



**Structured Data Environment (SDE)**

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

<b>Documentation Notes.....</b>	<b>1</b>
<b>Introduction to SDE.....</b>	<b>2</b>
<b>Getting Started with SDE.....</b>	<b>3</b>
Handling Long Command Strings.....	3
Opening an SDE Edit Window.....	3
Creating a Structure Definition Object from a COBOL Copy Book.....	3
Editing a Structured Data Set.....	3
Applying Additional Record Type Selection Criteria.....	4
Changing the Record Display.....	5
Displaying Records of Specific Record Type(s).....	6
Display Unformatted Record Data.....	6
Displaying a Record in Single Record View.....	7
Displaying Hexadecimal Representation of Record Data.....	8
Excluding Records from the Display.....	8
Displaying Selected Columns.....	9
Displaying the Record Headers.....	10
Searching Data Records.....	11
Finding Record Data by Search String.....	11
Locating Records Using a Search Expression.....	12
Creating a Subset of Records for Display.....	13
Editing Data.....	14
Typing New Record Data.....	14
Find and Replace Data.....	15
Inserting Records.....	15
Deleting Records.....	16
Moving Records.....	17
Copying Records.....	17
Duplicating Records.....	17
Undo and Redo Changes.....	17
<b>SDE Concepts.....</b>	<b>19</b>
Record Structure Definition.....	19
SDE Window Views.....	19
SDE Data Formats.....	21
SDE Data Types.....	22
SDE Current Window.....	24
Default Record Type.....	24
REXX Macros.....	24
<b>Command Line (Primary) Commands.....</b>	<b>26</b>
Executing SDE Commands from a CBLc Text Edit Window.....	26
Command Reference Syntax Conventions.....	26
BOTTOM.....	27
BROWSE.....	28
CHANGE.....	30
CREATE LIST.....	34
CREATE STRUCTURE.....	35
DISPLAY LIST.....	40
DISPLAY STRUCTURE.....	41
DOWN.....	43
DROP, DDROP.....	44
DUPLICATE.....	45
EDIT.....	46
END.....	47
EXCLUDE.....	48
EXTRACT.....	51
FIND.....	51
FLIP.....	55
FORMAT.....	55
HEX.....	56
INSERT.....	56
LEFT.....	57
LESS.....	59
LOCATE.....	60
MORE.....	61
MACRO.....	62
QUERY.....	62
QUIT.....	63
RCHANGE.....	63
REDO.....	63
RESET.....	64
RFIND.....	65
RIGHT.....	65
SAVE.....	67
SELECT.....	68

# Contents

## Command Line (Primary) Commands

SET.....	68
SORT.....	69
TOP.....	69
UNDO.....	69
UP.....	70
USE.....	71
VIEW.....	73
WHERE.....	73
ZOOM.....	76

## SET/QUERY/EXTRACT.....77

Syntax.....	77
DRECTYPE - SET/QUERY/EXTRACT Option.....	78
FIELD - EXTRACT Option.....	78
FOCUS - EXTRACT Option.....	79
MSGLINE - SET/QUERY/EXTRACT Option.....	80
MULTIPOINT - SET/QUERY/EXTRACT Option.....	81
POINT - SET/QUERY/EXTRACT Option.....	81
PREFIX - SET/QUERY/EXTRACT Option.....	82
REFERENCE - SET/QUERY/EXTRACT Option.....	83
REGION - QUERY/EXTRACT Option.....	84
SCALE - SET/QUERY/EXTRACT Option.....	84
SHADOW - SET/QUERY/EXTRACT Option.....	85
TYPE - SET/QUERY/EXTRACT Option.....	86
UNDOING - SET/QUERY/EXTRACT Option.....	87
UNNAMED - SET/QUERY/EXTRACT Option.....	88
USING - QUERY/EXTRACT Option.....	88
VALUE - EXTRACT Option.....	89
WRAP - SET/QUERY/EXTRACT Option.....	89

## Prefix Area (Line) Commands.....91

## Function Keys.....92

## Glossary.....93

# Documentation Notes

The **CBLi Reference and User Guide**, **CBLi Text Editor Manual** and **Structured Data Editor Manual** 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, CBLi and SD Editor 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, CBLi and SD Editor system and programs.

No reproduction of the whole or any part of the CBLi, CBLi or SD Editor 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. Where the program product differs from that stated herein, Compute (Bridgend) Ltd reserve the right to revise either the program or its documentation at their discretion.

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.

# Introduction to SDE

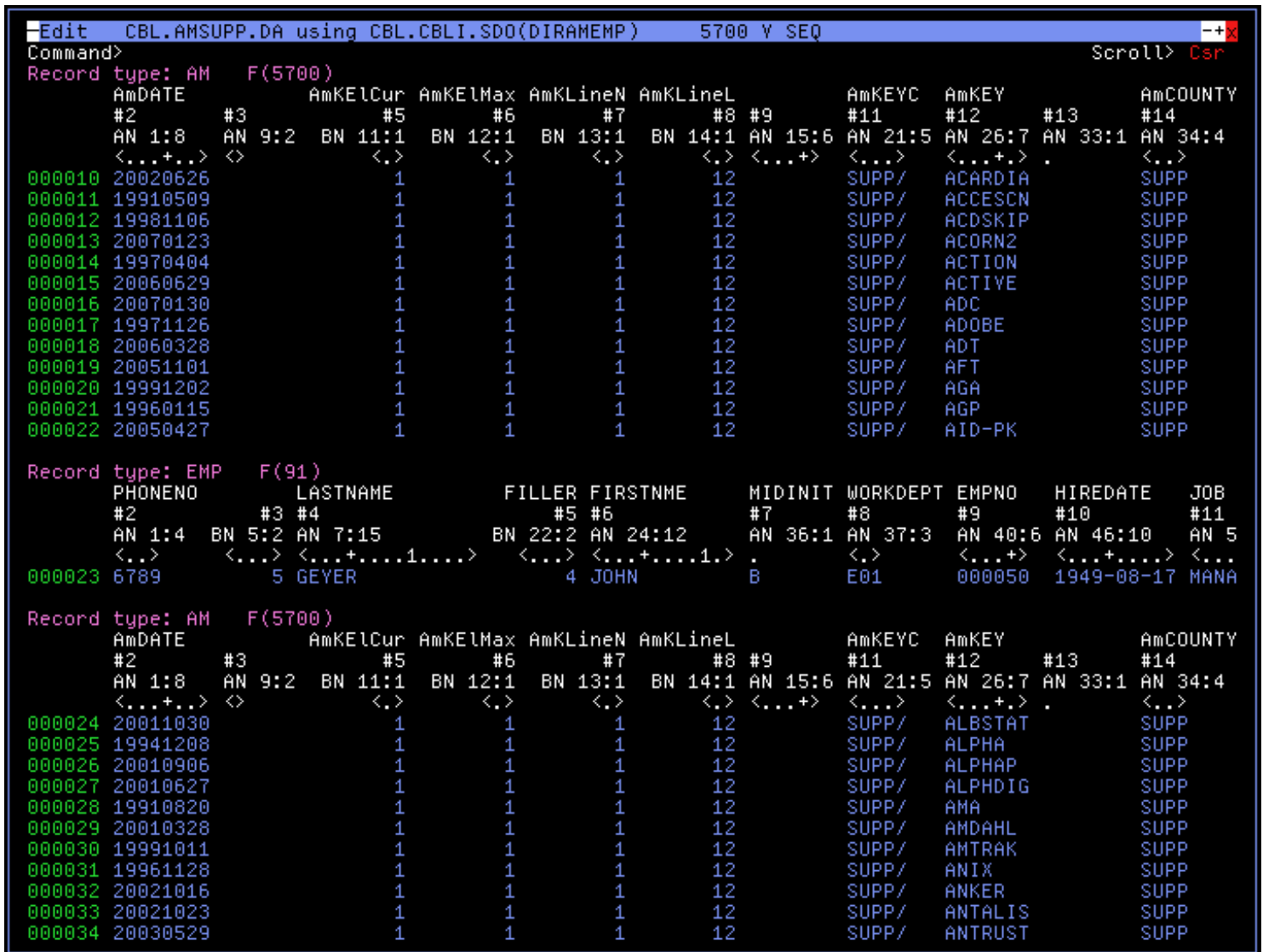
Data set records may have an associated file structure that maps field information (position, length and data type) for all data within each data set record. These structures often exist as a PL/1 or COBOL copybook.

The CBLi Structured Data Environment (SDE) allows users to display and process structured data sets using a pre-defined SDE structure so that record data is formatted and arranged in field columns. An SDE structure may be generated from a copybook or using SDE's Create Structure internal syntax, and can contain a number of mappings, one for each different type of data set record.

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

CBLi SDE enables users to do the following:

- Display, copy, edit and update structured data.
- Concurrently display records of more than one record type within the same view.
- Work with data in multiple or single record views.
- Select and order the fields displayed for each record type.
- Define new record structures using CBLi SDE syntax.



Edit CBL.AMSUPP.DA using CBL.CBLI.SDO(DIRAMEMP) 5700 V SEQ													
Command>													
Record type: AM F(5700)													
	AmDATE	AmKElCur	AmKElMax	AmKLineN	AmKLineL	AmKEYC	AmKEY	AmCOUNTY					
	#2	#3	#5	#6	#7	#8	#9	#11	#12	#13	#14		
	AN 1:8	AN 9:2	BN 11:1	BN 12:1	BN 13:1	BN 14:1	AN 15:6	AN 21:5	AN 26:7	AN 33:1	AN 34:4		
	<...+...>	<>	<.>	<.>	<.>	<.>	<...+>	<...>	<...+...>	.	<...>		
000010	20020626		1	1	1	12		SUPP/	ACARDIA		SUPP		
000011	19910509		1	1	1	12		SUPP/	ACCESC		SUPP		
000012	19981106		1	1	1	12		SUPP/	ACDSKIP		SUPP		
000013	20070123		1	1	1	12		SUPP/	ACORN2		SUPP		
000014	19970404		1	1	1	12		SUPP/	ACTION		SUPP		
000015	20060629		1	1	1	12		SUPP/	ACTIVE		SUPP		
000016	20070130		1	1	1	12		SUPP/	ADC		SUPP		
000017	19971126		1	1	1	12		SUPP/	ADOBE		SUPP		
000018	20060328		1	1	1	12		SUPP/	ADT		SUPP		
000019	20051101		1	1	1	12		SUPP/	AFT		SUPP		
000020	19991202		1	1	1	12		SUPP/	AGA		SUPP		
000021	19960115		1	1	1	12		SUPP/	AGP		SUPP		
000022	20050427		1	1	1	12		SUPP/	AID-PK		SUPP		
Record type: EMP F(91)													
	PHONENO	LASTNAME	FILLER	FIRSTNME	MIDINIT	WORKDEPT	EMPNO	HIREDATE	JOB				
	#2	#3 #4	#5	#6	#7	#8	#9	#10	#11				
	AN 1:4	BN 5:2 AN 7:15	BN 22:2	AN 24:12	AN 36:1	AN 37:3	AN 40:6	AN 46:10	AN 5				
	<...>	<...>	<...+...1...>	<...>	<...+...1...>	.	<...+>	<...+...>	<...>				
000023	6789	5 GEYER		4 JOHN	B	E01	000050	1949-08-17	MANA				
Record type: AM F(5700)													
	AmDATE	AmKElCur	AmKElMax	AmKLineN	AmKLineL	AmKEYC	AmKEY	AmCOUNTY					
	#2	#3	#5	#6	#7	#8	#9	#11	#12	#13	#14		
	AN 1:8	AN 9:2	BN 11:1	BN 12:1	BN 13:1	BN 14:1	AN 15:6	AN 21:5	AN 26:7	AN 33:1	AN 34:4		
	<...+...>	<>	<.>	<.>	<.>	<.>	<...+>	<...>	<...+...>	.	<...>		
000024	20011030		1	1	1	12		SUPP/	ALBSTAT		SUPP		
000025	19941208		1	1	1	12		SUPP/	ALPHA		SUPP		
000026	20010906		1	1	1	12		SUPP/	ALPHAP		SUPP		
000027	20010627		1	1	1	12		SUPP/	ALPHDIG		SUPP		
000028	19910820		1	1	1	12		SUPP/	AMA		SUPP		
000029	20010328		1	1	1	12		SUPP/	AMDAHL		SUPP		
000030	19991011		1	1	1	12		SUPP/	AMTRAK		SUPP		
000031	19961128		1	1	1	12		SUPP/	ANIX		SUPP		
000032	20021016		1	1	1	12		SUPP/	ANKER		SUPP		
000033	20021023		1	1	1	12		SUPP/	ANTALIS		SUPP		
000034	20030529		1	1	1	12		SUPP/	ANTRUST		SUPP		

Figure 1. Structured Data Environment View.

# Getting Started with SDE

The CBLi distribution includes sample structured data set members, COBOL copy books that map records within these members and the &PREFIX..CBLI.CMX(SDE) command centre (CMX) data set. The SDE CMX data set guides new users through some basic tasks related to structured data environment text editing, demonstrating each task by prompting the user to execute commands against the sample data.

It is strongly recommended that users take time to work their way through the SDE CMX data set in order to familiarise themselves with some of the features and capabilities of SDE edit.

Basic SDE edit tasks are also discussed in this section of the SDE documentation.

## Handling Long Command Strings

By default, CBLi windows, including SDE window views, include a single command line for entering commands to be passed to the local Command Line Interface (CLI).

Whereas it is recommended that command strings, particularly command strings of any significant length, are executed from and stored for later use in the user's HOME (CMX command centre) file, a window's command line may be extended to allow input of long command strings.

The **Command Line** dialog window includes the "Lines>" field which defines the number of continuous command lines (between 1 and 9) to be presented in the current window. Having updated this value and selected OK, the change will take immediate effect.

The Command Line dialog window is opened by selecting the "Command Line ..." option of the **system menu**, (select the **system menu** button to the left of the window's title bar). Alternatively, execute the **COMMANDLINE** CBLi CLI command (synonym **CLN**).

Any text entered on second and subsequent command lines is appended to text on the command line before.

## Opening an SDE Edit Window

Basic tasks related to starting an SDE edit session for structured data is discussed in the following:

- **Creating a Structure Definition Object from a COBOL Copy Book**
- **Editing a Structured Data Set**
- **Applying Additional Record Type Selection Criteria**

## Creating a Structure Definition Object from a COBOL Copy Book

Before SDE editing of a structured data set can be performed, an SDE structure definition object (SDO) must exist that contains one or more record type mappings for records in the data set. An SDO is generated using the SDE **CREATE STRUCTURE** CLI command which accepts a COBOL copy book DSN as one of its input sources.

e.g. To generate the structure definition object NBJ.CBLI.SDO(COBTYPES) for an existing COBOL copy book CBL.DIST.CBLI.SDE.SAMP.F80(COBTYPES), execute the following from a CBLi text edit command prompt (or via a CMX command centre file):

```
SDATA    CREATE STRUCT  NBJ.CBLI.SDO(COBTYPES)    FROM  CBL.DIST.CBLI.SDE.SAMP.F80(COBTYPES)
```

### Notes:

1. **SDATA** is a CBLi CLI command that submits command streams to the SDE environment command processor.
2. If executing commands from a command line that is unable to accommodate all of the command text, increase the number of command input lines using the "Command Line" dialog available via the system menu button on the window title bar or by executing the **CLN** CBLi CLI command.

## Editing a Structured Data Set

Having created an SDO, we can use it to apply structure to an edit of a data set via the SDE EDIT CLI command. e.g.

```
SDATA    EDIT    CBL.DIST.CBLI.SDE.SAMP.VAR(DATTYPES)    USING  NBJ.CBLI.SDO(COBTYPES)
```

Where the record length of a record within the edited data set matches that implied by a record type mapping within the SDO, then that record type mapping is applied to the record for the duration of the edit session.

If no suitable match exists within the SDO, the record is flagged as being NOT SELECTED and a default record mapping of "Record" (consisting of a single character field with length equal to the record length) is applied to the record data.

```

-CBLi for TSO 1.5B - Build=200804231235 OpSys=z/OS 1.6.0 User=NBj2
File List Utilities System Window SwapList Help
CBLi - Edit CBL.DIST.CBLI.SDE.SAMP.VAR(DATTYPES) using NBj2.CBLI.SDO(COBT
File Edit Actions Options Window Help ? Sv ToF BoF wS wR Pfx < > Scroll> Csr
Command>
000000 *** Top of Data ***
Record type: DATA-TYPES-1 V(39,49)
MYREC MYBINTEGER MYCHARACTER MYDECIMAL MYFIXED MYFLOAT MYHE
#2 #3 #4 #5 #6 #7 #8
AN 1:1 FB 2:2 AN 4:4 PD 8:4 FB 12:4 FP 16:4 AN 2
- <> <--> <---+---> <---+---1> <---+---1---> <--->
000001 1 36 abcd 1234.56 1234567.89 +0.10000000E+02 ..~.
000002 1 85 abcd -1234.56 -1234567.89 +0.10000000E+01 1...
000003 2 98 34 . ***** -9768284.72 +0.00000000E+00 cdef
000004 2 98 34 . ***** -9768284.72 +0.00000000E+00 cdef
000005 1 46 abcd 1234.56 8090640.56 +0.95367431E-06 ...
000006 1 01 abcd 1234.56 8090640.56 +0.83378875E-56 ....
000007 *** End of Data ***

```

Se Line=0 Col=1 Alt=0,0;0 Size=6 Recl=16384 Fmt=V Files=1 Views  
MA b 05/011

Figure 2. SDE Edit View.

Where more than one matching record type exists in the SDO, the first matching record type is used. If this is incorrect, the user can update the SDO to apply additional criteria for record type selection based on the contents of fields within the record.

