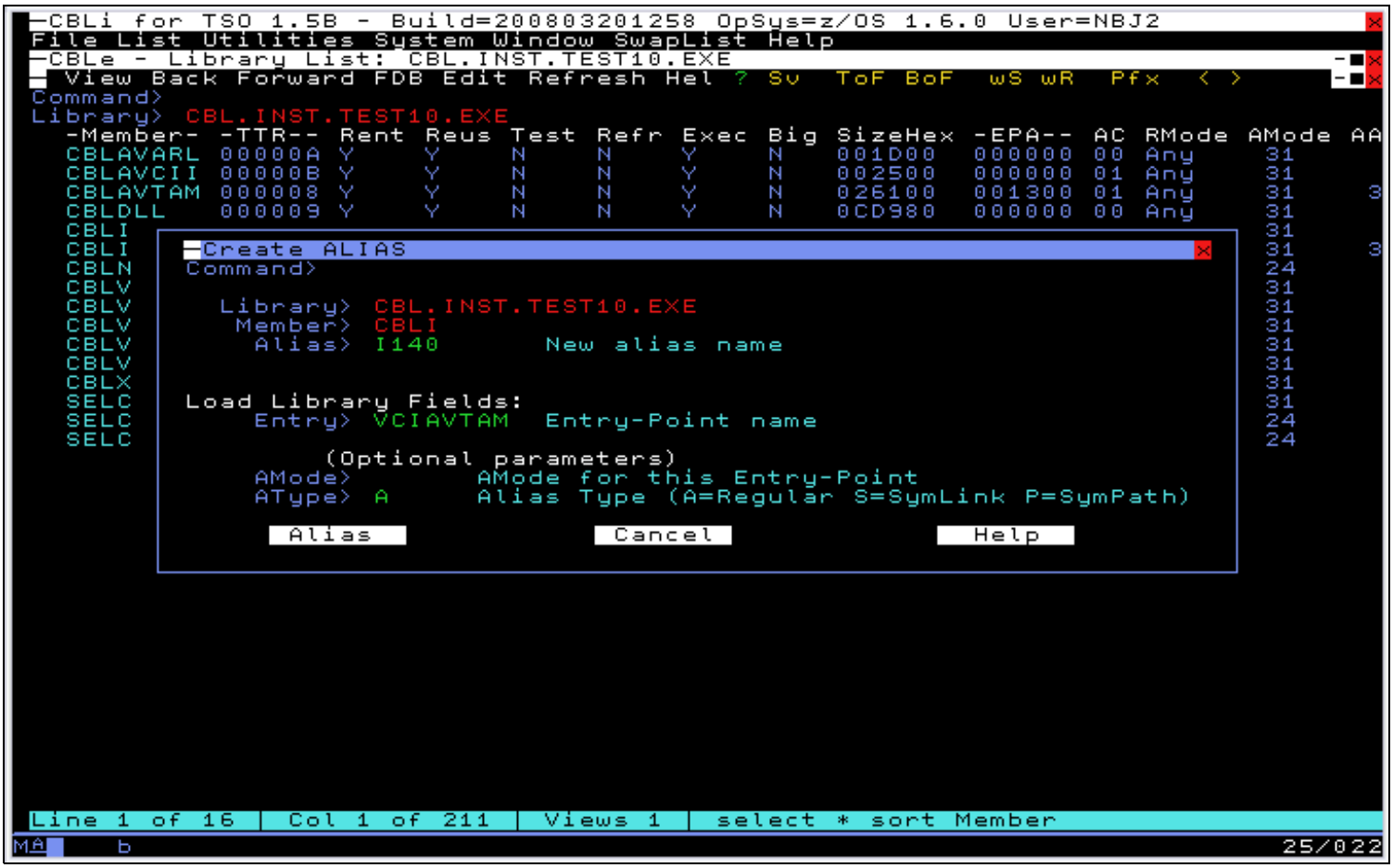
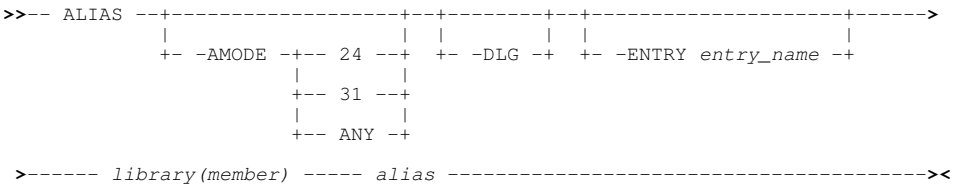


Figure 3. ALIAS Dialog Window.

CBLi ALIAS Command

Syntax:



Description:

Use the ALIAS command to create a new PDS/PDSE library member alias, or open "Create ALIAS" dialog window.

Note that aliases for load-library members are created using the binder to relink the module in being aliased. This will result in an update to the module's **TTR**.

Parameters:

library(member)
The PDS/PDSE library member to be aliased.

alias
The alias name to be added.

-AMODE
For a load-library member, used to specify the Addressing Mode for the new aliased entry-point. Valid arguments are 24, 31 and ANY.

-DLG
Invoke the "Create Alias" dialog window.

-ENTRY *entry_name*
For a load-library member, used to specify the symbolic name of the entry-point address to be used.

All: CBLVCAT Raw Window

Where **CBLVCAT** is licensed, support has been introduced for a new List window type: CBLVCAT Raw.

The CBLVCAT Raw window is opened using the CBLi CLI command, LVR, and is equivalent to the Raw Data Window generated as a by product of an interactive CBLVCAT execution. The window contains all report field information obtained during the execution of the supplied CBLVCAT LISTVCAT/LISTVTOC operation.