In the screen shot above, the record type "DATA-TYPES-1" has been assigned to all records in the data set CBL.DIST.CBLI.SDE.SAMP.VAR(DATTYPES). However, records 3 and 4 are not mapped correctly by this record type.

## Applying Additional Record Type Selection Criteria

Using a record's length as the only basis of identifying a matching record type may yield incorrect record mappings.

In order to sophisticate record type selection, additional criteria may be added to the definition of record types within the SDO. This is achieved via the SDE **USE** CLI command which applies selection criteria to record types based on the contents of fields within the data records.

For our DATTYPES example, the following USE commands may be executed to add additional record match criteria to the "DATA-TYPES-1" and "DATA-TYPES-2" record type definitions within the SDO. The record type assigned to a data record will be based on the record's length and the contents of the character field, "MYREC". From an SDE window command prompt:

```

USE DATA-TYPES-1 IN NBj.CBLI.SDO(COBTYPES) WHEN MYREC='1'
USE DATA-TYPES-2 IN NBj.CBLI.SDO(COBTYPES) WHEN MYREC='2'

```

Hit <PF3> to quit the SDE edit window, then re-opening the edit view using the same EDIT command as before, we can see that the "DATA-TYPES-2" record type definition has been assigned to records 3 and 4. By default, SDE displays all records that are assigned a record type and also displays the record headers before each new group of records with the same record type.

```

-CBLi for TSO 1.5B - Build=200804231235 OpSys=z/OS 1.6.0 User=NBj2
File List Utilities System Window SwapList Help
CBLi - Edit CBL.DIST.CBLI.SDE.SAMP.VAR(DATTYPES) using NBj2.CBLI.SDO(COBT
File Edit Actions Options Window Help ? Sv ToF BoF wS wR Pfx < >
Command> Scroll> Csr
000000 *** Top of Data ***
Record type: DATA-TYPES-1 V(39,49)
MYREC MYBINTEGER MYCHARACTER MYDECIMAL MYFIXED MYFLOAT MYHE
#2 #3 #4 #5 #6 #7 #8
AN 1:1 FB 2:2 AN 4:4 PD 8:4 FB 12:4 FP 16:4 AN 20:
- <> <--> <---+---> <---+---1> <---+---1---> <--->
000001 1 36 abcd 1234.56 1234567.89 +0.10000000E+02 ..~.
000002 1 85 abcd -1234.56 -1234567.89 +0.10000000E+01 1...

Record type: DATA-TYPES-2 V(42,46)
MYREC MYZONED MYINTEGER MYXVARCHAR MYLL MYSTRCH(1) MYSTRCH(2) MYSTRC
#2 #3 #4 #5 #7 #9 #9 #9
AN 1:1 ZD 2:4 FB 6:2 AN 8:8 FB 16:2 AN 18:1 AN 19:1 AN 20:
- <---+> <---> <---+---> <---> - - -
000003 2 12.34 8 ABCDEFGH ***** 7 a b c
000004 2 12.34 8 ABCDEFGH

Record type: DATA-TYPES-1 V(39,49)
MYREC MYBINTEGER MYCHARACTER MYDECIMAL MYFIXED MYFLOAT MYHE
#2 #3 #4 #5 #6 #7 #8
AN 1:1 FB 2:2 AN 4:4 PD 8:4 FB 12:4 FP 16:4 AN 20:
- <> <--> <---+---> <---+---1> <---+---1---> <--->
000005 1 46 abcd 1234.56 8090640.56 +0.95367431E-06 ....
000006 1 01 abcd 1234.56 8090640.56 +0.83378875E-56 ....
000007 *** End of Data ***

Se Line=0 Col=1 Alt=0,0;0 Size=6 Recl=16384 Fmt=V Files=1 Views
MA b 05/011

```

Figure 3. SDE Edit View with Updated SDO.

After the SDE edit window view is ended, the accompanying SDO will remain in storage until either the CBLx MDI parent window is closed or the SDO is dropped from storage using the SDE **DROP** CLI command. At this point, if changes have been made to the SDO via the USE command, the user will be prompted to save the updated SDO to disk.

e.g. Having first ended the SDE edit view, execute the following from a CBL text edit window:

SDATA DROP NBJ.CBLI.SDO(COBTYPES)

**Notes:**

1. The SDE DROP command will give an error if an attempt is made to drop an SDO that is in use.

## Changing the Record Display

The user can tailor the appearance of the SDE window view and the records displayed within.

An SDE window view is an MDI child window and can be resized and positioned using standard window manipulation as documented in the "*CBLi Reference and User Guide*". General manipulation of the SDE window contents is discussed in the following:

- Displaying Records of Specific Record Type(s)
- Display Unformatted Record Data
- Displaying a Record in Single Record View
- Displaying Hexadecimal Representation of Record Data
- Excluding Records from the Display
- Displaying Selected Columns
- Displaying the Record Headers



## Displaying Records of Specific Record Type(s)

The records displayed when the SDE window view is opened is determined by the VIEW parameter of the EDIT command. By default, all records that have an assigned record type are displayed.

If **SHADOW NOTSELECTED** is set on (the default), consecutive records that are not assigned a record type are represented by a single shadow line of type "NOT SELECTED". Otherwise, if **SHADOW NOTSELECTED** is set off, these records are not displayed.

Once the SDE window view has been opened, the display of selected records may be restricted to those assigned specific record type(s) using the SDE **VIEW** CLI command.

e.g. To display only records assigned record type "REC-CUST" or "REC-ORDER", execute the following from the SDE window view command line:

```
VIEW REC-CUST, REC-ORDER
```

Records assigned to records types that are not specified as arguments on the VIEW command, are flagged as being **SUPPRESSED**. If **SHADOW SUPPRESSED** is set on (the default), consecutive suppressed records that are assigned the same record type are represented by a single shadow line of type "SUPPRESSED".

```

-CBLi for TSO 1.5B - Build=200804231235 OpSys=z/OS 1.6.0 User=NBj2
File List Utilities System Window SwapList Help
CBLi - Edit NBj2.CBLI.SDE.SAMP.VAR(DATSALES) using NBj2.CBLI.SDO(COBSALES)
File Edit Actions Options Window Help ? Sv ToF BoF wS wR Pfx < >
Command> VIEW REC-CUST, REC-ORDER Scroll> Csr
Record type: REC-CUST F(192)
  CUST-ID PASS LASTNAME FIRSTNAME COUNTRY POSTCODE
  #2 #3 #4 #5 #6 #7
  FB 1:2 AN 3:15 AN 18:15 AN 33:15 AN 48:2 AN 50:12
  <---> <---+---1---> <---+---1---> <---+---1---> <---+---1--->
000001 9156 78fj2foa Ramstein Hans UK DN4 9BR
000002 7037 chico Richards Denise UK CF31 1BX
000003 6712 benj Hill Mike US 75204
000004 ----- 2 line(s) suppressed: record type REC-CARD -----

Record type: REC-ORDER V(86,106)
  CUST-ID ORDER-REF QTY ITEM-CODE UNIT-PRICE DELIVERY ORDER-DATE
  #2 #3 #4 #5 #6 #7 #8
  FB 1:2 FB 3:4 FB 7:2 FB 9:2 PD 11:4 PD 15:3 PD 18:5
  <---> <---+---> <---> <---> <---+---> <---+---> <---+--->
000006 9156 557833807 7 18385 84.82 4.79 20051022
000007 9156 696534904 1 19692 128.03 7.94 20070823
000008 ----- 1 line(s) suppressed: record type REC-PAYMENT -----
000009 ----- 1 line(s) suppressed: record type REC-CARD -----
000010 ----- 1 line(s) suppressed: record type REC-PAYMENT -----

Record type: REC-CUST F(192)
  CUST-ID PASS LASTNAME FIRSTNAME COUNTRY POSTCODE
  #2 #3 #4 #5 #6 #7
  FB 1:2 AN 3:15 AN 18:15 AN 33:15 AN 48:2 AN 50:12
  <---> <---+---1---> <---+---1---> <---+---1---> <---+---1--->
000011 3253 marine-boy Coustou jacque FR PA223 78
000012 ----- 3 line(s) suppressed: record type REC-CARD -----

Record type: REC-ORDER V(86,106)
  CUST-ID ORDER-REF QTY ITEM-CODE UNIT-PRICE DELIVERY ORDER-DATE
  #2 #3 #4 #5 #6 #7 #8
  FB 1:2 FB 3:4 FB 7:2 FB 9:2 PD 11:4 PD 15:3 PD 18:5
  <---> <---+---> <---> <---> <---+---> <---+---> <---+--->
000015 1820 756551020 8 32417 39.99 7.85 20060625
000016 4596 989428617 10 25366 142.27 5.33 20060503
000017 1812 898652388 4 22235 202.38 3.23 20050917
Se Line=1 Col=1 Alt=0,0;0 Size=63 Recl=16384 Fmt=V Files=1 View
MA b 05/035

```

Figure 4. SDE View Selected Record Types.

To re-display **all** records assigned a record type, execute the following from the SDE window view command line:

```
VIEW *
```

## Display Unformatted Record Data

The initial format of the record data in the SDE window view is determined by the FORMAT parameter of the EDIT command. By default, FORMAT TABLE is used so that record data is formatted and displayed in field columns as defined by the record's assigned record type definition.

Once the SDE window view has been opened, the data format can be changed so that the record data is displayed in unformatted character or hex using the SDE **FORMAT** CLI command with parameters CHAR or HEX respectively.

Unformatted record data is displayed as a single character field of length equal to the record length and with field name "Record". Interpretation of the record data based on its assigned record type definition is bypassed.

#### Notes:

1. FORMAT HEX is equivalent to FORMAT CHAR with **HEX** set on.
2. If the record data contains non-printable characters, then the CHAR representation of the data is non-enterable. This protects the user from accidental update of the non-printable text that would otherwise occur as a result of 3270 I/O. Update of record data may still be achieved by updating the HEX representation of the text or using the CHANGE command.

## Displaying a Record in Single Record View

A single record view arranges the fields vertically with the field name and formatted data type on the left of the display and the associated field data on the right.

Once the SDE window view has been opened, a single record view of a data record in the current format (TABLE, CHAR or HEX) can be achieved using the SDE **ZOOM** CLI command.

ZOOM operates on the **focus line** and so, having entered ZOOM at the SDE window command prompt, the cursor should be positioned on the required record before executing the command. Alternatively, assign ZOOM to a PFKey. As distributed, <PF2> is assigned to the SDE REXX macro, SDEZoomW, which opens a new window to display the single record view.

Only records that are visible may be viewed in the single record view (i.e. records that are flagged as being NOTSELECTED, SUPPRESSED or EXCLUDED cannot be viewed in this format.) Using **LEFT** and **RIGHT** commands (which, by default, are assigned to <PF10> and <PF11> respectively), the user can scroll to the previous and next visible record in the SDE edit session.

```

CBLi for TSO 1.5B - Build=200804231235 OpSys=z/OS 1.6.0 User=NBJ2
File List Utilities System Window SwapList Help
CBLi - Edit NBJ2.CBLI.SDE.SAMP.VAR(DATSALES) using NBJ2.CBLI.SDO(COBSALES)
File Edit Actions Options Window Help ? Sv ToF BoF wS wR Pfx < > Scroll> Csr
Command>
Record>
Record type: REC-CUST F(192)
Field Data
1 REC-CUST AN 1:192
2 CUST-ID FB 1:2 9156
2 PASS AN 3:15 78fj2foa
2 LASTNAME AN 18:15 Ramstein
2 FIRSTNAME AN 33:15 Hans
2 COUNTRY AN 48:2 UK
2 POSTCODE AN 50:12 DN4 9BR
2 CITY AN 62:15 DONCASTER
2 HOUSE FB 77:2 124
2 STREET AN 79:25 Springwell Lane
2 EMAIL AN 104:35 hansrams@abc.co.uk
2 PHONE AN 139:25 01302-619928
2 MOBILE AN 164:25 07941-922766
2 BALANCE PD 189:4 -230.48

Se Line=1 Col=1 Alt=0,0;0 Size=63 Recl=16384 Fmt=V Files=2 View
MA b 05/011

```

Figure 5. SDE Formatted Single Record View.

#### Notes:

1. An unformatted (FORMAT CHAR or HEX) in single record view is displayed with a single field name of "Record" with all the record's data displayed to the right of the field name.
2. Executing ZOOM in a FORMAT TABLE view is equivalent to executing the FORMAT SINGLE command.

3. Executing ZOOM when already in single record view, will switch back to a multiple record view in the prevailing (TABLE, CHAR or HEX) format.

## Displaying Hexadecimal Representation of Record Data

FORMAT HEX displays the unformatted record's data in both character and up/down hexadecimal.

Alternatively, the character and up/down hexadecimal representation may be applied to all fields within a formatted record (FORMAT TABLE or SINGLE) using the SDE **HEX** CLI command.

```

-CBli for TSO 1.5B - Build=200804281102 OpSys=z/OS 1.6.0 User=NBj2
File List Utilities System Window SwapList Help
CBle - Edit CBL.DIST.CBLI.SDE.SAMP.VAR(DATTYPES) using NBj2.CBLI.SDO(COBT
File Edit Actions Options Window Help ? Sv ToF BoF wS wR Pfx < >
Command> HEX ON Scroll> Csr
*** Top of Data ***
Record type: DATA-TYPES-1 V(39,49)
MYREC MYBINTEGER MYCHARACTER MYDECIMAL MYFIXED MYFLOAT MYHE
#2 #3 #4 #5 #6 #7 #8
AN 1:1 FB 2:2 AN 4:4 PD 8:4 FB 12:4 FP 16:4 AN 20:
- <-> <---> <----+--> <-----+-----1> <-----+-----1-----> <--->
000001 1 9 abcd 1234.56 1234567.89 +0.10000000E+02 ..~.
F 00 8888 0246 05C1 4A00 79BD
1 09 1234 135C 7BD5 1000 8ACE
000002 1 -1 abcd -1234.56 -1234567.89 +0.10000000E+01 1...
F FF 8888 0246 FA3E 4100 F246
1 FF 1234 135D 842B 1000 1357
Record type: DATA-TYPES-2 V(42,46)
MYREC MYZONED MYINTEGHER MYXVARCHAR MYLL MYSTRCH(1) MYSTRCH(2) MYSTRC
#2 #3 #4 #5 #7 #9 #9 #9
AN 1:1 ZD 2:4 FB 6:2 AN 8:8 FB 16:2 AN 18:1 AN 19:1 AN 20:
- <---+> <---> <---+--> <---> - - -
000003 2 12.34 8 ABCDEFGH *****
F FFFF 00 CCCCCCCC 00 0 0 0
2 1234 08 12345678 08 0 0 0
000004 2 12.34 8 ABCDEFGH 7 a b c
F FFFF 00 CCCCCCCC 00 8 8 8
2 1234 08 12345678 07 1 2 3
Record type: DATA-TYPES-1 V(39,49)
MYREC MYBINTEGER MYCHARACTER MYDECIMAL MYFIXED MYFLOAT MYHE
#2 #3 #4 #5 #6 #7 #8
AN 1:1 FB 2:2 AN 4:4 PD 8:4 FB 12:4 FP 16:4 AN 20:
- <-> <---> <----+--> <-----+-----1> <-----+-----1-----> <--->
000005 1 6 abcd 1234.56 8090640.56 +0.95367431E-06 ....
F 00 8888 0246 3357 4001 1357
1 06 1234 135C 0968 0000 2468
Se Line=0 Col=1 Alt=4,4;0 Size=6 Recl=16384 Fmt=V Files=1 Views
MA b 05/017

```

Figure 6. SDE Formatted Multiple Record View with HEX ON.

## Excluding Records from the Display

By default, the contents of records that are flagged as NOT SELECTED or SUPPRESSED are not displayed in the SDE window view.

Further filtering of the displayed records can be achieved by setting the EXCLUDED flag on those records that are to be excluded from the display. If **SHADOW EXCLUDED** is set on (the default), consecutive excluded records that are assigned the same record type are represented by a single shadow line of type "EXCLUDED".

The **Prefix Area Command** X(n) or XX may be used to exclude a record (and optionally n-1 records that follow) or a block of records, regardless of record type.

Alternatively, the **EXCLUDE** SDE CLI command may be used to exclude records of a specific record type that fall within a range of records and also match a supplied search string. Records that are assigned a different record type are unaffected. EXCLUDE is similar to the ISPF style text edit command with additional syntax to nominate those columns to be searched.

e.g. Exclude all records assigned the same record type as the focus record and contain the character string "London" (any case) within either of the fields named "X City" or "X County".