CBLVCAT Raw Window

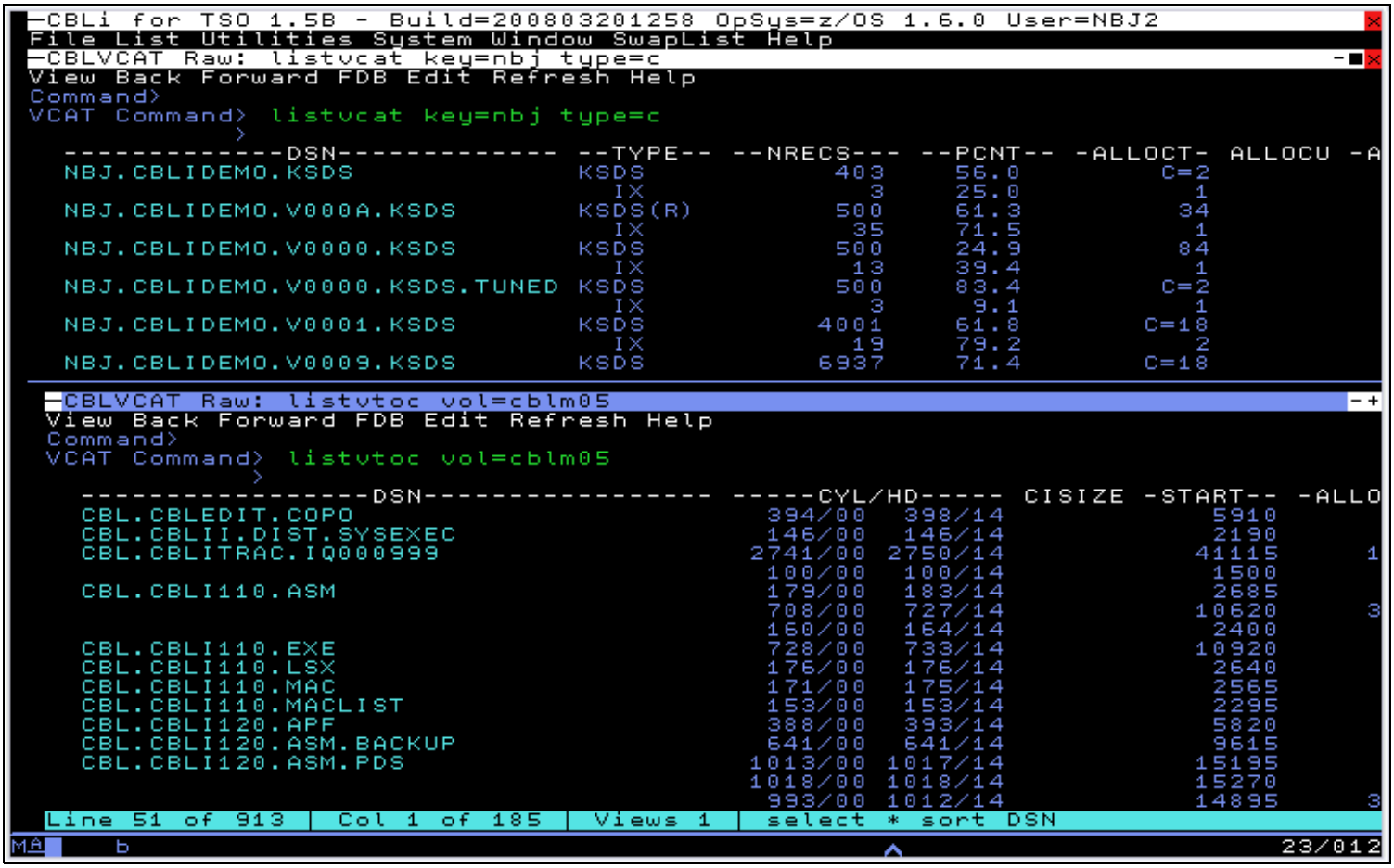


Figure 4. CBLVCAT Raw Windows.

Prefix Line Commands

The following prefix line commands are available in the CBLVCAT Raw window:

Command	Description
(blunk)	(Hit <Enter> with the cursor on a particular entry line). Prefix line command "M" if entry is a PDS/PDSE library, prefix line command "E" otherwise.
B	Open the CBLe text editor to edit this entry in read only mode.
D	Delete the entry. User will be prompted to verify the deletion.
E	Open the CBLe text editor to edit this entry. (Default for non-PDS/PDSE entries)
F	Open the File Search window for the entry.
I	Open an IDCAMS Command window and issue an IDCAMS LISTCAT for the entry.
K	Delete (Kill) the entry without prompting for verification.
M	If the entry is a PDS/PDSE, open a Library List window. (Default for PDS/PDSE entries)
Q	List dataset enqueues (major name SYSDSN) for this entry.
R	Rename the entry.
T	Open an Execute CBLVCAT window and issue a LISTVCAT TUNE DEFINE operation for the entry.
V	Open an Execute CBLVCAT window and issue a LISTVCAT operation for the entry.

CBLi LVR Command

Syntax:

```
>>-- LVR --+----------------------+-------------------------------------------><
           |                      |
           +--- cblvcat_syntax ---+
```

Description:

LVR opens the CBLVCAT Raw window and optionally executes CBLVCAT control statements.

Parameters:

cblvcat_syntax
Valid CBLVCAT syntax to be executed when the CBLVCAT Raw window is opened.
This parameter is placed in the "VCAT command line>" field of the CBLVCAT Raw window.

All: CBLi CLI CBLICANCEL

Syntax:

```
>>-- CBLICANcel ------------------------------------------------------------><
```

Description:

Exit and close the CBLi session without opening the quit session confirmation pop-up window.

All: CBLi CLI HOME

Syntax:

```
>>-- HOme -----------------------------------------------------------------><
```

Description:

Edit the user's personal command centre (CMX) file. A new CBLe text edit session is opened if one is not already open.

All: CBLe CLI STEMINSERT

Syntax:

```
>>-- STEMInsert --- rexx_stemvar ------------------------------------><
```

Description:

For use in CBLe REXX macros, STEMINSERT is a record mass insert command based on a REXX compound (stem) variable.