EXCLUDE ALL 'London' (X City, X County)

```

-CBLi for TSO 1.5B - Build=200804231235 OpSys=z/OS 1.6.0 User=NBj2
File List Utilities System Window Help
CBLi - Edit NBJ2.CBLI.SDE.SAMP.VAR(DATSALES) using NBJ2.CBLI.SDO(COBSALES)
File Edit Actions Options Window Help ? Sv ToF BoF wS wR Pfx < > Scroll> Csr
Command>
Record type: REC-CUST F(192)
CUST-ID PASS LASTNAME FIRSTNAME COUNTRY POSTCODE
#2 #3 #4 #5 #6 #7
FB 1:2 AN 3:15 AN 18:15 AN 33:15 AN 48:2 AN 50:12
<---> <---+---1---> <---+---1---> <---+---1---> <> <---+---
000001 9156 78fj2foa Ramstein Hans UK DN4 9BR
000002 ----- 2 line(s) excluded: record type REC-CUST -----

Record type: REC-CARD F(56)
CUST-ID SEQ CR-OR-DR COMPANY CARD-NUMBER NAME
#2 #3 #5 #6 #7 #8
FB 1:2 FB 3:2 AN 5:1 AN 6:7 PD 13:9 AN 22:25
<---> <-> <---+---1---> <---+---1---> <---+---1---> <---+---2---
000004 1124 0 D VISA 8754875676474656 CARL SLAIN
000005 9156 0 D DELTA 1282937860235632 MR HANS R RAMSTEIN
000009 1564 1 C DELT 6754875676474656 MR C SLAIN
000011 ----- 1 line(s) excluded: record type REC-CUST -----
000012 ----- 3 line(s) excluded: record type REC-CARD -----
000028 ----- 1 line(s) excluded: record type REC-CUST -----
000029 2991 0 X Aqua 8627368960235435 Jacques Cousteau
000030 6792 6 W VISA 9463829326644146 BRUCE W HOWARD

Record type: REC-CUST F(192)
CUST-ID PASS LASTNAME FIRSTNAME COUNTRY POSTCODE
#2 #3 #4 #5 #6 #7
FB 1:2 AN 3:15 AN 18:15 AN 33:15 AN 48:2 AN 50:12
<---> <---+---1---> <---+---1---> <---+---1---> <> <---+---
000044 6750 arlUoAy Howard Bruce UK UB2 1NG

Record type: REC-CARD F(56)
CUST-ID SEQ CR-OR-DR COMPANY CARD-NUMBER NAME
#2 #3 #5 #6 #7 #8
FB 1:2 FB 3:2 AN 5:1 AN 6:7 PD 13:9 AN 22:25
<---> <-> <---+---1---> <---+---1---> <---+---1---> <---+---2---
000045 1472 5 D VISA 1282937860235632 Tayler Martinez
000046 9804 6 D MAST 2637970469695876 Mr JOseph B Jones
000047 9316 5 C DELTA 1982894749696023 Taylor martines
Se Line=1 Col=1 Alt=0,0;0 Size=63 Recl=16384 Fmt=V Files=1 View
MA b 01/001

```

Figure 7. SDE Excluded Records.

The **FLIP** SDE CLI Command will flip the EXCLUDED flag on all records of a specific record type so that visible records are excluded and excluded records are brought back into view.

Excluded records may also be brought back into view using the Prefix Area Commands, F(n) and L(n). These remove the EXCLUDED flag from the First or Last records respectively (and optionally n-1 records that follow/precede) that belong to a block of excluded records, regardless of record type.

Alternatively, the **RESET EXCLUDED** SDE CLI Command may be executed to remove the EXCLUDED flag from All excluded records of a specific record type.

SDE commands that perform searches and selection on records may also alter the EXCLUDED flag setting of individual records in the display. See **"Searching Data Records"** and SDE CLI commands **FIND**, **LOCATE**, **WHERE**, **MORE** and **LESS**.

## Displaying Selected Columns

By default, all named fields belonging to records in the SDE window view are displayed in the order in which they occur within the data records.

Once the SDE window view has been opened, the SDE **SELECT** CLI command may be used to re-order and/or restrict the number of fields displayed in records that are assigned the specified record type. In addition to this, the HOLD parameter may be used to indicate that a specified field, and all fields before it, are to be held when scrolling the display of the record data left and right.

The supplied SDE macro "SDESEL" (assigned to PF14 by default) may be executed to generate a SELECT command in a temporary CMX file referencing all fields in the focus record type or all record types in the SDO. This is particularly useful when dealing with record types with a large number of fields required for display. It allows the user to easily tailor the presence and order of fields within the SELECT command without having to re-type the field names or references.

e.g. To re-order the occurrence of fields in records of record type "REC-CUST" and hold the first two fields, execute the following. Note that "\*" (asterisk) represent all remaining fields, in the order in which they occur within the record, that have not already been specified on the SELECT command.

```
SELECT FIRSTNAME, LASTNAME HOLD, EMAIL, * FROM REC-CUST
```

```

-CBLi for TSO 1.5B - Build=200804231235 OpSys=z/OS 1.6.0 User=NBj2
File List Utilities System Window Help
CBLi - Edit NBJ2.CBLI.SDE.SAMP.VAR(DATSALES) using NBJ2.CBLI.SDO(COBSALES)
File Edit Actions Options Window Help ? Sv ToF BoF wS wR Pfx < >
Command> SELECT FIRSTNAME, LASTNAME HOLD, EMAIL, * FROM REC-CUST Scroll> Csr
Record type: REC-CUST F(192)
FIRSTNAME LASTNAME EMAIL
#5 #4 #11
AN 33:15 AN 18:15 AN 104:35
<---+---1---> <---+---1---> <---+---1---+---2---+---3--->
000001 Hans Ramstein hansrams@abc.co.uk
000002 Denise Richards DRich153@fsnet.com
000003 Mike Hill MikeH@proveit.com

Record type: REC-ORDER V(86,106)
CUST-ID ORDER-REF QTY ITEM-CODE UNIT-PRICE DELIVERY ORDER-DATE
#2 #3 #4 #5 #6 #7 #8
FB 1:2 FB 3:4 FB 7:2 FB 9:2 PD 11:4 PD 15:3 PD 18:5
<---> <---+---> <---> <---> <---+---> <---+---> <---+--->
000006 9156 557833807 7 18385 84.82 4.79 20051022
000007 9156 696534904 1 19692 128.03 7.94 20070823
000011 ----- 1 line(s) excluded: record type REC-CUST -----
000015 1820 756551020 8 32417 39.99 7.85 20060625
000016 4596 989428617 10 25366 142.27 5.33 20060503
000017 1812 898652388 4 22235 202.38 3.23 20050917
000018 1722 866928690 2 29315 189.77 10.56 20060715
000019 8997 270547901 7 14136 210.56 4.29 20070617
000020 2967 360725448 3 17978 -40.72 -4.45 20070207
000021 9156 397432931 9 20943 125.15 8.97 20050111
000022 8616 934379132 7 25031 16.48 7.82 20070129

Record type: REC-CUST F(192)
FIRSTNAME LASTNAME EMAIL
#5 #4 #11
AN 33:15 AN 18:15 AN 104:35
<---+---1---> <---+---1---> <---+---1---+---2---+---3--->
000028 Joseph Jones joseph.jones@seventyseven.ca
000031 ----- 8 line(s) excluded: record type REC-ORDER -----
000044 Bruce Howard brucethoward@fsnet.com

Record type: REC-ORDER V(86,106)
CUST-ID ORDER-REF QTY ITEM-CODE UNIT-PRICE DELIVERY ORDER-DATE
Se Line=1 Col=1 Alt=0,0;0 Size=63 Recl=16384 Fmt=V Files=1 View
MA b 05/066

```

Figure 8. SDE Selected Field Columns.

By default, fields that are not assigned a field name (e.g. COBOL FILLER fields) are not displayed. In order to display these fields, use the **SET UNNAMED** SDE CLI command to set the display unnamed fields on.

## Displaying the Record Headers

In a multiple record view, all 5 field column header lines are displayed before each group of records of the same record type.

These are the **Record Type**, **Field Name**, **Field Reference**, **Field Type** and **Scale** header lines. These are discussed in detail under **SDE Window Views**.

The **Field Reference**, **Field Type** and **Scale** lines may be set on or off using the **SET REFERENCE**, **SET TYPE** and **SET SCALE** SDE CLI commands respectively.

```

-CBLi for TSO 1.5B - Build=200804251031 OpSys=z/OS 1.6.0 User=NBj2
File List Utilities System Window SwapList Help
CBLi - Edit NBj2.CBLI.SDE.SAMP.VAR(DATSALES) using NBj2.CBLI.SDO(COBSALES)
File Edit Actions Options Window Help ? Sv ToF BoF wS wR Pfx < >
Command> SET REF OFF; SET TYPE OFF Scroll> Csr
000000 *** Top of Data ***
Record type: REC-CUST F(192)
CUST-ID PASS LASTNAME FIRSTNAME COUNTRY POSTCODE
<---> <---+---1---> <---+---1---> <---+---1---> <> <---+---
000001 9156 78fj2foa Ramstein Hans UK DN4 9BR
000002 7037 chico Richards Denise UK CF31 1BX
000003 6712 benj Hill Mike US 75204

Record type: REC-ORDER V(86,106)
CUST-ID ORDER-REF QTY ITEM-CODE UNIT-PRICE DELIVERY ORDER-DATE
<---> <---+---> <---> <---> <---+---> <---+---> <---+--->
000006 9156 557833807 7 18385 84.82 4.79 20051022
000007 9156 696534904 1 19692 128.03 7.94 20070823
000015 1820 756551020 8 32417 39.99 7.85 20060625
000016 4596 989428617 10 25366 142.27 5.33 20060503
000017 1812 898652388 4 22235 202.38 3.23 20050917
000018 1722 866928690 2 29315 189.77 10.56 20060715
000019 8997 270547901 7 14136 210.56 4.29 20070617
000020 2967 360725448 3 17978 -40.72 -4.45 20070207
000021 9156 397432931 9 20943 125.15 8.97 20050111
000022 8616 934379132 7 25031 16.48 7.82 20070129

Record type: REC-CUST F(192)
CUST-ID PASS LASTNAME FIRSTNAME COUNTRY POSTCODE
<---> <---+---1---> <---+---1---> <---+---1---> <> <---+---
000028 4622 revelator Jones Joseph CA H3A 3E5

Record type: REC-ORDER V(86,106)
CUST-ID ORDER-REF QTY ITEM-CODE UNIT-PRICE DELIVERY ORDER-DATE
<---> <---+---> <---> <---> <---+---> <---+---> <---+--->
000031 4169 497047541 6 16967 35.28 3.06 20060315
000032 9156 122075050 9 14757 190.37 2.01 20070516
000033 2532 819485175 8 20508 -117.70 -5.84 20070610
000034 1427 104613891 7 12951 168.96 3.03 20071025
000035 4484 938490727 10 29782 93.75 7.86 20070512
000036 9156 512510042 9 16036 57.24 6.82 20060827
000037 8348 276846581 8 14305 106.95 3.82 20060624

Se Line=0 Col=1 Alt=0,0;0 Size=63 Recl=16384 Fmt=V Files=1 View
MA b 05/038

```

Figure 9. SDE Suppressed Header Lines.

## Searching Data Records

SDE supports powerful search commands that allow the user to find and create subsets of records based on specified search criteria. (FIND, LOCATE, WHERE, MORE and LESS)

These are covered by the following:

- Finding Record Data by Search String
- Locating Records Using a Search Expression
- Creating a Subset of Records for Display

## Finding Record Data by Search String

Like EXCLUDE (see *"Excluding Records from the Display"*), the FIND and RFIND SDE CLI commands are similar to the ISPF style text edit commands with additional syntax to nominate those columns to be searched.

In FORMAT TABLE or SINGLE view, the search scope for FIND is supplied as a list of individual fields and/or field ranges that occur in records that match a specific record type. Records that are assigned a different record type and fields that have been removed from view by a SELECT command, are automatically excluded from the search.

e.g. To find the first occurrence of the mixed case string 'James' within the character fields "LASTNAME" or "FIRSTNAME" (field reference #4) which belong to records of the same record type as the focus record, then enter the following from the SDE command prompt:

```
FIND FIRST C'James' (LASTNAME, #4)
```

Un-quoted, numeric search strings are treated as numeric values when testing non-character data fields, i.e. the value of the numeric field data must be arithmetically equal to the search value. When testing character fields, numeric search strings are treated as character data.



e.g. To find all occurrences of the value "1" within any numeric field or the character string "1" in any character field within a record of the same record type as the focus record, then enter the following from the SDE command prompt:

```
FIND ALL 1 #ALL
```

If the specified occurrence (ALL, FIRST, LAST, NEXT or PREV) of the search string or numeric value is found, then:

1. If the record is EXCLUDED, the EXCLUDED flag is set off so that the record is brought back into view.
2. The cursor is positioned at the beginning of the string or numeric field.
3. If necessary, scrolling occurs to display the found data.

```

-CBLi for TSO 1.5B - Build=200804281102 OpSys=z/OS 1.6.0 User=NBJ2
File List Utilities System Window SwapList Help
-CBLi - Edit CBL.DIST.CBLI.SDE.SAMP.VAR(DATTYPES) using NBJ2.CBLI.SDO(COBT
File Edit Actions Options Window Help ? Sv ToF BoF wS wR Pfx < >
Command> FIND ALL 1 #ALL
SDE186I 6 occurrences of string "1" were found in records of type DATA-TYPES-1.
MYREC MYBINTEG MYCHARACT MYDECIMAL MYFIXED MYFLOAT MYHE
#2 #3 #4 #5 #6 #7 #8
AN 1:1 FB 2:2 AN 4:4 PD 8:4 FB 12:4 FP 16:4 AN 20:
- <> <--> <---+---> <---+---1> <---+---1---> <-->
000001 1 9 abcd 1234.56 1234567.89 +0.10000000E+02 ..~.
000002 1 -1 abcd -1234.56 -1234567.89 +0.10000000E+01 1...

Record type: DATA-TYPES-2 V(42,46)
MYREC MYZONED MYINTEGER MYXVARCH MYLL MYSTRCH(1) MYSTRCH(2) MYSTRC
#2 #3 #4 #5 #7 #9 #9 #9
AN 1:1 ZD 2:4 FB 6:2 AN 8:8 FB 16:2 AN 18:1 AN 19:1 AN 20:
- <---+> <---+---> <---+---> - - -
000003 2 12.34 8 ABCDEFGH ***** 7 a b c
000004 2 12.34 8 ABCDEFGH 7 a b c

Record type: DATA-TYPES-1 V(39,49)
MYREC MYBINTEG MYCHARACT MYDECIMAL MYFIXED MYFLOAT MYHE
#2 #3 #4 #5 #6 #7 #8
AN 1:1 FB 2:2 AN 4:4 PD 8:4 FB 12:4 FP 16:4 AN 20:
- <> <--> <---+---> <---+---1> <---+---1---> <-->
000005 1 6 abcd 1234.56 8090640.56 +0.95367431E-06 ....
000006 1 -3 abcd 1234.56 8090640.56 +0.83378875E-56 ....
000007 *** End of Data ***

Se Line=1 Col=1 Alt=0,0;0 Size=6 Recl=16384 Fmt=V Files=1 Views
MA b 11/009

```

Figure 10. SDE FIND Value.

## Locating Records Using a Search Expression

The **LOCATE** SDE CLI command will scroll the display to make a specific or relative record number the current (first visible) line of the window view. Alternatively, a record of a specific record type that matches the search criteria is made the current line of the display.

LOCATE supports an SQL like *<where\_clause>* that allows for more sophisticated search criteria than the search for equality supported by FIND. The *<where\_clause>* expression supports:

1. Field versus field compare.
2. Brackets "(" and ")" and logical operators "&" (AND), "|" (OR).
3. Relational operators "=", "<", ">=", "\=", etc.

e.g. Locate the last record with the same record type as the focus record, where the field "UNIT-PRICE" has a value greater than 100 **and** the field "DELIVERY" has a value less than 10.

```
LOCATE LAST UNIT-PRICE > 100 & DELIVERY < 10
```

## Creating a Subset of Records for Display

Using a *<where\_clause>* expression, records can be excluded from the display to create a subset of all records of a particular record type that also satisfy the specified *<where\_clause>*.

The **WHERE** SDE CLI command filters all records of a specific record type so that only those records that match the supplied *<where\_clause>* are displayed. Records that **do not** match the expression are excluded whereas records that are assigned a different record type are unaffected.

e.g. Display only those records assigned the same record type as the focus record and where the contents of field "DESPATCH\_DATE" is less than the contents of field "ORDER\_DATE" **or** either of the following are **not** true:

1. Value of numeric field "ORDER\_VALUE" is greater than or equal to 256.55.
2. Value of numeric field "ORDER\_BALANCE" is equal to 0.

```
WHERE (DESPATCH_DATE < ORDER_DATE) | \ (ORDER_VALUE >= 256.55 & ORDER_BALANCE = 0)
```

The **LESS** SDE CLI command allows the user to filter the subset further so that only visible records of the specific record type are excluded if they **do not** satisfy the supplied *<where\_clause>*. Records that are already excluded or assigned a different record type are unaffected.

e.g. Exclude only records that are in view that are assigned the same record type as the focus record and where the contents of field "ORDER\_DATE" is greater than or equal to '2007/11/28'.

```
LESS ORDER_DATE >= '2007/11/28'
```

Similarly, the **MORE** SDE CLI command allows the user to add the subset of records so that excluded records of the specific record type are made visible if they satisfy the supplied *<where\_clause>*. Records that are already visible or assigned a different record type are unaffected.

e.g. Bring back into view only excluded records that are assigned the same record type as the focus record and where the character field "BACKUP\_VOL" **does not** begin with the string "Z16".

```
MORE BACKUP_VOL \>> C'Z16'
```

```

-CBLi for TSO 1.5B - Build=200804281102 OpSys=z/OS 1.6.0 User=NBj2
File List Utilities System Window SwapList Help
-CBLi - Edit NBj2.CBLI.SDE.SAMP.VAR(DATSALES) using NBj2.CBLI.SDO(COBSALES)
File Edit Actions Options Window Help ? Sv ToF BoF wS wR Pfx < >
Command> WHERE #2 > 3000 & COMPANY \>> 'DELT' ToF BoF wS wR Pfx < >
SDE1781 6 lines selected by WHERE #2 > 3000 & COMPANY \>> 'DELT'. Scroll> Csr
CUST-ID SEQ CR-OR-DR COMPANY CARD-NUMBER NAME
#2 #3 #5 #6 #7 #8
FB 1:2 FB 3:2 AN 5:1 AN 6:7 PD 13:9 AN 22:25
<---> <-> <---+---1---+---> <---+---1---+---2---
000012 9804 2 C Aqua 2890100161893840 Joseph B Jones
000013 9928 2 C MAST 8965229969967546 Mr Bruce W Howard
000014 6239 4 D VISA 1098717650442152 Msr J Cousteau
000029 ----- 1 line(s) excluded: record type REC-CARD -----
000030 6792 6 W VISA 9463829326644146 BRUCE W HOWARD
000045 ----- 1 line(s) excluded: record type REC-CARD -----
000046 9804 6 D MAST 2637970469695876 Mr JOeseph B jones
000047 ----- 1 line(s) excluded: record type REC-CARD -----
000048 9156 1 C MAST 7678746370682730 Mr H R Ramstein
000049 ----- 1 line(s) excluded: record type REC-CARD -----
000064 *** End of Data ***
  
```

Se Line=12 Col=1 Alt=0,0;0 Size=63 Recl=16384 Fmt=V Files=1 Vie

MA b 05/011

Figure 11. SDE Subset Using WHERE.



## Editing Data

Structured data editing in CBLi supports the standard functions required for basic data edit with additional consideration given to specification, recognition and preservation of the data structure.

These basic features include:

- Typing New Record Data
- Find and Replace Data
- Inserting Records
- Deleting Records
- Moving Records
- Copying Records
- Duplicating Records
- Undo and Redo Changes

In order to perform edit tasks in SDE, the SDE window view must have been opened for EDIT, not BROWSE.

Changes to edited text are saved in storage. Unless **QQUIT** is used, the user will be prompted to save the changes to disk when the file is closed. The **SAVE** SDE CLI command should be used to save the changes without closing the file.

## Typing New Record Data

Simply position the cursor within the record at which text is to be entered and begin typing. Existing data may be overtyped and deleted, or new text may be inserted within the existing record data.

Beware when editing unformatted data (FORMAT CHAR or HEX) that the integrity of the formatted data is maintained. e.g. Inserting or deleting data so that it changes the length of the record or causes data to overlap a formatted field boundary, will corrupt the record.

In FORMAT SINGLE or TABLE, text entered within a field is restricted by the size of that field. On encountering the end of a field, the cursor is automatically positioned at the next field in the display. Therefore, users should beware not to accidentally overtype data in subsequent fields. Note, however, that inserting text before existing data in a field will not propagate that data into subsequent fields in the display.

If character field data contains non-printable text, then the CHAR representation of the data is non-enterable. This protects the user from accidental update of the non-printable text that would otherwise occur as a result of 3270 I/O. Update of the field's data may still be achieved by overtyping the HEX representation of the text or using the CHANGE command.

On inserting data in the variable length character field of a formatted record, the new length of the field data will automatically be updated in the accompanying length field. Similarly, any alteration to the value in the length field will automatically be reflected in the variable length character data (i.e. the data will be truncated or padded with blanks.)

On entering data in formatted numeric fields, the following occurs:

1. The field is scanned from left to right and leading blanks ignored.
2. The numeric data is identified as being the non-blank text terminated by the end of the field or a blank character.
3. Data within the field that is to the right of a terminating blank is ignored.
4. The numeric data is interpreted and validated to establish whether it complies with the field's defined data type, precision and scale.

If an invalid value is entered in a numeric field then a warning popup window is opened, giving the reason for rejecting the value and prompting the user to either enter a valid numeric value or discard the change for that particular field. This occurs for each field that has been updated with an invalid numeric value since the last transaction.

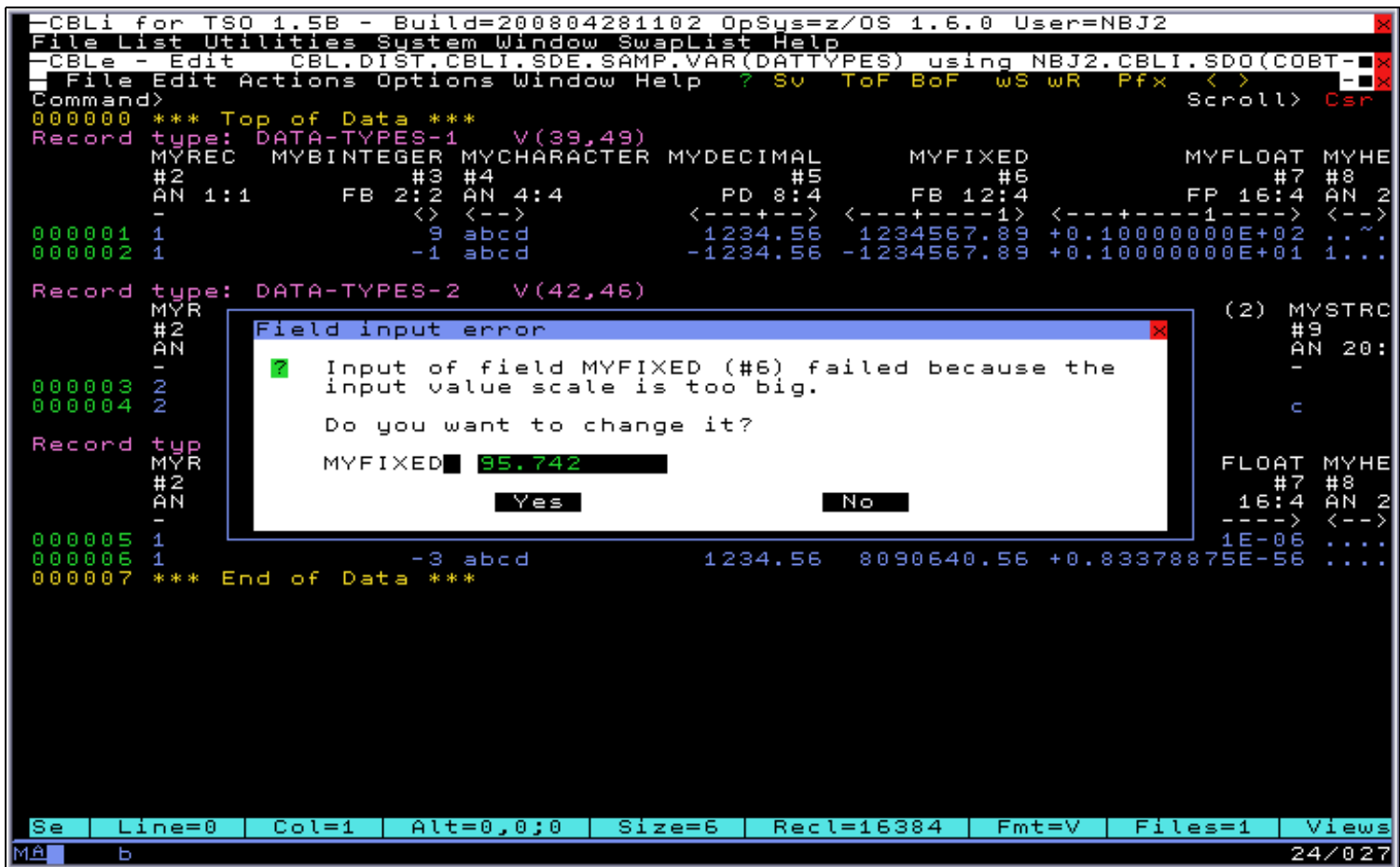


Figure 12. SDE Field Input Error.

## Find and Replace Data

The **CHANGE** SDE CLI command allows users to selectively change occurrences of a supplied character string or numeric value to another string/value, for records of a specific record type.

The **CHANGE** command will search for the first specified string/value using the same specifications used for a **FIND** command (see *"Finding Record Data by Search String"*), then replace it with the second specified string/value.

e.g. To find and change the next occurrence of the value "1234.56" to "9876.54" in the numeric field "MYDECIMAL", in records of the same record type as the focus record, enter the following:

```
CHANGE 1234.56 9876.54 (MYDECIMAL)
```

To find and change the next occurrence of the search string/value execute the **RCHANGE** SDE CLI command (assigned to PF6 by default). To find and optionally change the next occurrence of the search string/value execute a combination of **RFIND** (assigned to PF5 by default) and then **RCHANGE**.

**CHANGE ALL** will change all occurrences of the string/value in visible or **EXCLUDED** records of the default record type.

## Inserting Records

New records of a specific record type may be inserted before or after existing records using any display format (**TABLE**, **SINGLE**, **CHAR** or **HEX**).

The **Prefix Area Command** **I(n)** may be used to insert one or more records with a record type that matches that of the line on which the command is entered.

Alternatively, the **INSERT** SDE command will insert one or more new lines of the specified record type following the focus line. In **TABLE** or **SINGLE** view, **INSERT** also supports specification of one or more field columns and the default values to be inserted in each field.

e.g. Insert 3 new records of record type "DATA-TYPES-1" containing some initial values in fields "MYCHARACTER", "MYDECIMAL" and "MYFLOAT".

```
INSERT DATA-TYPES-1 (MYCHARACTER, MYDECIMAL, MYFLOAT) VALUE("TEMP", 3569.93, 23.0e-12) 3
```

When inserting new lines, unformatted records are initialised to blanks of length equal to the LRECL value for FIXED RECFM records or of length 1 for VARIABLE RECFM records. For FORMAT SINGLE or TABLE, then, unless otherwise specified, fields are initialised as follow:

- Numeric fields are set to 0.
- Character fields are set to blanks.

```

-CBLi for TSO 1.5B - Build=200804281102 OpSys=z/OS 1.6.0 User=NBj2
File List Utilities System Window SwapList Help
CBLi - Edit CBL.DIST.CBLI.SDE.SAMP.VAR(DATTYPES) using NBj2.CBLI.SDO(COBT
File Edit Actions Options Window Help ? Sv ToF BoF wS wR Pfx < >
Command> INSERT DATA-TYPES-1 (MYCHARACTER, MYDECIMAL, MYFLOAT) Scroll> Csr
> VALUE("TEMP", 3569.93, 23.0e-12) 3
Record type: DATA-TYPES-1 V(39,49)
MYREC MYBINTEGER MYCHARACTER MYDECIMAL MYFIXED MYFLOAT MYHE
#2 #3 #4 #5 #6 #7 #8
AN 1:1 FB 2:2 AN 4:4 PD 8:4 FB 12:4 FP 16:4 AN 2
- <> <--> <---+---> <---+---1> <---+---1---> <--->
000001 1 9 abcd 1234.56 1234567.89 +0.10000000E+02 ..~.
000002 1 0 TEMP 3569.93 0.00 +0.23999999E-10 ..~.
000003 1 0 TEMP 3569.93 0.00 +0.23999999E-10 ..~.
000004 1 0 TEMP 3569.93 0.00 +0.23999999E-10 ..~.
000005 1 -1 abcd -1234.56 -1234567.89 +0.10000000E+01 1...

Record type: DATA-TYPES-2 V(42,46)
MYREC MYZONED MYBINTEGER MYXVARCHAR MYLL MYSTRCH(1) MYSTRCH(2) MYSTRCH(3)
#2 #3 #4 #5 #6 #7 #8 #9
AN 1:1 ZD 2:4 FB 6:2 AN 8:8 FB 16:2 AN 18:1 AN 19:1 AN 20:
- <---+> <---> <---+---> <---> - - -
000006 2 12.34 8 ABCDEFGH ***** 7 a b c
000007 2 12.34 8 ABCDEFGH ***** 7 a b c

Record type: DATA-TYPES-1 V(39,49)
MYREC MYBINTEGER MYCHARACTER MYDECIMAL MYFIXED MYFLOAT MYHE
#2 #3 #4 #5 #6 #7 #8
AN 1:1 FB 2:2 AN 4:4 PD 8:4 FB 12:4 FP 16:4 AN 2
- <> <--> <---+---> <---+---1> <---+---1---> <--->
000008 1 6 abcd 1234.56 8090640.56 +0.95367431E-06 ....
000009 1 -3 abcd 1234.56 8090640.56 +0.83378875E-56 ....
000010 *** End of Data ***

Se Line=1 Col=1 Alt=1,1;1 Size=9 Recl=16384 Fmt=V Files=1 Views
MA b 13/009

```

Figure 13. SDE Insert Records.

Following an insert, the cursor is placed on the first character of the first field in the display.

## Deleting Records

The **Prefix Area Command** D(n) or DD may be used to delete a record (and optionally n-1 records that follow) or a block of records, regardless of record type.

Alternatively, the **DELETE** SDE command may be used to delete one or more EXCLUDED and/or visible records of any record type that fall within a specified range of lines. DELETE operates on the record occupying the focus line and records that follow.

e.g. Delete all excluded lines that fall within a range of lines denoted by label names .FTXT and .FLST.

```
DELETE ALL EX .FTXT .FLST
```

Records that are SUPPRESSED or NOT SELECTED are unaffected by record delete operations.

## Moving Records

The **Prefix Area Command** M(n) or MM is used to move a record (and optionally n-1 records that follow) or a block of records, regardless of record type.

Having marked a record or block of records to be moved, position the cursor in the prefix area of the record where the records are to be moved and enter A (After) or B (Before). The marked record or block of records are moved after or before the target record respectively.

If the target destination of the move operation is within the block of records marked for move, then an error message is returned.

Following a move, the cursor is placed on the first character of the first record moved.

Only records that are visible or excluded may be moved. Records that are SUPPRESSED or NOT SELECTED can not be moved.

## Copying Records

The **Prefix Area Command** C(n) or CC is used to copy a record (and optionally n-1 records that follow) or a block of records, regardless of record type.

Having marked a record or block of records to be copied, position the cursor in the prefix area of the record where the records are to be copied and enter A (After) or B (Before). The marked record or block of records are copied after or before the target record respectively.

If the target destination of the copy operation is within the block of records marked for copy, then an error message is returned.

Following a copy, the cursor is placed on the first character of the first record copied.

Only records that are visible or excluded may be copied. Records that are SUPPRESSED or NOT SELECTED can not be copied.

## Duplicating Records

A single record or group of records may be duplicated a specified number of times using the Replicate **Prefix Area Command** R(n) or RR(n). Where RR is entered in the prefix areas of different lines, a block of records will be duplicated regardless of the record type of records within the marked block.

The numeric following R or the first RR marking a block of records, specifies the number of times the record(s) will be duplicated.

The target line for duplicated record(s) is immediately following the record or record block marked for duplication.

Alternatively, the **DUPLICATE** SDE CLI command may be used to duplicate the focus line a specific number of times. e.g.

```
DUPLICATE 3
```

Following the duplicate operation, the cursor is placed on the first character of the first duplicated record.

Only records that are visible or excluded may be duplicated. Records that are SUPPRESSED or NOT SELECTED can not be duplicated.

## Undo and Redo Changes

A new change level is added for each field whose contents have changed since the last 3270 transaction.

Having made changes to field data during the edit session, the **UNDO** SDE CLI command (assigned to PF22 by default) may be used to undo the change represented by the last change level. Repeatedly executing UNDO will undo each previous change in the change level stack.

Where changes have been undone, they may be re-applied, one change level at a time, using the **REDO** SDE CLI command (assigned to PF23 by default.)

The number of change levels that may be undone is represented by the third number in the "Alt=n,n;m" alteration count displayed in the status line of the MDI parent window. If this number has an "\*" (asterisk) appended, then this indicates that change levels are available to be re-applied with REDO.

**Note:** If updates are made to field data after undoing changes, then those change levels can no longer be re-applied and the trailing "\*" on the alteration count is dropped.

The maximum number of change levels that can be stored for each individual file edited with SDE is set by the **SET UNDOING** SDE CLI command. This command also controls the maximum amount of storage that may be allocated for storing information required by SDE to UNDO and REDO a change represented by a change level.

Once the maximum amount of defined change levels or change storage is reached, SDE silently discards as much of the oldest change level information as is required to continue editing data. Consequently, changes represented by these discarded change levels can no longer be undone.

# SDE Concepts

Although CBLi SDE runs within the CBLi text edit environment, it is a much more sophisticated method of editing data.

Presentation and edit of record data within SDE must adhere strictly to precedents defined by the associated file structure. In order to do this, SDE introduces certain concepts and terminology that are used in its interpretation of data and are referenced throughout this documentation.

It is recommended that users familiarise themselves with the terms detailed in this section prior to performing any advanced edit operations.

---

## Record Structure Definition

In order to display a file's data records correctly within CBLi SDE, an appropriate data definition must exist that accurately maps all fields within the data records.

For CBLi SDE, a data definition is called the **structure** and must exist in an internal SDE format generated by the **CREATE STRUCTURE** command. Any existing COBOL or PL1 copybooks that map the file's data records may be used to generate the SDE structure.

## Structure Definition Object

The Structure Definition Object (SDO) is an SDE structure which has been loaded into storage in order to map data records within a file. An SDO consists entirely of one or more Record Type Objects and is referenced by its structure name. An SDO structure name is also the data set name allocated when it is saved to disk (Structure Definition File).

An SDO is generated when a structure is created (via **CREATE STRUCTURE**) or when a structure is nominated on an **EDIT** or **BROWSE** command. An SDO is changed if any of its Record Type Objects are updated during the SDE session, in which case the user will be prompted to save the changed SDO to disk.

## Record Type Object

A Record Type Object (RTO) is a single record structure definition within an SDO which contains the following information:

1. Field offset, length and data format of all fields within a data record.
2. Conditional logic which, together with the record's LRECL, will be used to identify the data records to which the RTO will apply.

So that its data can be mapped correctly, each structured record within a data set should have characteristics that match the criteria defined by an RTO within the associated SDO.

Each RTO is referenced by a name which is unique within the SDO. This name is also referred to as the **record type** and is assigned to an RTO when it is defined (**CREATE STRUCTURE**). The RTO record type may be used as a generic description of those data records that are associated with that specific RTO. e.g. Client1 data records are considered to be records that are mapped by an RTO with the name "Client1".

An RTO (and hence the SDO) may be altered (e.g. via the **USE WHEN** command) and saved to a Structure Definition File.

## Structure Definition File

A Structure Definition File (SDF) is a disk file (Sequential data set or PDS/PDSE member) which contains a saved Structure Definition Object.