STEMINSERT determines the number of lines to insert from the *rexx_stemvar*.0 value and inserts new lines with text obtained from the value *rexx_stemvar*.***n_line***, where ***n_line*** is the insert line index (***n_line***=1,2,3,...,*rexx_stemvar*.0). The new lines are inserted following the focus line.
STEMINSERT is a fast method of insert and should be used in place of the following REXX syntax:

```
do  i =  1  to  linetext.0
  'insert' linetext.i
end
```

Replace this with:

```
'steminsert'  linetext
```

Parameters:

rexx_stemvar
The stem portion of an assigned REXX compound variable.

All: CBLe CLI SDATA

Syntax:

```
>>-- SData -- sde_command ------------------------------------------------><
```

Description:

Direct a command to the CBLi Structured Data Environment (SDE).

The SDATA command allows SDE commands to be issued from a CBLe text edit window, typically using PF4 to point-and-shoot at commands stored in a command centre (CMX) file, such as your HOME file.

Parameters:

sde_command
: Any SDE command. (See the Structured Data Environment Manual)

Examples:

```
<sdata create structure   CBL.CBLI.STRUCT(COMPSTR)    \
      from cobol CBL.COPYBOOK.COBOL(COMPDEF)
```
: Issue the SDE CREATE STRUCTURE command.

```
<sd edit   CBL.SDE.EMP   using CBL.CBLI.STRUCT(COMPSTR)
```
: Issue the SDE EDIT command to perform structured editing using the previously generated SDE structure.

```
<sd select Key,InvNumb,DeliveryDate from Orders in CBL.CBLI.STRUCT(COMPSTR)
```
: Issue the SDE SELECT command restrict and order the columns in the focus SDE edit view.

All: CBLe CLI SET/QUERY/EXTRACT FIDCHANGED

Syntax:

```
>>-+-----------+-- FIDCHanged ----+-- ON ----+------------------------------><
                                  |          |
                                  +-- OFF ---+

>>--- Query ------ FIDCHanged ------------------------------------------------><

>>--- EXTract --- /FIDCHanged/ ------------------------------------------------><
```

Description:

This option controls the internal CBLe flag indicating that the fileid of the current file has been changed by the user. (i.e. via the SET options DSN, FMODE, FNAME, MBR, FPATH, FTYPE or FILEID.)

If the FIDCHANGED flag is on, then CBLe QUIT command will prompt the user to save the file before the window is closed.

The SET FIDCHANGED functionality allows the user to assign a temporary DSN to a data being edited, thus releasing the ENQ on the original edited DSN, perform operations on that data set and subsequently restore the DSN then quit without the prompt to save. This technique is employed in the distributed ERA and RENAME CBLe REXX macros.

SET FIDCHANGED takes effect at the File level.

SET Value:

`ON|OFF`
: FIDCHANGED flag is set ON or OFF.

QUERY Response:

The current setting of the FIDCHANGED flag, **ON** or **OFF**.

EXTRACT Rexx variables:

`fidchanged.0`	1
`fidchanged.1`	The current setting of the FIDCHANGED flag, **ON** or **OFF**.

All: CBLe CLI QUERY/EXTRACT LSCREEN

Syntax:

```
>>--- Query ------ LScreen ------------------------------------------------><

>>--- EXTract --- /LScreen/ ----------------------------------------------><
```

Description:

QUERY/EXTRACT LSCREEN provides size and location information about the logical screen (current MDI child window and MDI client area).

These are the values that would be supplied on the CBLe SET options, WINSIZE and WINPOS, in order to achieve the size and location display characteristics of the focus edit view.

QUERY Response:

The LSCREEN values of the focus MDI child window and MDI client area within the parent window.

1. MDI child window depth (number of rows).
2. MDI child window width (number of columns).
3. MDI child window vertical position within the client area (row number).
4. MDI child window horizontal position within the client area (column number).
5. MDI client area depth (number of rows).
6. MDI client area width (number of columns).

EXTRACT Rexx variables:

`lscreen.0`	6
`lscreen.1`	MDI child window depth (number of rows).
`lscreen.2`	MDI child window width (number of columns).
`lscreen.3`	MDI child window vertical position within the client area (row number).
`lscreen.4`	MDI child window horizontal position within the client area (column number).
`lscreen.5`	MDI client area depth (number of rows).
`lscreen.6`	MDI client area width (number of columns).

Section 03: Other Changes

MVS: DB2 Dynamic SQL SSN, LIMIT, PLAN, COMMIT & EXEC

Please note that the SELCOPY DB2 interface (included with SELCOPY at no extra charge) must be configured for use with CBLi DB2 dynamic SQL features.

Enhancements have been made to the DB2 Dynamic SQL window and CBLi CLi command, SQL, in order to improve its flexibility and ease of use.

Users are now able to:

1. Override the default DB2 Plan name (defined in the CBLNAME module) via the SQL line command.
2. Override the default DB2 Subsystem name (defined in the CBLNAME module) via the SQL line command.
3. Specify a limit on the number of selected rows for display.
4. Specify whether a COMMIT is automatically issued following each transaction.

To support variable default values, the following CBLiINI options have been introduced:

DB2.SelectLimit=*n_rows*
The default value for the -LIMIT parameter of the CBLi SQL command and "Select Limit>" field of the Dynamic SQL window.

DB2.Exec=Immediate|Delay
The default action for the -EXEC parameter of the CBLi SQL command.

In addition to this, a new menu item, "Log" has been added to the menu bar of the Dynamic SQL window. This opens a new window, the SQL log, which contains diagnostic information (return codes, etc.) for all transactions issued in the current Dynamic SQL window session.

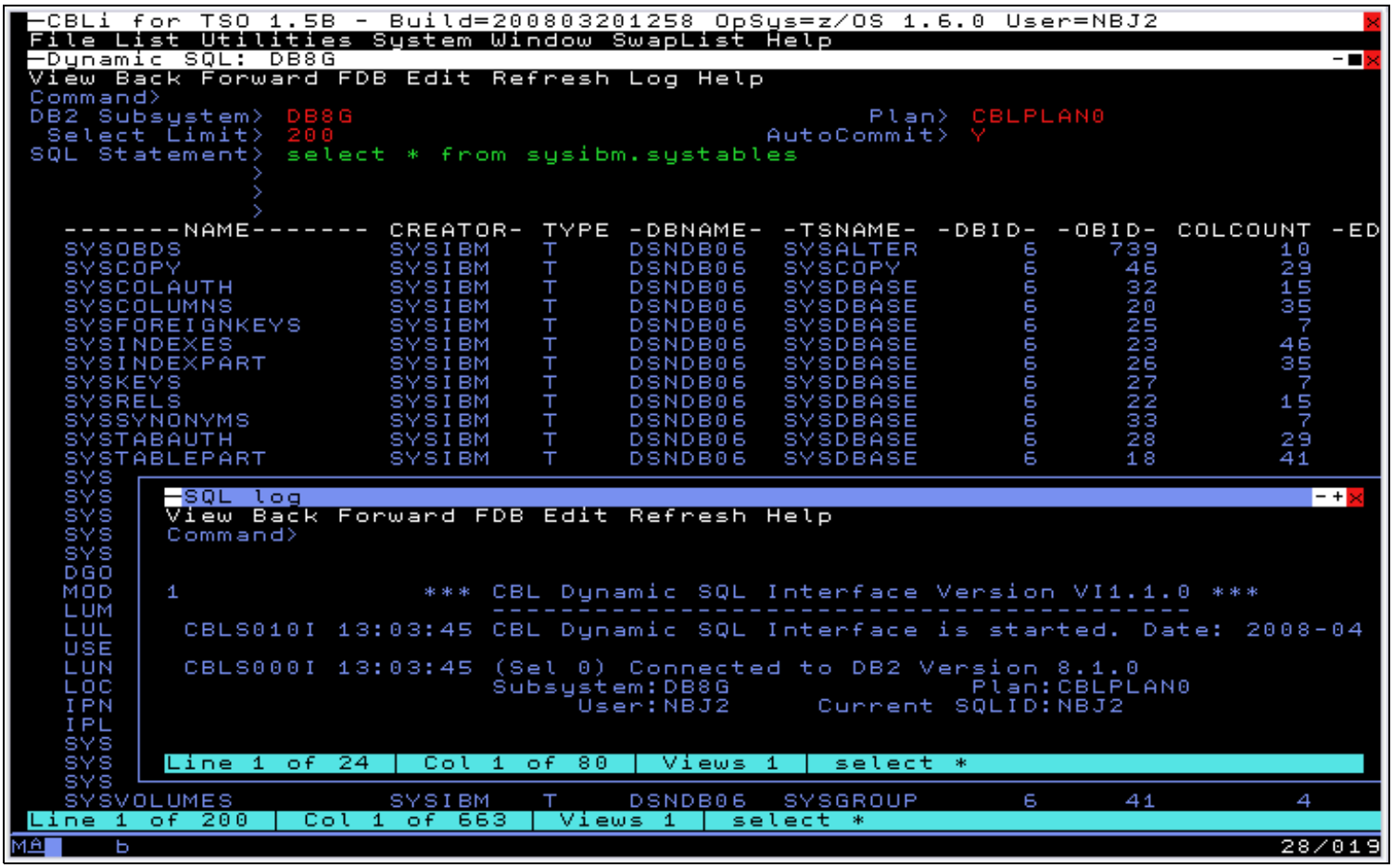


Figure 5. Dynamic SQL Window.

CBLi SQL Command

Syntax:

```
>>-- SQL --+--------------------+--+--------------------+--+------------------+->
           |                    |  |                    |  |                  |
           +- -SSN=ssn_name ---+  +- -PLAN=plan_name -+  +- -LIMit=n_rows -+

           +- -EXec=Immediate -+  +- -COmmit=Yes -----+
           |                   |  |                   |
 >---------------------------+------------------------+------------------+-><
           |                   |  |                   |  |                |
           +- -EXec=Delay -----+  +- -COmmit=No ------+  +- sql_syntax ----+
```

Description:

Use the SQL command to open the Dynamic SQL window.

The Dynamic SQL window may also be opened via the File menu of the CBLi main window menu bar.

A connection is made to the DB2 subsystem using the user's userid.

Parameters:

-SSN=*ssn_name*
: The DB2 sub-system to be the target of the CONNECT.
This parameter is placed in the "DB2 Subsystem>" field of the Dynamic SQL window.
Default is that defined by the CBLiINI option, DB2.SSN, otherwise the sub-system name specified in the DB2SubSys field of the CBLNAME load module is used.

-PLAN=*plan_name*
: The SELCOPY DB2 plan name which has been bound to the DB2 sub-system.
This parameter is placed in the "Plan>" field of the Dynamic SQL window.
Default is that defined by the CBLiINI option, DB2.Plan, otherwise the plan name specified in the DB2Plan field of the CBLNAME load module is used.

-LIMIT=*n_rows*
: Limit the number of rows to be displayed in the Dynamic SQL window following a SELECT transaction. Once the limit threshold has been reached, a pop-up message window is displayed and no further attempt is made to retrieve selected rows of data.
The *n_rows* value is placed in the "Select Limit>" field of the Dynamic SQL window.
The default limit is that defined by the CBLiINI option, DB2.SelectLimit, otherwise no limit is implied.

-EXEC=IMMEDIATE|DELAY
: Determine whether the SQL command is to be executed immediately when the Dynamic SQL window is opened or simply placed on the SQL Statement command line.
The default is IMMEDIATE.