The SDF data set name is equivalent to the SDO structure name.  
On executing an **EDIT** or **BROWSE** command, if the specified SDO is not already in storage, it gets loaded from the SDF.

If an SDO is altered during an SDE edit session and is not flagged as being temporary, then the user will be prompted to save the SDO to its SDF. This may be done automatically during a **CREATE STRUCTURE** command.

---

## SDE Window Views

An SDE window is an MDI child window (view) that may be opened from within the CBLi Text Editor or SELCOPY Interactive application parent window.

Data within an SDE window is presented as either a multi record or single record view.

- **Multi Record View**

- **Single Record View**

## Multi Record View

An SDE MDI child window containing multiple records displayed horizontally so that each record occupies a single line of the display.

### Field Column-Heading Lines

By default, a group of 5 field column-heading lines are displayed within the window display area immediately above the first data record in view. Unless records associated with a different record type are visible (see **Record Data Display** below), no further groups of column-heading lines are displayed within the current display area view.

```
Record type: AM    F(5700)
AmDATE    AmKElCur AmKElMax AmKLineN AmKLineL    AmKEYC    AmKEY    AmCOUNTY
#2         #3         #5         #6         #7         #8 #9         #11        #12        #13        #14
AN 1:8    AN 9:2    BN 11:1    BN 12:1    BN 13:1    BN 14:1 AN 15:6 AN 21:5 AN 26:7 AN 33:1 AN 34:4
<...+..> <>        <.>        <.>        <.>        <.> <...+> <...>    <...+..> .    <...>
```

Figure 14. SDE Multi-Record Display Header Lines.

#### Header Line 1: Record Type

The **record type**, defined RECFM and LRECL are identified in the first line of the column header.

#### Header Line 2: Field Name

Where present, the names of fields selected for display, are used as the column header names and are located in the second line of the column header.

This header line may not be switched off.

#### Header Line 3: Field Reference

The third column header line displays the field number (reference) within the record. By default, when table format is specified, the fields displayed will be the lowest level of a nested group of fields. Therefore, the names of fields that define a group of fields, will not be displayed and so their corresponding field numbers will also be missing from the sequence of displayed fields.

This header line may be switched on/off using the **SET REFERENCE** command.

#### Header Line 4: Field Type

The fourth column header line displays the field structure (type) as a 2-character data type code followed by the field's start position (byte number within the record) and the field length. For fields whose lengths are defined in number of bits, the field start position includes an offset into the byte following a "." (decimal point) and the unit for field length is bits. The start position and length displays are separated by a ":" (colon).

Recognised data type codes are as follow:

AN	CHARACTER, VARCHAR, XVARCHAR, STRUCTURE or UNION
BI	BINTEGER
BN	INTEGER
BT	BIT
FB	FIXEDHEX, FIXEDBIN
FP	FLOAT
PD	DECIMAL
X	HEXADECIMAL
ZD	ZONED

A definition of each of these data types may be found under **Data Type Clause** in the CREATE STRUCTURE command description.

This header line may be switched on and off using the **SET TYPE** command.

#### Header Line 5: Scale

The fifth column header line displays a scale for each field in the display. The width of the scale is equivalent to the width of the field display area which is wide enough to display all the field data or, for numeric fields, the largest possible numeric value.

Where the display field is wider than 2 bytes, a scale begins with "<" (less than) as position 1 and ends with ">" (greater than) as the last position of the field.

The scale characters displayed and the ability to switch this header line on and off is controlled by the **SET SCALE** command.

## Prefix Area

By default, the record prefix is set on and so the prefix area is displayed containing the record sequence numbers. The record prefix area may be tailored or set on/off using the **SET PREFIX** command.

## Data Records

By default, only the first record of the file and all records that share the same Record Type Object (**RTO**) as the first record, are selected for display. All other records associated with other RTOs are suppressed. Records that have been selected for display are identified as **data records**.

Using the **VIEW** command, records of different record types may be displayed concurrently within a multi record view.

Where multiple record types are visible, each record in the window display area that is of a different record type to the previous record, will be immediately preceded by its column headers. Therefore, when viewing a mixture of records of different record types, the display area may contain multiple groups of column header lines. It is also possible that only part of a group of column header lines is displayed at the bottom of the display area.

Scrolling left and right will scroll through fields within the record.  
Scrolling up and down will scroll through the records.

## Shadow Lines

A record or group of consecutive records that are not displayed because they are suppressed, specifically excluded by the user or have no associated RTO (not selected) are, by default, replaced by a **shadow line**.

A shadow line specifies the number of consecutive records excluded from the display, the reason for their exclusion ("suppressed", "excluded" or "not selected") and, if applicable, the associated **record type**. Shadow lines may be set on/off using the **SET SHADOW** command.

## Single Record View

An SDE MDI child window containing data from a single record displayed vertically so that each field of the record occupies a single line of the display.

A group of 3 field column-heading lines are displayed within the window display area immediately above the first field in view. Header line 1 contains the record sequence number; Header line 2 the **record type** (as illustrated in Header line 1 of **multi record view**) and Header line 3 contains the column headings **Field** and **Data**.

The **Field** column displays the field name (if defined), the level of nesting of the field within field groups and the field structure (type) as illustrated above in header line 4 of the **multi record view**.  
The **Data** column displays the data within the record for the specific field.

Each line of data identifies an individual field and its equivalent value, collectively referenced as a **field data line**.

Records that are suppressed, excluded or not selected cannot be viewed within a single record view.

Scrolling left and right will display the previous and next records respectively that have not been suppressed.  
Scrolling up and down will display the previous and next fields within the record respectively.

---

## SDE Data Formats

Data within an SDE window view is displayed in one of the following data formats.

- **Table Format**
- **Single Format**
- **Character Format**
- **Hexadecimal Format**

### Table Format

Forces a **multi record** window view where record data is expanded to fixed positions and split into its component fields as defined by the associated Record Type Object (**RTO**). The data is then processed so that all fields are displayed as printable character, numeric data fields having first been converted to decimal.



Record type: Instruction F(46)												
Language	RecTypeX	RecTypeN	ALevel	ADAF	Flag	FLevel	DataLen	INSTESD	INSTStm			
#3 #5	#6	#7 #8	#9	#10	#11	#12	#13					
BN 1:1 X 2:2	BN 2:2	BN 4:1 X 5:1	X 6:1	X 7:4	BN 11:2	BN 13:4	BN 17:4					
<...> <...>	<...>	<...>	<...>	<...>	<...>	<...>	<...>					
000029	HLASM 0036	54	3 00	01	00000000	32	1	10				
000043	HLASM 0036	54	3 00	01	00000000	32	1	20				
000045	HLASM 0036	54	3 00	01	00000000	32	1	21				
000047	HLASM 0036	54	3 00	01	00000000	30	1	22				
000101	HLASM 0036	54	3 00	01	00000000	30	1	75				
000103	HLASM 0036	54	3 00	01	00000000	32	1	76				
000105	HLASM 0036	54	3 00	01	00000000	32	1	77				
000109	HLASM 0036	54	3 00	01	00000000	32	1	80				

Figure 15. SDE Table Format.

## Single Format

Record data is processed as for a table view except that the display becomes a **single record** window view of the **default record**.

Record>			000001
Record type: AM F(5700)			
Field		Data	
1 AM	AN 1:5700		
2 AmDATE	AN 1:8	20040323	
2	AN 9:2		
2 AmK	AN 11:27		
3 AmKElCur	BN 11:1	1	
3 AmKElMax	BN 12:1	1	
3 AmKLineN	BN 13:1	1	
3 AmKLineL	BN 14:1	12	
3	AN 15:6		
3 AmKEY12	AN 21:12		
4 AmKEYC	AN 21:5	SUPP/	
4 AmKEY	AN 26:7	A1EQUIP	
3	AN 33:1		
3 AmCOUNTY	AN 34:4	SUPP	
2	AN 38:3		
2 AmLRECL	AN 41:4		

Figure 16. SDE Single Format.

## Character Format

Record data in the current multi or single record view is mapped as a single, variable length character field with field name "Record". No conversion of numeric or unprintable data is performed and the window view (either multi or single record) remains unchanged.

## Hexadecimal Format

Record data is processed as for character format with the addition that the data is also displayed in hexadecimal below the character data. Note that the Hex display occupies an additional 2 lines of the display per record.

## SDE Data Types

Structured records may be mapped by a record type object (RTO) whereby fields may be interpreted as being of any of the the following supported field types:

### Bit Flags

A Binary field of length specified in number of bits where each bit is interpreted as a flag switch which is either ON (1) or OFF (0). Fields of this data type have a data type record header of **"BT *ppp:nn*"** where *ppp* is the position of the field within the record (in bytes) and *nn* is the length of the field (in bits).

### Character Fixed

A Character field of fixed length specified in number of bytes whose contents are interpreted as printable ASCII or EBCDIC character data. Fields of this data type have a data type record header of **"AN *ppp:nn*"** where *ppp* is the position of the field within the record (in bytes) and *nn* is the length of the field (in bytes).

### Character Variable (Nominated Length Field)

A Character field of variable length specified in number of bytes by a nominated field within the record data. The nominated field may be of any numerical data type and the character field contents are interpreted as printable ASCII or EBCDIC character data. Fields of this data type have a data type record header of "**AN *ppp:nn***" where *ppp* is the position of the field within the record (in bytes) and *nn* is the length of the field (in bytes).

### Character Variable (Fixed Position Length Field)

A Character field of variable length specified in number of bytes by an **Integer Binary (Byte)** field located immediately before the character data. The Integer Binary field is of a predefined length which may be defined as being included with or excluded from the character field length. The character field contents are interpreted as printable ASCII or EBCDIC character data. Fields of this data type have a data type record header of "**AN *ppp:nn***" where *ppp* is the position of the field within the record (in bytes) and *nn* is the length of the field (in bytes).

### Fixed Point Binary

A Binary field defined as having a precision and scale specified in number of bytes, and whose value is displayed as a fixed point decimal number. The length of the binary field is implied by the precision which defines the total number of decimal digits. The scale defines the number of digits to the right of the decimal point. Fields of this data type have a data type record header of "**FB *ppp:nn***" where *ppp* is the position of the field within the record (in bytes) and *nn* is the length of the field (in bytes).

### Fixed Point Packed Decimal

A Packed Decimal field defined as having a precision and scale specified in number of bytes, and whose value is displayed as a fixed point decimal number. The length of the packed decimal field is implied by the precision which defines the total number of decimal digits. The scale defines the number of digits to the right of the decimal point. Fields of this data type have a data type record header of "**PD *ppp:nn***" where *ppp* is the position of the field within the record (in bytes) and *nn* is the length of the field (in bytes).

### Fixed Point Zoned Decimal

A Zoned Decimal Character field defined as having a precision and scale specified in number of bytes, and whose value is displayed as a fixed point decimal number. The length of the zoned decimal field is implied by the precision which defines the total number of decimal digits. The scale defines the number of digits to the right of the decimal point. Fields of this data type have a data type record header of "**ZD *ppp:nn***" where *ppp* is the position of the field within the record (in bytes) and *nn* is the length of the field (in bytes).

### Floating Point Hexadecimal

A Hexadecimal Floating Point field of length 4 bytes (short) or 8 bytes (long) whose value is displayed as a decimal number consisting of a normalised mantissa and exponent. Fields of this data type have a data type record header of "**FP *ppp:nn***" where *ppp* is the position of the field within the record (in bytes) and *nn* is the length of the field (in bytes).

### Floating Point Binary

A Binary Floating Point (IEEE 754) field of length 4 bytes (short) or 8 bytes (long) whose value is displayed as a decimal number consisting of a normalised mantissa and exponent. Fields of this data type have a data type record header of "**FP *ppp:nn***" where *ppp* is the position of the field within the record (in bytes) and *nn* is the length of the field (in bytes).

### Hexadecimal

A Hexadecimal field of a fixed length specified in number of bytes whose contents are displayed as a printable hexadecimal character string. Fields of this data type have a data type record header of "**X *ppp:nn***" where *ppp* is the position of the field within the record (in bytes) and *nn* is the length of the field (in bytes).

### Integer Binary (Bit)

A Binary field of length specified in number of bits, the contents of which are interpreted as being numeric and whose value is displayed as an integer decimal number. Fields of this data type have a data type record header of "**BI *ppp:nn***" where *ppp* is the position of the field within the record (in bytes) and *nn* is the length of the field (in bits). Compare with **Integer Binary (Byte)** data type which is a binary integer field with length expressed in number of bytes.

### Integer Binary (Byte)

A Binary field of length specified in number of bytes, the contents of which are interpreted as being numeric and whose value is displayed as an integer decimal number. Fields of this data type have a data type record header of "**BN *ppp:nn***" where *ppp* is the position of the field within the record (in bytes) and *nn* is the length of the field (in bytes). Compare with **Binary Integer (Bit)** data type which is a binary integer field with length expressed in number of bits.

### Structure

A Structure consisting of one or more fields each defined as being any of the supported data types. The structure field length is defined as being the sum of all field lengths within the structure. A field within a structure may itself be of the structure data type, so defining a structure nested within a structure.

### Union

A Union of one or more fields of any of the supported data types. Each field in the union occupies the same start position in the data record. The union field length is defined as being the length of the longest field in the union. Unions allow the same data to be interpreted as more than one data-type.

---

## SDE Current Window

The current SDE window is the SDE window view on which focus was last placed. This may be the focus window or, if the focus window is not an SDE window view, the last SDE window view visited.

The concept of a current SDE window view is important when executing SDE commands from a CBL<sub>e</sub> text edit view via the **SDATA** command which directs the SDE command to the current SDE window view.

Using **SDATA** is particularly useful for issuing stored SDE commands from within a **CMX** file.

---

## Default Record Type

SDE windows are capable of displaying records of more than one **record type** simultaneously within the same display area.

Because many SDE commands and functions operate on records of a single, specific record type, the concept of a default record type exists for use when a record type has not been specifically identified via a command parameter.

The following identifies the order in which the default record type is determined by CBL<sub>i</sub> SDE:

1. The record type of the **focus line** if the focus line is a visible record or an EXCLUDED shadow line.
2. The prevailing setting of DRECTYPE.

The value of DRECTYPE is assigned when the SDE window is opened and is established in the following order:

1. The record type that is the first, non-generic argument specified on the VIEW parameter of the **EDIT** or **BROWSE** command.
2. The record type of the first visible record of the edited or browsed data.
3. The record type of the first **RTO** in the **SDO**.

The DRECTYPE value may be updated by any of the following:

- **SET DRECTYPE** explicitly sets the DRECTYPE value.
- Execution of the VIEW command will update DRECTYPE to be the first, non-generic record type argument.
- Execution of any command that supports a *record\_type* argument will update DRECTYPE to be the record type selected for the command. This is true whether the record type has been specified explicitly in the command syntax or implied by the record type of focus line. (e.g. **WHERE**, **LOCATE**, etc.)

The current value DRECTYPE may be interrogated using the QUERY or **EXTRACT DRECTYPE** commands.

---

## REXX Macros

As in the CBL<sub>e</sub> text edit environment, user macros may be written to perform functions within SDE sessions using the REXX procedure language.

The name associated with the Structured Data Environment is CBLSDATA and should be specified on the REXX instruction ADDRESS if commands are to be directed to the SDE environment. However, if a macro is executed from within an SDE window, CBLSDATA is automatically set as the default environment.

The current environment may be identified using the REXX built-in function ADDRESS().

## Macro Path

An SDE REXX macro is executed by executing the **MACRO** command followed by the full fileid of the macro.

Alternatively, if the macro exists in a library within the macro path, simply specify the macro name. Libraries in the macro path are searched in order until a file name that matches the macro name is found.

The macro path used by SDE is the same as that defined for the CBL<sub>e</sub> text edit environment. i.e. The macro path is a list of directories assigned to the **Edit.MacroPath** variable set via the System or User **CBLINI** file, or via the CBL<sub>e</sub> **SET MACROPATH** command.

For MVS systems, the macro path usually consists of three PDS/PDSE libraries as follow:

1. A user specific macro library.

2. A site-wide macro library common to all users.
3. An installation macro library common to all users and containing useful macros written and owned by CBL and distributed to all CBLi (SELCOPY or CBLVCAT) customers.

Users should not update or modify macros in this library as they may be subject to change at the discretion of CBL. A detailed description on the use of each of an individual macro is documented within the macro executable file itself. (See **CBLi REXX Macros** for a list of CBL supplied macros.)

## SDE Profile (SDEPROF) Macro

When an SDE window view is opened, the macro path libraries are searched for the first occurrence of the SDE profile macro, **SDEPROF**.

The SDEPROF macro may be used to define the user's SDE environment. Any System or User CBLIINI file options that correspond to SDE SET command options, may be overridden for an SDE session by including the SET command in the SDEPROF macro.

A default SDEPROF macro is distributed as part of the CBLi product bundle and exists within the installation macro library.

# Command Line (Primary) Commands

SDE commands may be issued from:

1. The SDE command line.
2. A CBLc text edit window command line using the **SDATA** command.
3. A CBLc edited text file using the **SDATA** command and **CMDTEXT** facility.
4. A CBLc/SDE REXX edit macro.
5. A programmable **function key**.

Multiple SDE commands may be issued in a single invocation by separating each command with the line end character (set to ";" semi-colon by default.)

## Executing SDE Commands from a CBLc Text Edit Window

The CBLc CLI command **SData** may be used to prefix a command stream to be passed to the Structured Data Environment (SDE).

On using this method to execute an SDE command that updates the display of data within an SDE edit view, the **current SDE window** display gets updated and the focus remains on the current focus CBLc edit view.