-COMMIT=YES|NO
: Determine whether a COMMIT is to be automatically issued following every transaction (AutoCommit). If COMMIT=NO, then the user should issue COMMIT manually to commit any changes made to the data. A commit is executed automatically when the Dynamic SQL window is closed, regardless of the AutoCommit field setting.
The commit value is reflected in the "AutoCommit>" field of the Dynamic SQL window.
The default is YES.

sql_syntax
: Valid SQL syntax to be executed when the Dynamic SQL widow is opened.
The *sql_syntax* string is placed in the first SQL Statement line field of the Dynamic SQL window.

All: SELCOPY Interactive Loop Break-in

SELCOPY Interactive is particularly useful for the development of SELCOPY job streams enabling the user to trace the actions of each control statement and interactively debug the SELCOPY job.

Part of the debugging process may involve identification of conditions or sequence of statements that cause the job stream to loop. However, the user may not know that the loop condition exists until SELCOPY processing has been restarted without a break point and, since SELCOPY is executing in the foreground, the terminal is unresponsive.

For this reason, the SELCOPY default break-in facility has been introduced, allowing the user to pre-define a default number of times that any control statement within the SELCOPY job stream may be executed before a virtual break point is encountered and processing paused.

When the break-in threshold has been reached, a pop-up message window is opened and control is passed back to the user to continue debug investigation. This means that there is no need to forcibly end the CBLi session and restart the SELCOPY debug process.

The default break-in value is defined via the following CBLiINI option:

SELCOPY.LoopBreakIn=*n_times*
The Loop break-in counter for SELCOPY Interactive.
The default *n_times* value is 1000000.

Note that a loop break-in may occur even though a loop is not infinite. For example, a SELCOPY READ operation inputs one record at a time and so the logic flow is such that processing returns to the start of the control statements until end-of-file of the prime input file is encountered. Therefore, SELCOPY main control card execution is an implied loop. If the number of records read from the prime input exceeds the *n_times* value, then the loop break-in will occur even though processing loop is not infinite.

All: CBLVCAT TUNE Prefix Command for CBLVCAT & LIST Windows

Where **CBLVCAT** is licensed, the prefix command "T" has been introduced to perform a CBLVCAT LISTVCAT with TUNE and DEFINE parameters. The prefix is supported in the following List type windows:

1. Catalog List
2. CBLVCAT Raw List
3. Dataset List
4. Execute CBLVCAT Window

The "T" prefix command executes CBLVCAT to generate and display the following:

1. A CBLVCAT LISTVCAT report for the selected data set displayed in an Execute CBLVCAT window.
2. An AMS IDCAMS DEFINE job, which includes any new CBLVCAT TUNE recommended DEFINE parameters, is edited in a CBLe text edit view.

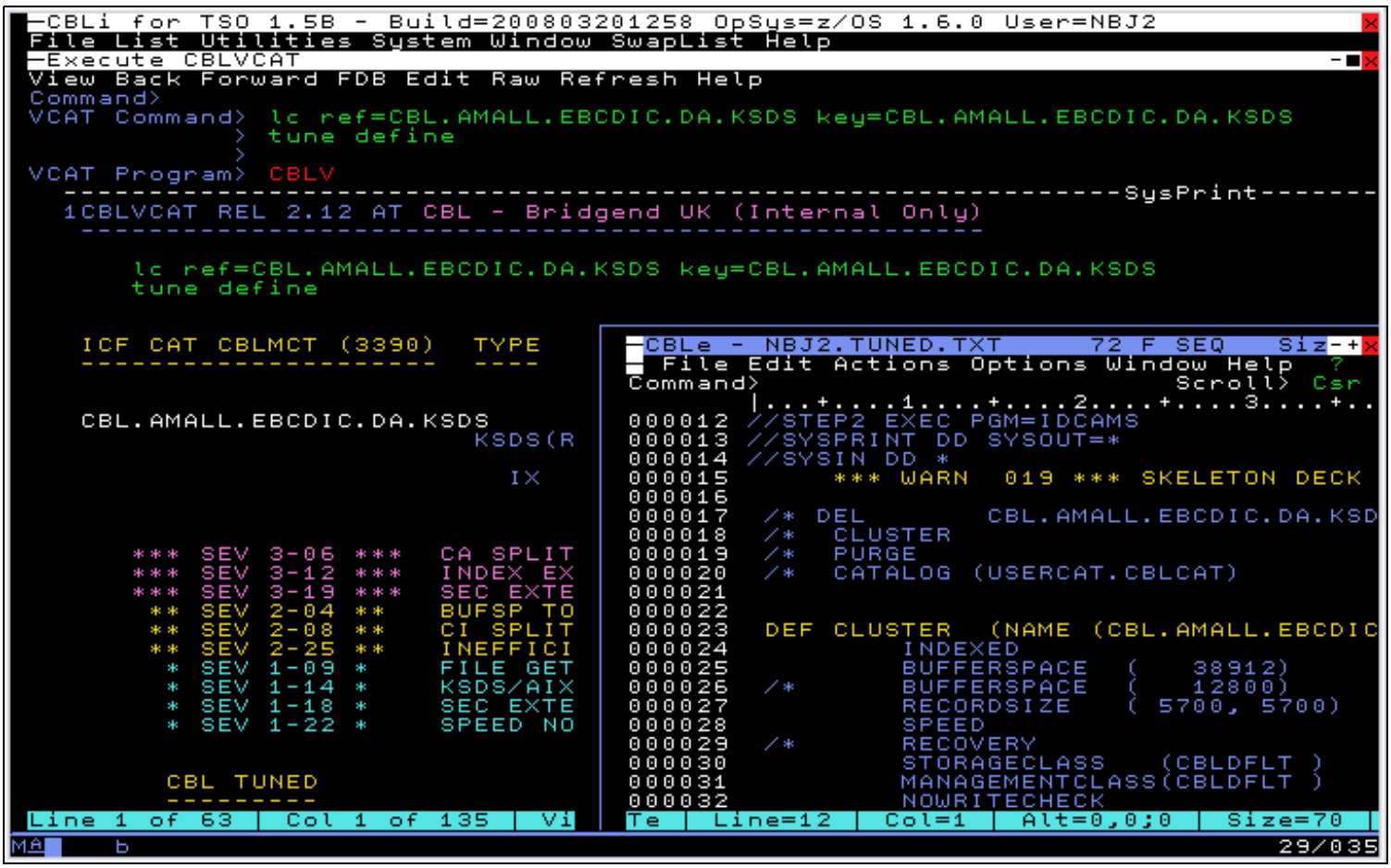


Figure 6. CBLVCAT TUNE Output.

VSE: ERASE and RENAME support

ERASE and RENAME has been enhanced to include support for sequential and VSAM files on VSE systems. Support already exists for ERASE and RENAME on CMS and MVS systems.

CBLVCAT must be licensed to be able to perform ERASE and RENAME on VSE sequential files. This is because CBLi utilises the CBLVCAT DEL and MOD operations to perform these functions.

The following VSE List type windows now support the "D" (Delete), "K" (Kill) and "R" (Rename) prefix commands:

1. Catalog List
2. CBLVCAT Raw List
3. VTOC List
4. VTOC Extents List
5. Execute CBLVCAT Window

CBLi ERASE Command for VSE

Syntax:

```
>>-- ERAse -----+- volid ---+-- : -- fileid ----------------------------------><
                |           |
                +- catdsn --+
```

Description:

Erase (delete) the specified sequential or VSAM file.

Parameters:

volid
The volume serial number of the DASD volume on which the sequential file resides.
catdsn
The full fileid of the VSAM catalog to which the VSAM managed file belongs.

fileid
The full fileid of the file to be erased.

Examples:

`erase SYSWK1:CBL.SELCOPY.NAM`
Erase sequential file CBL.SELCOPY.NAM on SYSWK1. (CBLVCAT must be licensed.)
`erase VSESP.USER.CATALOG:CBL.TEST.KSDS`
Erase VSAM managed data set CBL.TEST.KSDS cataloged in the VSAM catalog, VSESP.USER.CATALOG.

CBLi RENAME Command for VSE

Syntax:

```
>>-- REName ----+- volid ---+-- : -- fileid1 --------- fileid2 -------><
                |           |
                +- catdsn --+
```

Description:

Rename the specified sequential or VSAM file.

Parameters:

volid
The volume serial number of the DASD volume on which the sequential file resides.
catdsn
The full fileid of the VSAM catalog to which the VSAM managed file belongs.

fileid1
The current fileid in full of the file to be renamed.

fileid2
The new fileid to be assigned to the file.

Examples:

`rename SYSWK1:CBL.SELCOPY.NAM CBL.SELCOPY.NAM.NEWNAME`
Rename a sequential file. (CBLVCAT must be licensed.)
`rename VSESP.USER.CATALOG:CBL.TEST.KSDS CBL.TEST.KSDS.NEWNAME`
Rename a VSAM managed data set.

VSE: Catalog List Window

Catalog Listing has been enhanced to include support for files in VSE VSAM catalogs. Support already exists for listing cataloged data sets on MVS systems.

CBLVCAT must be licensed to be able to generate VSE VSAM catalog lists. This is because CBLi utilises the CBLVCAT LISTVCAT operation to perform this function.