**EDIT** and **BROWSE** commands will pass focus to the current SDE window.

```
<sdata create structure   CBL.CBLI.STRUCT(COMPSTR)   from cobol CBL.COPYBOOK.COBOL(COMPDEF)
<sd edit   CBL.SDE.EMP   using CBL.CBLI.STRUCT(COMPSTR)
<sd select  Key,InvNumb,DeliveryDate  from Orders    in CBL.CBLI.STRUCT(COMPSTR)
```

---

## Command Reference Syntax Conventions

### How to read the Syntax Diagrams

The following rules apply to the syntax diagrams used in this command reference.

1. The diagrams should be read from left to right, from top to bottom, following the path of the line.
  - ◆ The >>- symbol indicates the beginning of a statement.
  - ◆ The ->< symbol indicates the end of a statement.

2. Required items appear on the horizontal line (the main path).

```
>>--- required_item -----><
```

3. Optional items appear below the main path.

```
>>--- required_item ---+-----+-----><
                        |               |
                        +-- optional_item -----+
```

4. If an optional item appears above the main path, then that item has no effect on the execution of the statement and is used only for readability.

```
                        +-- optional_item -----+
                        |               |
>>--- required_item ---+-----+-----><
```

5. If you can choose from two or more items, they appear vertically, in a stack.

6. If you must choose one of the items, one item of the stack appears on the main path.

```
>>--- required_item ---+-- required_choice1 ---+-----><
                        |               |
                        +-- required_choice2 ---+
```

7. If choosing one of the items is optional, the entire stack appears below the main path.

```
>>--- required_item ---+-----+-----><
                        |         |
                        +-- optional_choice1 --+
                        |         |
                        +-- optional_choice2 --+
```

8. If one of the items is the default, it appears above the main path and the remaining choices are shown below.

```
>>--- required_item ---+-----+-----><
                        |         |
                        +-- default_choice ----+
                        |         |
                        +-- optional_choice1 --+
                        |         |
                        +-- optional_choice2 --+
```

9. An arrow returning to the left, above the main line, indicates an item that can be repeated.

```
>>--- required_item ---+-----+-----><
                        v         |
                        +-- repeatable_item ---+
```

10. If the repeat arrow contains a comma, you must separate repeated items with a comma.

```
>>--- required_item ---+-----+-----><
                        v ,         |
                        +-- repeatable_item ---+
```

11. A repeat arrow above a stack indicates that you can repeat the items in the stack.

```
>>--- required_item ---+-----+-----><
                        v         |
                        +-----+-----+
                        |         |
                        +-- optional_choice1 --+
                        |         |
                        +-- optional_choice2 --+
```

12. Uppercase characters in keywords indicate the minimum abbreviation allowed for that particular command or parameter and must be spelled exactly as shown.

13. Variables appear in all italic, lowercase letters and represent user-supplied names or values.

14. If punctuation marks, parentheses, arithmetic operators, or other such symbols are shown, you must enter them as part of the syntax.

15. Where a parameter, immediately following a command verb, begins with a non-alpha character, no separating blank is required between the command verb and the parameter. e.g. Add8, CHANGE/abc/xyz/

---

## BOTTOM

### Syntax:

```
>>-- B0tt0m -----><
```

Display the last page of data.  
Equivalent to the **DOWN MAX** command.

None.

DOWN  
TOP

[illegible]

Perform a read-only edit of a structured data set within **SDE**.

An SDE edit display window is opened and focus is passed to the new window. Depending on the value on the FORMAT parameter, the SDE display is either a **multi record** window view or a **single record** window view for the first record selected.

If the Structure Definition Object (SDO) specified by the USING STRUCTURE parameter is not already in storage, it is loaded from the appropriate Structure Definition File (SDF) Record Type Objects (RTO) within the specified SDO are then used to map the records in the data set.

For every record in the file, each RTO is checked until one is found whose criteria is matched by the record characteristics. This RTO is then used to map the record. By default, all fields within records that have an associated RTO are displayed as printable character, numeric data fields having first been converted to decimal.

If no SDO is specified or a record does not possess the characteristics required to satisfy any of the available RTO criteria, then FORMAT CHARACTER is applied to the record. i.e. the data is displayed as a single, variable (RECFM=V) or fixed (RECFM=F) length character field with field name "Record". No data conversion is performed.

Unlike the **EDIT** command, **BROWSE** displays the logical record length of each record to the left of the first column of record data in the display area. Although the display of record lengths has a column heading "LRecL", it is not part of the record data and so is excluded from commands that affect the data display. e.g. scrolling operations.

If the VIEW parameter is specified with a non-generic argument, the DRECTYPE setting is initialised as the first *record\_type* parameter specified on the VIEW parameter. Otherwise, the DRECTYPE setting is initialised as the record type of the first visible record in the file.

Data that does not meet the specifications of the field definition is considered to be invalid and is represented by asterisks.

## Parameters:

*fileid*

The fileid of the data set containing records to be displayed.  
Fileid may be the DSN of a sequential or VSAM data set or the DSN of a PDS/PDSE member.

USING (STRUCTURE) *struct\_name*

Identifies the SDO to be applied to the data records. *struct\_name* is the SDF fileid assigned to the SDO in a **CREATE STRUCTURE** command.

FORMAT

FMT

Specifies the format in which record data will be displayed in the initial SDE view.

SINGLE  
SNGL Single record format.

TABLE Table format. (Default)

CHARACTER Multi record view with all records mapped as a single, variable length character field with field name "Record". No data conversion is performed.

HEX Same as CHARACTER with the addition that the data is also displayed in Hex below the character display. Note that the Hex display occupies an additional 2 lines of data.

The data display format may be later altered using the **FORMAT** and/or **ZOOM** commands.

VIEW *record\_type*

VIEW \*

Identifies one or more **record types** for which records will be displayed in the initial SDE view. Records that are associated with different record types are not selected for display and are displayed as shadow lines instead.  
If \* (asterisk) is specified, then records of all record types are displayed.

Records within the display may be later selected or deselected using the **VIEW** command.

Default is VIEW \*.

PROFILE *macro\_name*

Specifies the REXX SDE edit macro to be executed as the profile when the data set is displayed.

*macro\_name* must exist in a library within the SDE macro path.

The PROFILE option only alters the profile used for the file currently being displayed. It does not define the profile macro to be used for subsequent SDE edit/browse.

The default is to execute a macro with member name SDEPROF if it exists.

NOPROFILE

Suppresses use of a profile macro when displaying the file.

The NOPROFILE option only suppresses use of a profile for the file currently being displayed. It does not suppress use of a profile macro for subsequent SDE edit/browse.

## Examples:

```
<sd browse CBL.SDE.MOD.ZJ2202 using CBL.CBLI.SDO(PRODL) view ARCX1,PRODMAST,PRODX
```

Browse records from data set "CBL.SDE.MOD.ZJ2202" within the SDE.

Initially display all records of type ARCX1, PRODMAST and PRODX where the record type objects (RTO) are found in the structure definition object (SDO) referenced by CBL.CBLI.SDO(PRODL).

## See Also:

**EDIT**  
**CREATE STRUCTURE**



## CHANGE

### Syntax:

```

>>--- Change ----- string1 -- string2 ---+-----+-----+-----+-----+----->
                                           +- NEXT ---+ +- CHARs ---+
                                           |           | |           |
                                           +- ALL  ---+ +- PREFIX ---+ +- EX  ---+
                                           |           | |           |
                                           +- FIRST ---+ +- SUFFIX ---+ +- NX  ---+
                                           |           | |           |
                                           +- LAST  ---+ +- WORD  ---+ +- X   ---+
                                           |           | |           |
                                           +- PREV ---+

+--- #ALL  -----+--- .ZFIRST ----- .ZLAST ---+
>-----+-----+-----+-----+-----+-----><
+--- pos1 -----+--- .name1 -----+
|               | |               |
|               +- pos2 ---+         +- .name2 ---+
|               | |               |
|               +-----+-----+
|               |               |
|               +- , -----+
|               |               |
+--- ( -+--- field_col -----+ ) ---+
|               |               |
+--- field_col1:field_col2 -----+

```

### Description:

Search data records in the current edit or browse view for the specified character string or numeric value (*string1*) and replace it with *string2*. Only data records that are of the **default record type** (visible and EXCLUDED records) are included in the search. Records groups that are of a different record type or are NOTSELECTED or SUPPRESSED, are excluded from the search.

If the specified occurrence (all, first, last, next or previous) of the search string or numeric value (*string1*) is found within a record, then:

1. If the record is EXCLUDED, it is made visible.
2. The cursor is positioned at the beginning of the string or numeric field.
3. If necessary, scrolling occurs to display the found data.
4. If the replace string (*string2*) satisfies the data type, precision and scale requirements for the field, then *string2* replaces *string1* in the field.

All occurrences of *string1* are highlighted in the text. (Enter the **RESET FIND** command to turn off the highlighting.)

Use of CHANGE with no parameters is equivalent to **RCHANGE** (assigned to function key **PF6** by default) and repeats the last change command executed, including all its specified parameters.

To find and optionally change the next occurrence of *string1*, execute a combination of **RFIND** (assigned to PF5 by default) followed by RCHANGE.

If *string1* and *string2* are character strings of unequal length, then the following will occur:

- If the length of *string1* is greater than the length of *string2*, then text to the right of the replaced string will be shifted left.
- If the length of *string1* is less than the length of *string2*, then text to the right of the replaced string will be shifted right absorbing multiple, consecutive blanks but leaving at least one blank between each word.

The **FORMAT** of the SDE display affects the execution of CHANGE.

### FORMAT CHAR and HEX

A character compare for the supplied search string is performed against entire, unformatted data records.

*field\_col* and #ALL parameters are not applicable to these display formats and are ignored.

### FORMAT SINGLE and TABLE

The search string is compared against **individual fields** that have been selected for display in the expanded data record. (See **SELECT**)

Fields are searched from left to right in the order that they appear in the display. This is true regardless of the order in which field columns are specified on the CHANGE command, or the order in which fields are encountered within the expanded record.

If a field column is specified on the CHANGE command that is not part of the display (e.g. an expanded group field or a field that has been removed from display by a SELECT command), then an error message is returned.

For character data type fields, a string compare is performed. For numeric data type fields (binary, packed decimal, floating point, zoned, etc.), then the following will occur:

1. If *pos1*, *pos2* positional parameters are **not** specified, the search string is interpreted as a signed numeric value and an arithmetic compare for equality is performed against the field's formatted numeric value.

The length and data type of the numeric field, and the number of digits in the search value are not significant. e.g.

```
CHANGE 67 23
```

Finds numeric fields with value "67" (e.g. "0067", "67.00", "0.0670E+03") and character fields containing the string "67" (e.g. "167 Baker Street") and replaces with the value "23" using the appropriate data type

If the search string is non-numeric, then numeric data type fields are not searched. Therefore, to bypass searching numeric data type fields, explicitly set the search string to be character or hex using 'string', C'string' or X'string' formats respectively. e.g.

```
CHANGE '67' '23'
CHANGE C'67' C'23'
CHANGE X'F6F7' X'F2F3'
```

Finds only character fields containing the string "67" and replaces with character string "23".

2. If *pos1*, *pos2* positional parameters are specified, the search string is interpreted as being a character string and a string compare is performed on the character representation of numeric data.

Although the range of positions may span a number of fields, the search is still performed against individual fields within the range. i.e. a match for the search string will not be found if the data spans a field boundary.

A match for the search string on just part of the character representation of a numeric field is considered to be a successful hit and the whole numeric field is highlighted. e.g.

```
CHANGE 476 850 21 100
```

Finds character and numeric fields where character representation of the data contains the string "476" and replaces it with "850". (e.g. A zoned decimal field with value "14760" would become "85060")

Where a character string or numeric value is found in a numeric field, the entire field is highlighted and the cursor positioned at the start of the field.

If replacing a string in a character field would exceed the defined maximum length of the field then no update will take place.

## Parameters:

*string1*

The CHANGE search string. The search string may be any of the following:

- ◊ An unquoted numeric value. The search string is treated as a numeric value when a numeric field is searched in a SINGLE or TABLE format view. In all other cases a numeric search string is treated as a character string.
- ◊ An unquoted character string containing no commas or blanks. The search for the character string will be case-insensitive so that uppercase and lowercase characters are treated as being the same.
- ◊ A character string enclosed in single (') or double (") quotation marks. The search string may contain embedded commas and blanks and the character string will be case-insensitive. A string enclosed in quotes may still be interpreted as a numeric value.

Two adjacent quotation mark characters that are embedded in a search string which is enclosed by the same quotation mark characters, will be treated as a single occurrence of the character. e.g.

```
CHANGE 'Jim O''Brien' 'James O''Brien'
```

Find the character string "Jim O'Brien" and replaces it with "James O'Brien".

- ◊ A character string enclosed in single (') or double (") quotation marks with the prefix (or suffix) C. This is equivalent to specifying a quoted search string but that the string search will be case-sensitive. (e.g. C'Book')
- ◊ A hexadecimal string enclosed in single (') or double (") quotation marks with the prefix (or suffix) X.

*string2*

The CHANGE replace string used to replace *string1*. The replace string may be any of the following:

◇ An unquoted numeric value. The replace string is treated as a numeric value when it replaces a value in a numeric field in a SINGLE or TABLE format view. In all other cases a numeric replace string is treated as a character string.

3. An unquoted character string containing no commas or blanks. String2 may be null (").
4. A character string enclosed in single (') or double (") quotation marks that may contain embedded commas and blanks. String2 may be null (C").

Two adjacent quotation mark characters that are embedded in a replace string which is enclosed by the same quotation mark characters, will be treated as a single occurrence of the character. (See example under *string1*.)

5. A character string enclosed in single (') or double (") quotation marks with the prefix (or suffix) C. This is equivalent to specifying a quoted search string but that the string search will be case-sensitive. (e.g. C'Book')
6. A hexadecimal string enclosed in single (') or double (") quotation marks with the prefix (or suffix) X.

## ALL

Where field conditions are satisfied, change all occurrences of *string1* to *string2*. A message is displayed providing the number of occurrences of *string1* that have been changed to *string2*. If NX is not specified, all excluded records that contain an occurrence of the *string1* are made visible whether or not *string1* is replaced.

## FIRST

Search forwards from the top of the file data (i.e. the first position of the first data record) to find the first occurrence of *string1* and attempt to replace it with *string2*.

## LAST

Search backwards from the bottom of the file data (i.e. the last position of the last data record) to find the last occurrence of *string1* and attempt to replace it with *string2*.

## NEXT

Search forwards from the current cursor location to find the next occurrence of *string1* and attempt to replace it with *string2*. If the cursor is not within the window's data display area, the search begins at the first position of the first visible or excluded record within the display area that is of the default record type.

## PREV

Search backwards from the current cursor location to find the previous occurrence of *string1* and attempt to replace it with *string2*. If the cursor is not within the window's data display area, the backwards search begins at the first position of the first visible or excluded record within the display area that is of the default record type.

## CHARS

For non-numeric search strings only, CHARS indicates that a successful match occurs if *string1* is found anywhere within the data being searched.

## PREFIX

For non-numeric search strings only, PREFIX indicates that a successful match only occurs if *string1* is found at the start of a word within the data being searched. i.e. the matched text must precede an alphanumeric character and either be preceded by a non-alphanumeric character or be at the start of a line or field.

## SUFFIX

For non-numeric search strings only, SUFFIX indicates that a successful match only occurs if *string1* is found at the end of a word within the data being searched. i.e. the matched text must be preceded by an alphanumeric character and either precede a non-alphanumeric character or be at the end of a line or field.

## WORD

For non-numeric search strings only, WORD indicates that a successful match only occurs if *string1* is found to be a complete word within the data being searched. i.e. the matched text must either be preceded by a non-alphanumeric character or be at the start of a line or field, and either precede a non-alphanumeric character or be at the end of a line or field.

EX  
X

Search EXCLUDED data records only. By default, both visible and EXCLUDED records are searched. CHANGE does not search records that are NOTSELECTED or SUPPRESSED.

## NX

Search only visible data records (i.e. not EXCLUDED). By default, both visible and EXCLUDED records are searched. CHANGE does not search records that are NOTSELECTED or SUPPRESSED.

*pos1*

The first position of a range of positions within the data record to be searched.

For SINGLE and TABLE formats, this is a position in the expanded record. Only those fields, or parts of fields, that fall within the position range will be searched. Fields will be searched in the order that they occur within the display area.

*pos1* must be a number greater than or equal to 1 and less than or equal to the maximum length of the data records (or length of the expanded records in the case of SINGLE or TABLE format views.)

For all display formats, *string1* is treated as being non-numeric and the search is actioned on the character representation of the record data.

*pos2*

The last position of a range of positions within the data record to be searched.

if *pos1* is greater than *pos2* in the data record, then the values are swapped.

If *pos2* is greater than the maximum length of the data records (or length of the expanded records in the case of SINGLE or TABLE format views) then *pos2* is set equal to the maximum (or expanded) record length.

Default is *pos1* so defining a search range of width 1.

#ALL

Search all field columns in the current TABLE or SINGLE format display.

*field\_col*

An individual field column to be searched within a TABLE or SINGLE format display.

The field column may be identified by its reference number (e.g. #6) or its name (e.g. JobID). If a field name is used, enclosing parentheses are **mandatory**. Field names are automatically converted to a field reference and Referencing the same field column more than once will not cause an error.

If the field is an array, then all elements of the array are searched. To search an individual element of an array, the element occurrence should be specified in parentheses as a subscript to the field reference/name. e.g. #13(3) for the 3rd element of a single dimension array. An entry must exist for each dimension of the array. e.g. RoomSize(6,2,4) - a three dimensional array.

Specification of multiple field columns must either be enclosed in parentheses and/or separated by commas. The field columns may be specified in any order, however, the data is always searched from the first column in the display to the last.

Field column search specifications may be a combination of individual field columns and columns ranges. e.g. (JobID #6 Tax\_Reference #12 #15:#20).

A search is only performed on field columns that are selected for display. Therefore any field column specified on the CHANGE command that has not been selected for display, will be ignored.

*field\_col1:field\_col2*

The first and last fields of a range of field columns to be searched within a TABLE or SINGLE format display.

*field\_col1* and *field\_col2* must be separated by ":" (colon) and if *field\_col1* occurs after *field\_col2* in the data record, then the values are swapped. *field\_col1* and *field\_col2* have the same specifications as *field\_col*.

Specification of multiple field column ranges must either be enclosed in parentheses and/or separated by commas. Field column search specifications may be a combination of individual field columns and field columns ranges. e.g. (FirstName:LastName, #2, Salary:Bonus, EmpNo, #10:#15). If field names are used, enclosing parentheses are **mandatory**.

*.name1*

A label name identifying the first record of a range of data records to be searched. The preceding "." (dot) is mandatory. Default is .ZFIRST.

*.name2*

A label name identifying the last record of a range of data records to be searched. The preceding "." (dot) is mandatory. If *.name2* occurs before *.name1* in the display, then the order is reversed. If CHANGE PREV is executed and *.name1* is specified, the default is .ZFIRST. Otherwise the default is .ZLAST.

## Examples:

change 22.3 303.25

Search all fields in the display belonging to the default record type for the next occurrence of the search string/value 22.3. Numeric fields are tested for numeric value 22.3, character fields are tested for character string "22.3" anywhere within the field. Where the field width and, for numeric fields, the precision and scale definitions are satisfied by the replacement string/value 303.25, text update will takes place.

change all 'ing' 's' suffix ex (DsName:ProcLib, JobStep)

Search selected character fields in the display that belong to excluded records of the default record type, for all occurrences of the word suffix string "ing" and replace the text with "s".

## See Also:

EXCLUDE  
FIND  
RCHANGE  
RFIND  
SELECT  
Find and Replace Data

## CREATE LIST

### Syntax:

```
>>-- CREate LIst -- list_name ----- FROM fileid ----->

>-- USING ---+-----+--- struct_name ---+-----+-----><
           |           |           |           |
           +- STRUCTure +-         +-- TITLE title_text --+
```

### Description:

Create an in storage list from records within a **structured data set**, that may then be displayed in a CBLi **List window** via the **DISPLAY LIST** command.

Only records of one record type may be accurately displayed within a list window. By default, the first record type object (**RTO**) within the **SDO** is used to map **all** the records within the file when generating the list.

A list column entry is generated for every field name defined by the RTO. Therefore, data within named fields that are part of a named group of fields will be repeated in different columns.

The List window displaying the list data will support most of the standard list window features as follow:

- **Field Descriptor Block (FDB)**
- **Edit View**
- **Selecting, Sorting and Filtering**
- **Sorting with the Cursor**

On exiting the List window display, the in-storage list is destroyed and storage is freed.

### Parameters:

*list\_name*  
The name to be assigned to the in-storage list data for reference in a DISPLAY LIST command.

FROM *fileid*  
Identifies the fileid of the data set containing the structured records.  
*fileid* may be the DSN of a sequential or VSAM data set or the DSN of a PDS/PDSE member.

USING (STRUCTURE) *struct\_name*  
Identifies the SDO to be applied to the data records. *struct\_name* is the SDF fileid assigned to the SDO in a **CREATE STRUCTURE** command.

TITLE *title\_text*  
Specifies the text to be displayed in the List window's title bar.

### Examples:

```
<sd create list AMLIST from CBL.AMSUPP.DA using CBL.CBLI.SDO(AMMAP) title "Contact Address Details"
Generate in-storage list AMLIST.
```

```
<sd display list AMLIST
Open a list window for the in-storage list AMLIST.
```

### See Also:

**DISPLAY LIST**  
**CREATE STRUCTURE**

# CREATE STRUCTURE

## Syntax:

```
>>- CREATE STRUCTURE --- struct_name ----->

+ COBOL+ +- , -----+
|      | v          |
>+- FROM -+-+-----+ copy_book -+-+-----+
|      | +- PL1 +    |
|      | +- , -----+
|      | v          |
+- ( -- record_definition +- ) -----+
|                                     +- ASM -----+
|                                     | COBOL -----+
+- NAMES ( --+- COBOL -----+ ) +-
|                                     +- DB2 -----+
|                                     +- PL1 -----+
|                                     | xlc +-
|
+- NOREplace --+
>+-+-----+-----+-----+-----+-----+<
|      |      |      |      |      |
+-- REPLACE --+      +- IGNORE --+
|      |      |      |      |
+-- CASE ( --+ RESPECT --+ ) +-
```

## Description:

Create a Structure Definition Object (SDO) to be used within SDE, and save it to a Structure Definition File (SDF). An SDO provides the information required to interpret individual logical fields within a file's data records.

Usually a CREATE STRUCTURE command will refer to a file-structure already defined within an existing COBOL or PL/1 copybook, but direct definition syntax is also supported (see *record\_definition* below).

## Parameters:

*struct\_name*

The name you are giving to the structure being created. *struct\_name* is the SDF fileid assigned to the SDO. This is the name that will subsequently be referenced on the *USING* parameter of *EDIT/BROWSE*. It must be a valid file-id in which to store the file-structure definition.

FROM COBOL|PL1 *copy\_book*

Identifies an existing COBOL or PL/1 copybook that defines the file-structure.

REPLACE

NOREPLACE

Either overwrite or give an error if *struct\_name* already exists.  
Default is NOREPLACE.

CASE IGNORE|RESPECT

Respect or ignore the case setting of any variable names used in the structure definition. Default is **IGNORE**.

*record\_definition*

Described fully below, this syntax provides the method of directly defining file record structures, each comprising one or more fields.  
e.g. From your HOME command-centre file ...

```

<sd create structure    CBL.CBLI.SDO(FILSTRUC)      \
(                                                              \
  REC-TYPE01 struct                                          \
  (  REC-TYPE          char(2),                             \
    NAME               char(20),                           \
    EMPLOYEE-NO        int(2) unsigned,                    \
    AGE                int(2) unsigned,                    \
    SALARY              dec(7),                             \
    MONTH              int(4) dim(12) unsigned,            \
    ) use when REC-TYPE='01'                                \
  ,                                                            \
  REC-TYPE02 struct                                          \
  (  REC-TYPE          char(2),                             \
    NAME               char(20),                           \
    JOB-TITLE          char(14),                           \
    ADDR1              char(20),                           \
    ADDR2              char(20),                           \
    POSTCODE           char(4)                             \
    ) use when REC-TYPE='02'                                \
  )                                                            \
names(COBOL)          replace      case respect

```

## Examples:

```

<sd create structure      CBL.CBLI.STRUCT(EMP) from cobol CBL.COPYBOOK.COBOL(EMP)
<sd edit  CBL.SDE.EMP     using CBL.CBLI.STRUCT(EMP)

```

## record\_definition Clause:

```

      +- , -----+
      v           |
>-- record_type -- STRUCTure -- ( --- data_definition -+-- ) -----><

```

## record\_definition Parameters:

*record\_type*

The record type used to identify the new record type object (**RTO**).

*data\_definition*

Described fully below, this syntax provides the method of directly defining the fields within a record structure.





SIGNED  
UNSIGNED

This numeric field is to be treated as signed or unsigned.  
Default is SIGNED.

ALIGNED  
UNALIGNED

Determines whether the field will be boundary aligned for its data type.  
Default is UNALIGNED.

REMark *remark*

A comment string placed in either single or double quotes.

DEFault *default\_value*

Specifies the default value for this field should a new record be added. It can be defined either as a numeric or quoted string constant, or as an expression referring to field names within the current record.

ENUMeration

Create an enumeration definition or specify the name of an existing enumeration for use with the current field.

*enam* may be specified to allocate a name to the enumeration definition or as a reference to an enumeration already defined within the current field definition clause.

Each *enumerator* should be specified in the form *display=value*, where *display* will be displayed when this field contains *value*. e.g.

```
enum (HLA=16, Cobol=17, P11=40)
```

*display* should be enclosed in quotes if it is to contain blanks. *display* may be specified without a corresponding *value*, in which case the previous value plus 1 will be implied.

USE WHEN *filter\_expression*

Often an individual record can be associated with an RTO by matching on record length. Where several RTOs exist for the same length record then a **USE WHEN** filter is required to match a record to its structure.

For a more detailed description of *filter\_expression* see the **USE** command.  
e.g. From your HOME command-centre file ...

```
<||sd create structure CBL.CBLI.SDO(FILSTRUC) \
(
  REC-TYPE01 struct
  (
    REC-TYPE      char(2),
    NAME          char(20),
    EMPLOYEE-NO   int(2) unsigned,
    AGE           int(2) unsigned,
    SALARY        dec(7),
    MONTH         int(4) dim(12) unsigned,
    char(2)
  ) use when REC-TYPE='01'
,
  REC-TYPE02 struct
  (
    REC-TYPE      char(2),
    NAME          char(20),
    JOB-TITLE     char(14),
    ADDR1         char(20),
    ADDR2         char(20),
    POSTCODE      char(4)
  ) use when (
    (
      REC-TYPE < '02'
      & REC-TYPE > '05'
    )
    |
    (
      REC-TYPE < '22'
      & REC-TYPE > '25'
    )
  )
)
names(COBOL) replace case respect
```

```

>> +- BINTeger -----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     |
|                                     +- ( n_bits ) -----+
|
+- BIT -----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     |
|                                     +- ( n_bits ) -----+
|
+- CHARACTER -----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     |
|                                     +- ( n_bytes ) -----+
|
+- DECimal -----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     |
|                                     +- (precision+-----+ ) -----+
|                                     |
|                                     +- , scale ---+
|
+- FIXed -----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     |
|                                     +- (precision+-----+ ) -----+
|                                     |
|                                     +- , scale ---+
|
+- FLOATBin -----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     |
|                                     +- ( n_bytes ) -----+
|
+- FLOAThex -----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     |
|                                     +- ( n_bytes ) -----+
|
+- HEXadecimal+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     |
|                                     +- ( n_bytes ) -----+
|
+- INTeger -----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     |
|                                     +- ( n_bytes ) -----+
|
+- STRUCTure -- ( --- data_definition --- ) -----+
+- UNION ----- ( --- data_definition --- ) -----+
+- VARCHAR -----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     |
|                                     +- ( max_bytes+-----+ ) -----+
|                                     |
|                                     +- , len_bytes+ , exclusive+
|
+- XVARCHAR -----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     |
|                                     +- ( max_bytes, len_field) -----+
|
+- ZONEd -----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     |
|                                     +- (precision+-----+ ) -----+
|                                     |
|                                     +- , scale ---+

```

BINteger	Defines a field to be interpreted as binary whole-number occupying <i>n_bits</i> bits, where <i>n_bits</i> defaults to <b>1</b> .
BIT	Defines a field to be interpreted as a series of ON OFF (1 0) switches occupying <i>n_bits</i> bits, where <i>n_bits</i> defaults to <b>1</b> .
CHARacter	Defines a field to be interpreted as a fixed-length character string occupying <i>n_bytes</i> bytes, where <i>n_bytes</i> defaults to <b>1</b> .
DECimal	Defines a field to be interpreted as a packed-decimal number allowing <i>precision</i> number of digits in total, of which <i>scale</i> number of digits will be displayed following the decimal-point. Default for <i>precision</i> is <b>7</b> . Default for <i>scale</i> is <b>0</b> .
FIXed	Defines a field to be interpreted as a binary number allowing <i>precision</i> number of digits in total, of which <i>scale</i> number of digits will be displayed following the decimal-point. Default for <i>precision</i> is <b>7</b> . Default for <i>scale</i> is <b>0</b> .

**FLOATBIN**

Defines a field to be interpreted as a Binary (IEEE 754) Floating-point number occupying *n\_bytes* bytes, where *n\_bytes* may be 4 or 8 (defaults to **4**).

**FLOATHEX**

Defines a field to be interpreted as a Hexadecimal Floating-point number occupying *n\_bytes* bytes, where *n\_bytes* may be 4 or 8 (defaults to **4**).

**HEXadecimal**

Defines a field to be interpreted as printable hexadecimal character string occupying *n\_bytes* bytes, where *n\_bytes* defaults to **1**.

**INTEger**

Defines a field to be interpreted as a binary whole-number occupying *n\_bytes* bytes, where *n\_bytes* defaults to **4**.

**STRUCTure**

Defines a field to be interpreted as a structure containing one or more elements of any of the supported data types, including other structures. Definition of each element should be separated by a comma.

**UNION**

Defines a field to be interpreted as a union of one or more elements of any of the supported data types. Definition of each element should be separated by a comma. Unions allow the same data to be interpreted as more than one data-type.

**VARCHAR**

Defines a field to be interpreted as a variable-length character string. The maximum possible length of the string is defined as **max\_bytes** bytes, defaulting to **0**. The actual length of the string is defined by a **len\_bytes** bytes binary integer at the start of the field. Default for *len\_bytes* is **2**. The value of the *exclusive* parameter indicates whether this length integer includes or excludes the length of itself. Permissible values are EXCLUSIVE (the default) and INCLUSIVE.

**XVARCHAR**

Defines a field to be interpreted as a variable-length character string. The maximum possible length of the string is defined as *max\_bytes* bytes, defaulting to **0**. The actual length of the string is defined by field name *len\_field*, which must be numeric.

**ZONEd**

Defines a field to be interpreted as a zoned-decimal character number allowing *precision* number of digits in total, of which *scale* number of digits will be displayed following the decimal-point. Default for *precision* is **1**. Default for *scale* is **0**.

**See Also:**

[EDIT](#)  
[BROWSE](#)

## DISPLAY LIST

**Syntax:**

```
>>- DISplay LIst -- list-name --+-----+----->
                                |         |
                                +--- select_clause ---+

>-----+-----+-----+-----<
        |         |         |         |
        +--- where_clause ---+ +--- sort_clause ---+
```

**Description:**

Open a CBLi [List window](#) to display an in-storage list generated via the [CREATE LIST](#) command.

The List window displaying the list data will support most of the standard list window features as follow:

- [Field Descriptor Block \(FDB\)](#)

- [Edit View](#)
- [Selecting, Sorting and Filtering](#)
- [Sorting with the Cursor](#)

On exiting the List window display, the in-storage list is destroyed and storage is freed.

## Parameters:

*list\_name*

The name of the in-storage list data to be displayed.

*select\_clause*

Identifies a valid **SELECT Clause** to display only those selected columns.

*where\_clause*

Identifies a valid **WHERE Clause** to filter rows of data.

*sort\_clause*

Identifies a valid **SORT Clause** to sort the specified columns of data.

## Examples:

```
>sd display list AMLIST select KEY,COMPNAME,FIRSTNAME,LASTNAME,DEPT where lastname=JONES sort DEPT a
```

Open a list window for the in-storage list AMLIST with select, sort and filter clauses applied.

## See Also:

[CREATE LIST](#)

---

# DISPLAY STRUCTURE

## Syntax:

```
>>-+- DISplay STRUCture -+--+-----+-----><
  |                         | |      |
  +- LS -----+ +- struct_name -+
```

## Description:

Open a CBLi **List window** to display the field definitions for each **record\_type** in the specified structure definition object (**SDO**).

The List window displaying supports the standard list window features as follow:

- [Field Descriptor Block \(FDB\)](#)
- [Edit View](#)
- [Selecting, Sorting and Filtering](#)
- [Sorting with the Cursor](#)

## Parameters:

*struct\_name*

The name of the SDO structure.

Default is the name of the SDO structure used in the **current SDE window**.

## Examples:

```
display structure CBL.CBLI.SDO(CBLDIST)
```

Open a list window for the structure definition PDSE member CBL.CBLI.SDO(DIRTYPES).

-CBLi for TSO 1.5B - Build=200805011107 OpSys=z/OS 1.6.0 User=NBJ2  
 File List Utilities System Window SwapList Help  
 -CBLi - Structure NBJ2.CBLI.SDD(COBSALES) data elements  
 View Back Forward FDB Edit Refresh Hel ? Sv ToF BoF wS wR Pfx < >  
 Command>

--RecType--	RefNo-	Level-	---Name----	Data Type-	--MaxLen--	--MinLen--	--SOf
REC-CARD	1	1	REC-CARD	STRUCTURE	56	56	
REC-CARD	2	2	CUST-ID	FIXED	2	2	
REC-CARD	3	2	SEQ	FIXED	2	2	
REC-CARD	4	2	CARD-TYPE	STRUCTURE	8	8	
REC-CARD	5	3	CR-OR-DR	CHARACTER	1	1	
REC-CARD	6	3	COMPANY	CHARACTER	7	7	
REC-CARD	7	2	CARD-NUMBER	DECIMAL	9	9	
REC-CARD	8	2	NAME	CHARACTER	25	25	
REC-CARD	9	2	VALID-FROM	DECIMAL	4	4	
REC-CARD	10	2	EXPIRES	DECIMAL	4	4	
REC-CARD	11	2	SEC	DECIMAL	2	2	
REC-CUST	1	1	REC-CUST	STRUCTURE	192	192	
REC-CUST	2	2	CUST-ID	FIXED	2	2	
REC-CUST	3	2	PASS	CHARACTER	15	15	
REC-CUST	4	2	LASTNAME	CHARACTER	15	15	
REC-CUST	5	2	FIRSTNAME	CHARACTER	15	15	
REC-CUST	6	2	COUNTRY	CHARACTER	2	2	
REC-CUST	7	2	POSTCODE	CHARACTER	12	12	
REC-CUST	8	2	CITY	CHARACTER	15	15	
REC-CUST	9	2	HOUSE	FIXED	2	2	
REC-CUST	10	2	STREET	CHARACTER	25	25	
REC-CUST	11	2	EMAIL	CHARACTER	35	35	
REC-CUST	12	2	PHONE	CHARACTER	25	25	
REC-CUST	13	2	MOBILE	CHARACTER	25	25	
REC-CUST	14	2	BALANCE	DECIMAL	4	4	
REC-NOTE	1	1	REC-NOTE	STRUCTURE	56	56	
REC-NOTE	2	2	CUST-ID	FIXED	2	2	
REC-NOTE	3	2	ORDER-REF	FIXED	4	4	
REC-NOTE	4	2	NOTE	CHARACTER	50	50	
REC-ORDER	1	1	REC-ORDER	STRUCTURE	106	86	
REC-ORDER	2	2	CUST-ID	FIXED	2	2	
REC-ORDER	3	2	ORDER-REF	FIXED	4	4	
REC-ORDER	4	2	QTY	FIXED	2	2	
REC-ORDER	5	2	ITEM-CODE	FIXED	2	2	
REC-ORDER	6	2	UNIT-PRICE	DECIMAL	4	4	

Line 1 of 55 | Col 1 of 600 | Views 1 | select \* sort RecType,RefNo,Level,Name  
 MA b 05/011

Figure 17. Display Structure List View.