Catalog List Window for VSE

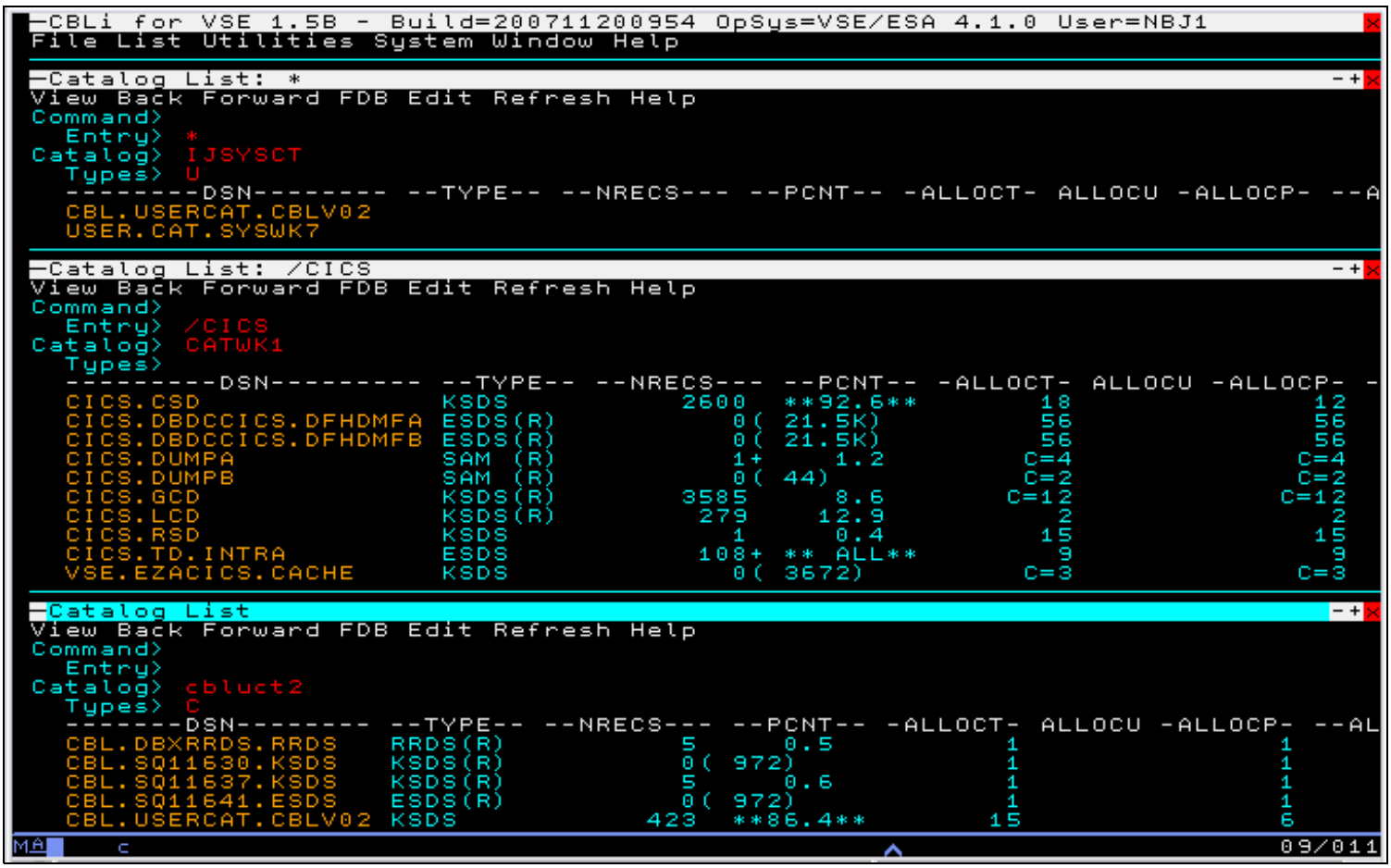


Figure 7. Catalog List Window for VSE.

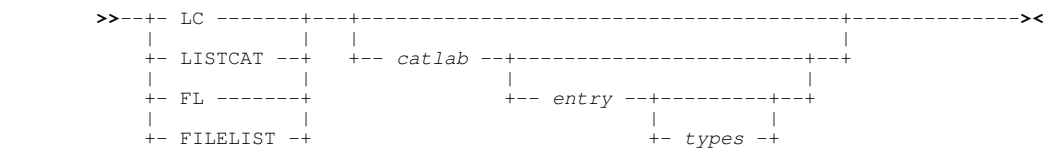
Prefix Line Commands

The following prefix line commands are available in the CBLVCAT Catalog List window:

Command	Description
D	Delete the entry. User will be prompted to verify the deletion.
K	Delete (Kill) the entry without prompting for verification.
R	Rename the entry.
T	Open an Execute CBLVCAT window and issue a LISTVCAT TUNE DEFINE operation for the entry.
V	Open an Execute CBLVCAT window and issue a LISTVCAT operation for the entry.

CBLi LISTCAT Command for VSE

Syntax:



Description:

Use the LC (List Catalog entries) command to open a Catalog List window and optionally list information about entries in the VSAM catalog.

The following CBLVCAT LISTVCAT operation is generated and passed to CBLVCAT:

```
  LISTVCAT  DD=catlab   KEY=entry
```

The Catalog List window may also be opened via the List menu of the CBLi main window menu bar.

Parameters:

catlab
: A disk label assigned to the VSAM catalog for which entries are listed. Default is the master catalog.
This parameter is placed in the "Catalog>" field of the Catalog List window.

entry
: A valid CBLVCAT LISTVCAT KEY parameter file mask. i.e. entries with file name **beginning** with *entry* or if prefixed by "/" (slash) entries with file name **containing** *entry*. (See CBLVCAT User Manual)
This parameter is placed in the "Entry>" field of the Catalog List window.

types
: Specifies the catalog entry types required. Default is all types. One or more of the following types may be specified with no intervening blanks:

A	VSAM SAM data set.
C	Cluster.
G	Alternate Index.
R	VSAM PATH.
X	Alias.
U	User catalog connector entry.

This parameter is placed in the "Types>" field of the Catalog List window.

Examples:

`lc VSESPUC CICS`
: List all entries in the VSESPUC VSAM catalog that have file names beginning "CICS".

`lc IJSYSUC /SELC A`
: List all VSAM managed SAM entries in the IJSYSUC VSAM catalog that have file names containing "SELC".

VSE: POWER Command Window Prefix Commands

The VSE Power Command Window, displaying lists of entries in the power queues, has been enhanced to support prefix commands 'K' (Kill) and 'D' (Delete) allowing trusted or authorised users to remove entries from the power queue.

All: CBLe LIST LVR/POWER/STRUCTURE

CBLe CLI LIST command enables users to extract rows of data returned from various CBLi LIST type commands and either place the output in a temporary edit view or assign the fields to REXX stem variables for use in a CBLe REXX macro.

The LIST command has been enhanced to support VSE POWER Queue list output, MVS SDE DISPLAY STRUCTURE output and the, new to CBLi 1.50, CBLVCAT Raw list output.

Syntax:

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

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

Parameters:

listtype
The CBLi list type function to extract. The following have been added in CBLi 1.50:

LVR	List CBLVCAT Raw output.
POWER	VSE only: POWER Queue list output.
STRUCTure	MVS only: Structured Data Environment DISPLAY STRUCTURE list output.

MVS: CBLe Environment Variable %TSOPREFIX%

CBLe supports a set of system determined and user defined environment variables that may be used in CBLe commands executed from the command line, via command execution (CMX) files and via macros.

Support for the following standard CBLe environment variable has been included when running CBLi in an MVS TSO environment:

VarName	Description
tsoprefix **tsopfx**	The defined TSO prefix. In many TSO environments, this has the same value as the %USER% userid environment variable.

Examples:

```
LD    %tsopfx%.DEV.JCLLIB
GET   %tsopfx%.IX016273.%mm%%dd%%yyyy%.LOG
```

All: REFRESH for CBLVCAT, LIST & POWER Windows

List type window displays (including the Execute CBLVCAT, File Search, Dynamic SQL and Power Command output windows) are updated automatically when a list prefix command is exceuted within the list window to delete, rename or copy entries.

However, where alterations to file entries occur outside of the list window (e.g. via a second list window or a CBLe CLI command), the list window contents are unchanged.

The "Refresh" menu item has been introduced in CBLi 1.50 to allow users to easily refresh the contents of the list so that all field columns reflect the current status.

All: CBLVCAT & SQL as MDI Child Windows

In order to bring Execute CBLVCAT and Dynamic SQL windows into line with other list type windows, these are now supported as MDI child windows of the CBLe text edit and Interactive SELCOPY MDI applications in CBLi.

When the CBLe option MDILIST is set ON (the default), any list window started from within CBLe (using a CBLi CLI list type command executed from a CBLe command prompt, CMX file or CBLe REXX macro) is opened as an MDI child of CBLe. When MDILIST is set OFF, the List window is opened as a child of the CBLi main window.

MVS: CBLi Main Menu Bar SWAP -> SWAPLIST

For CBLi 1.50, the CBLi main menu bar item "Swap" (ISPF SWAP), present when executing CBLi as an MVS ISPF application, has been replaced with the item "SwapList" (ISPF SWAPLIST).

"SwapList" was deemed more useful as it allows users to start new ISPF split screens or direct focus to any of the existing ISPF split screens. "Swap" may still be achieved using the CBLi CLI command, ISPF, followed by ISPF command string, SWAP.

All: CBLi 1.40 Zaps applied

Zap ID	Op. Sys.	Query Ref.	Description
I140z01	All	(IQ01039 - 2007/02/02)	CBLe: 0C4 Abend when a menu-bar item of a List window (e.g. LD output) is accidentally overtyped.
I140z02	MVS	(IQ01041 - 2007/02/19)	List type commands that perform **CATALOG searches** (i.e FL, LC, LD) where supplied filemask has asterisk as the first qualifier (e.g. 'LC *.CBLI.INI') should give a warning that ALL catalogs will be searched, which for **very large systems** takes a great deal of time and resource. Prior to this zap the warning occured only if the supplied filemask was '*.**'. This zap also ensures that filemask 'hlq' is treated as 'hlq.**' (in line with ISPF 3.4) instead of 'hlq*.**' as previously.
I140z03	All	(IQ01038 - 2007/02/19)	INTERFACE=ISPF edit command "FIND PREFIX" gave: `EDT105E No find string (string-1) given on the FIND command.` 'FIND' and 'CHANGE' should treat keywords such as 'PREFIX' as string arguments if no others are provided.
I140z04	All	(IQ01055 - 2007/02/19)	Focus is not returned to the correct window when a DIALOG window is opened by a REXX edit macro.
I140z05	All	(IQ01057 - 2007/02/19)	0C4 at VCIFCBLV+07EA after LOSTERM condition while CBLIVTAM user in SELCOPY Interactive.
I140z06	All	(IQ01065 - 2007/03/19)	0C1 at MEMFBSP0+097E when doing a descending sort on a LIST enum column where one of the rows in the list has an invalid enum value (e.g. RECFM field in a LV (VTOC) list for the VVDS entry). Could also show up as an abend in CNVFLVO0 at various offsets.
I140z07	All	(IQ01066 - 2007/03/19)	Chained commands separated with the LINEND character are potentially issued in the wrong order.
I140z08	MVS	(IQ01071 - 2007/03/19)	The CBLi CBLVCAT SVC is wrongly reported as not found in the LPA even though system LPA list shows it is in the LPA. A CBLVCAT catalog list gives: `VCII019E SVC module IGX00222 not found in either static or dynamic LPA.`
I140z09	MVS	(IQ01089 - 2007/09/18)	Previously the rule for LISTCAT/LISTDATASET commands stated that if the provided filemask contains any wildcards, then all wildcards must be explicitly provided. For compatibilty with ISPF 3.4, this has been altered so that '*.**' (or '.**' if last qualifier is len=8) is appended to the filemask regardless of the presence of a wildcard, unless either of the following are true: 1. Last character is '*' in which case there is no modification to the filemask. 2. Last character is '.' (in order to indicate the preceding qualifier must be the last) in which case the '.' is stripped from the filemask. See also I140z02 which is a prerequisite. I140z09 also extends the warning that ALL catalogs are to be searched for filemasks begining '**.'
I140z10	VSE	(IQ01095 - 2007/09/18)	Immediate termination of CBLIVTAM following logon when CBLNAME option 'SSP2RetCode=Yes' is in effect (or running SELCOPY 2.00 without S200z35).
I140z11	VSE	(IQ01095 - 2007/09/18)	Protection Exception following failed attempt to allocate GETVIS below 16M for partitions that are larger than 16M. Problem caused by failure to free previously allocated I/O buffers.
I140z12	All	(IQ01098 - 2007/09/18)	Support CBLi command 'CBLICAN' to cancel the user's CBLi session, bypassing the usual prompt: `Do you want to quit this CBLi session?`
I140z13	All	(IQ01136 - 2007/09/18)	Recursive loop in CBLVDUMP.

I140z14	All	(IQ01162 - 2007/09/18)	Support CBLi command 'HOME' (abbrev HO) to pass focus directly to the user's command-centre (CMX) file edit window.
I140z15	MVS	(IQ01194 - 2007/10/29)	Use of the 'C' prefix command from a LC/LD output window in order to 'COPY' to a new dataset, fails if UNITNAME contains the literal 'Shared'.

All: CBLi 1.40 Fixes applied at Source

Op. Sys.	Description
MVS	Recursive loop in CBLi ABEND Trap and formatted dump facility.
MVS	Trap abends in Edit macros when executing in TSO REXX environment.
VSE	Exit without terminating CBLIVTAM if obsolete SELCOPY 2.00 is active without s200z35.
All	Trap user arithmetic overflow within SELCOPY execution in SELCOPY Interactive.
All	Parm values not passed to SELCOPY in SELCOPY Interactive environment.
MVS	Using model data set in "Define KSDS" dialog window populates STORAGECLASS field with model MANAGEMENTCLASS value.
All	Fix 0C4 Abend in Calendar utility when invalid month entered.
All	Avoid potential abend by switching AMODE when appropriate for internal calls to IDCAMS