Field Descriptor Block for columns displayed with the DISPLAY STRUCTURE command.

```

-CBLi for TSO 1.5B - Build=200805011107 OpSys=z/OS 1.6.0 User=NBj2
File List Utilities System Window SwapList Help
-CBLi - Structure data element list
View Back Forward FDB Edit Refresh Hel ? Sv ToF BoF wS wR Pfx < >
Command>

--Name-- --Type-- Key Offset Length -----
Aligned BitFlag No 74 1 This data element is aligned on its natura
ASCII BitFlag No 75 1 This field must be displayed with ASCII to
BitOff Int No 44 1 Bit offset
CPAct Int No 78 1 Actual number of create parameters.
CPChar01 ALPair No 84 8 Create parameter character value 01
CPChar02 ALPair No 97 8 Create parameter character value 02
CPChar03 ALPair No 110 8 Create parameter character value 03
CPChar04 ALPair No 123 8 Create parameter character value 04
CPIntV01 Int No 80 4 Create parameter integer value 01
CPIntV02 Int No 93 4 Create parameter integer value 02
CPIntV03 Int No 106 4 Create parameter integer value 03
CPIntV04 Int No 119 4 Create parameter integer value 04
CPMax Int No 77 1 Maximum number of create parameters.
CPTType01 Enum No 79 1 Create parameter type 01
CPTType02 Enum No 92 1 Create parameter type 02
CPTType03 Enum No 105 1 Create parameter type 03
CPTType04 Enum No 118 1 Create parameter type 04
DataType ALPair No 20 8 Data type
Dim Int No 59 2 Dimension at current level
Dims Int No 57 2 Total dimensions
DimExp01 ALPair No 139 8 Dimension expression 01
DimExp02 ALPair No 155 8 Dimension expression 02
DimExp03 ALPair No 171 8 Dimension expression 03
DimExp04 ALPair No 187 8 Dimension expression 04
DimExp05 ALPair No 203 8 Dimension expression 05
DimMax01 Int No 131 4 Maximum dimension 01
DimMax02 Int No 147 4 Maximum dimension 02
DimMax03 Int No 163 4 Maximum dimension 03
DimMax04 Int No 179 4 Maximum dimension 04
DimMax05 Int No 195 4 Maximum dimension 05
DimMin01 Int No 135 4 Minimum dimension 01
DimMin02 Int No 151 4 Minimum dimension 02
DimMin03 Int No 167 4 Minimum dimension 03
DimMin04 Int No 183 4 Minimum dimension 04
DimMin05 Int No 199 4 Minimum dimension 05
No list view yet, enter parameters
MA b 05/010

```

Figure 18. Display Structure List FDB.

## DOWN

### Syntax:

```

>>- Down ----->>
|
|  --- Cursor -----
|
|  --- CSR -----
|
|  --- Data -----
|
|  --- Half -----
|
|  --- Max -----
|
|  --- Page -----
|
|  --- n_lines -----

```

### Description:

Scroll the view of the data within the SDE window downwards towards the bottom of the file data.

Where no parameter is specified, the scroll amount will be the value specified in the **Scroll>** field in the top right corner of the window display.

Note that the first data record in any **multi record view** will **always** be preceded by its complete group of column header lines.

For **single record views**, the standard headers are always visible at the top of the display area and are not included as part of the scrollable text. Each non-header line within a single record view is identified as being a **field data line**.

Where the cursor is positioned at an offset within a column header line of a multi record view, the cursor is deemed to be located at the same offset within the data record that immediately follows the group of header lines. Therefore, DOWN CURSOR will operate successfully when the cursor is positioned within a header line.

UP and DOWN scroll commands will cause the window display area to be adjusted by a number of lines determined by the number of multi record view data records and shadow lines, or single record view field data lines. Other lines in the display area, i.e. **Header lines** and blank filler lines that precede a group of header lines or follow the "End of Data" record, are not included in this calculation.

Attempting to scroll down beyond the last entry (data record, shadow line or field data line) will make the "End of Data" record the first line of the scrolled display.

## Parameters:

CURSOR  
CSR

The data record, shadow line or field data line on which the cursor is positioned becomes the first line of the scrolled display.  
If the cursor is positioned on a header line, the data record or field data line immediately following the header line becomes the first line in the display area.  
If the cursor is positioned outside the display area or on the first line within the display area, then DOWN PAGE is executed instead.

DATA

Scroll down to display one page (display window depth) less one line of data.  
The last data record, shadow line or field data line in the current display area becomes the first line of the scrolled display.

HALF

Scroll down half a page of data.  
The data record, shadow line or field data line that is half way down the page of data in the current display area becomes the first record of the scrolled display.

MAX

Scroll down to display the last page of data.  
Where more than one page of data exists, the "End of Data" line becomes the last line of the scrolled display. Otherwise, the first line of data becomes the first line of the scrolled display.  
Equivalent to the **BOTTOM** command.

PAGE

Scroll down to display the next whole page of data.  
The data record, shadow line or field data line following the last line of the current display area becomes the first line of the scrolled display.

*n\_lines*

Scroll down a specified number of lines.  
The data record, shadow line or field data line that is *n\_lines* below the current line becomes the first line of the scrolled display.

## Examples:

down 5

Scroll the display area down 5 lines.

## See Also:

**LEFT**  
**RIGHT**  
**UP**

---

## DROP, DDROP

### Syntax:

```
>>- DROP --- struct_name -----><
>>- DDROP -- struct_name -----><
```





## EDIT

### Syntax:

[illegible]

**Description:**

Edit a structured data set within **SDE**.

An SDE edit display window is opened and focus is passed to the new window. Depending on the value on the **FORMAT** parameter, the SDE display is either a **multi record** window view or a **single record** window view for the first record selected.

If the Structure Definition Object (SDO) specified by the USING STRUCTURE parameter is not already in storage, it is loaded from the appropriate Structure Definition File SDF. Record Type Objects (RTO) within the specified SDO are then used to map the records in the data set.

For every record in the file, each RTO is checked until one is found whose criteria is matched by the record characteristics. This RTO is then used to map the record. By default, all fields within records that have an associated RTO are displayed as printable character, numeric data fields having first been converted to decimal.

If no SDO is specified or a record does not possess the characteristics required to satisfy any of the available RTO criteria, then FORMAT CHARACTER is applied to the record. i.e. the data is displayed as a single, variable (RECFM=V) or fixed (RECFM=F) length character field with field name "Record". No data conversion is performed.

If the VIEW parameter is specified with a non-generic argument, the DRECTYPE setting is initialised as the first *record type* parameter specified on the VIEW parameter. Otherwise, the DRECTYPE setting is initialised as the record type of the first visible record in the file.

Data that does not meet the specifications of the field definition is considered to be invalid and is represented by asterisks. If character field data contains non-printable text, then the CHAR representation of the data is non-enterable.

### Parameters:

```
fileid
```

The fileid of the data set containing records to be edited.  
Fileid may be the DSN of a sequential or VSAM data set or the DSN of a PDS/PDSE member.

USING (STRUCTURE) *struct\_name*

Identifies the SDO to be applied to the data records. *struct\_name* is the SDF fileid assigned to the SDO in a **CREATE STRUCTURE** command.

FORMAT  
FMT

Specifies the format in which record data will be displayed in the initial SDE view.

SINGLE	Single record format.
SNGL	

TABLE      Table format. (Default)

CHARACTER	Multi record view with all records mapped as a single, variable length character field with field name "Record". No data conversion is performed.
-----------	---

**HEX** Same as CHARACTER with the addition that the data is also displayed in Hex below the character display. Note that the Hex display occupies an additional 2 lines of data.

The data display format may be later altered using the **FORMAT** and/or **ZOOM** commands.

**VIEW** *record\_type*  
**VIEW** \*

Identifies one or more **record types** for which records will be displayed in the initial SDE view. Records that are associated with different record types are not selected for display and are displayed as shadow lines instead. If \* (asterisk) is specified, then records of all record types are displayed.

Records within the display may be later selected or deselected using the **VIEW** command.

Default is VIEW \*.

**PROFILE** *macro\_name*

Specifies the REXX SDE edit macro to be executed as the profile when the data set is edited.

*macro\_name* must exist in a library within the SDE macro path.

The PROFILE option only alters the profile used for the file currently being edited. It does not define the profile macro to be used for subsequent SDE edit.

The default is to execute a macro with member name SDEPROF if it exists.

**NOPROFILE**

Suppresses use of a profile macro when editing the file.

The NOPROFILE option only suppresses use of a profile for the file currently being edited. It does not suppress use of a profile macro for subsequent SDE edit.

**READONLY**

Perform read-only edit of the data.

Attempts to save changes made to the data during the SDE edit session will fail.

## Examples:

```
<sd edit CBL.SDE.MOD.ZJ2202 using CBL.CBLI.SDO(PRODL) view ARCX1,PRODMAS,PRODX
```

Edit data set "CBL.SDE.MOD.ZJ2202" using Structure Definition Object (SDO) "CBL.CBLI.SDO(PRODL)" to map records. Initially, all records of type ARCX1, PRODMAS and PRODX are displayed.

## See Also:

**BROWSE**  
**CREATE STRUCTURE**

---

## END

### Syntax:

```
>>-- END -----><
```

### Description:

Close an SDE EDIT or BROWSE view window. Set on **PF3** by default.

If the END operation is closing a file's only view window, the user will be prompted to save any changes to either the data or structure.

### Parameters:

END has no parameters.

**See Also:**

DROP  
QUIT

## EXCLUDE

**Syntax:**

```

>>+- EXclude +-+ string -----+
|          |          |          |          |          |
+- X -----+          +- NEXT ---+  +- CHARS ---+
|          |          |          |          |          |
+- ALL ---+  +- PREFIX ---+
|          |          |          |          |          |
+- FIRST ---+  +- SUFFIX ---+
|          |          |          |          |          |
+- LAST ---+  +- WORD ---+
|          |          |          |          |          |
+- PREV ---+

+- #ALL -----+  +- .ZFIRST ---+  +- .ZLAST ---+
>+-----+-----+-----+-----+-----+-----+<
+- pos1 -----+  +- .name1 -----+
|          |          |          |          |          |
+- pos2 ---+  +- .name2 ---+
|          |          |          |          |          |
|          |          |          |          |          |
|          |          |          |          |          |
|          |          |          |          |          |
|          |          |          |          |          |
+- ( +- field_col -----+ ) +-
|          |          |          |          |          |
+- field_col1:field_col2 -----+

```

**Description:**

Exclude data records in the current edit or browse view that satisfy a search for the specified character string or numeric value. EXCLUDE is not supported in [Single Record View](#).

Only data records that are of the default record type are included in the search. Record groups that are already excluded will remain excluded, therefore, execution of successive EXCLUDE commands has a cumulative effect. Records groups that are of a different record type or are NOTSELECTED or SUPPRESSED, are excluded from the search.

EXCLUDE employs the same string search and field highlighting methods as the [FIND](#) command. All occurrences of the search string or numeric value that are not excluded by the EXCLUDE command, are highlighted in the text. A record group which has been excluded via the EXCLUDE command, will contain the highlighted field matching the search string if it is subsequently redisplayed. (e.g. using [RESET EXCLUDED](#)) Enter the RESET FIND command to turn off the highlighting.

Use of EXCLUDE with no parameters repeats the last EXCLUDE command executed including all its specified parameters.

The [FORMAT](#) of the SDE display affects the execution of EXCLUDE.

**FORMAT CHAR and HEX**

A character compare for the supplied search string is performed against entire, unformatted data records.

*field\_col* and #ALL parameters are not applicable to these display formats and are ignored.

**FORMAT TABLE**

The search string is compared against **individual fields** that have been selected for display in the expanded data record. (See [SELECT](#))

Fields are searched from left to right in the order that they appear in the display. This is true regardless of the order in which field columns are specified on the EXCLUDE command, or the order in which fields are encountered within the expanded record.

If a field column is specified on the EXCLUDE command that is not part of the display (e.g. an expanded group field or a field that has been removed from display by a SELECT command), then an error message is returned.

For character data type fields, a string compare is performed. For numeric data type fields (binary, packed decimal, floating point, zoned, etc.), then the following will occur:

1. If *pos1*, *pos2* positional parameters are **not** specified, the search string is interpreted as a signed numeric value and an arithmetic compare for equality is performed against the field's formatted numeric value.

The length and data type of the numeric field, and the number of digits in the search value are not significant. e.g.

```
EXCLUDE 67
```

Excludes records containing a numeric field with value "67" (e.g. "0067", "67.00", "0.0670E+03") or a character field containing the string "67" (e.g. "167 Baker Street").

If the search string is non-numeric, then numeric data type fields are not searched. Therefore, to bypass searching numeric data type fields, explicitly set the search string to be character or hex using 'string', C'string' or X'string' formats respectively. e.g.

```
EXCLUDE '67'
EXCLUDE C'67'
EXCLUDE X'F6F7'
```

Excludes records only if a character field contains the string "67".

2. If *pos1*, *pos2* positional parameters are specified, the search string is interpreted as being a character string and a string compare is performed on the character representation of numeric data.

Although the range of positions may span a number of fields, the search is still performed against individual fields within the range. i.e. a match for the search string will not be found if the data spans a field boundary.

A match for the search string on just part of the character representation of a numeric field is considered to be a successful hit and the record is excluded. e.g.

```
EXCLUDE 476 21 100
```

Exclude records which contain a character or numeric field where character representation of the data contains the string "476". (e.g. a zoned decimal field with value "14760" or "-4762")

## Parameters:

*string*

The EXCLUDE search string. The search string may be any of the following:

- ◊ An unquoted numeric value. The search string is treated as a numeric value when a numeric field is searched in a SINGLE or TABLE format view. In all other cases a numeric search string is treated as a character string.
- ◊ An unquoted character string containing no commas or blanks. The search for the character string will be case-insensitive so that uppercase and lowercase characters are treated as being the same.
- ◊ A character string enclosed in single (') or double (") quotation marks. The search string may contain embedded commas and blanks and the character string will be case-insensitive.

Two adjacent quotation mark characters that are embedded in a search string which is enclosed by the same quotation mark characters, will be treated as a single occurrence of the character. e.g.

```
EXCLUDE 'Jim O''Brien'
```

Find the character string "Jim O'Brien".

- ◊ A character string enclosed in single (') or double (") quotation marks with the prefix (or suffix) C. This is equivalent to specifying a quoted search string but that the string search will be case-sensitive. (e.g. C'Book')
- ◊ A hexadecimal string enclosed in single (') or double (") quotation marks with the prefix (or suffix) X.

If a no search string is specified, the default search string used is that specified on the most recent FIND, CHANGE, or EXCLUDE command executed against the current file. **EXCLUDE ALL** with no search string is a special case and will exclude all data records of the default record type.

ALL

Search forwards from the top of the file data (i.e. the first position of the first data record) and excluded all records containing an occurrence of the string. EXCLUDE ALL with no other parameters excludes all records of the default record type.

FIRST

Search forwards from the top of the file data (i.e. the first position of the first data record) to exclude the first record of the default record type to contain an occurrence of the search string.

LAST

Search backwards from the bottom of the file data (i.e. the last position of the last data record) to exclude the last record of the default record type to contain an occurrence of the search string.

NEXT

Search forwards from the current cursor location to exclude the next record of the default record type to contain an occurrence of the search string. If the cursor is not within the window's data display area, the search begins at the first position of the first visible or excluded record within the display area that is of the default record type.

**PREV** Search backwards from the current cursor location to exclude the previous record of the default record type to contain an occurrence of the search string. If the cursor is not within the window's data display area, the backwards search begins at the first position of the first visible or excluded record within the display area that is of the default record type.

**CHARS** For non-numeric search strings only, CHARS indicates that a successful match occurs if the search string is found anywhere within the data being searched.

**PREFIX** For non-numeric search strings only, PREFIX indicates that a successful match only occurs if the search string is found at the start of a word within the data being searched. i.e. the matched text must precede an alphanumeric character and either be preceded by a non-alphanumeric character or be at the start of a line or field.

**SUFFIX** For non-numeric search strings only, SUFFIX indicates that a successful match only occurs if the search string is found at the end of a word within the data being searched. i.e. the matched text must be preceded by an alphanumeric character and either precede a non-alphanumeric character or be at the end of a line or field.

**WORD** For non-numeric search strings only, WORD indicates that a successful match only occurs if the search string is found to be a complete word within the data being searched. i.e. the matched text must either be preceded by a non-alphanumeric character or be at the start of a line or field, and either precede a non-alphanumeric character or be at the end of a line or field.

*pos1* The first position of a range of positions within the data record to be searched.

For TABLE format, this is a position in the expanded record. Only those fields, or parts of fields, that fall within the position range will be searched. Fields will be searched in the order that they occur within the display area.

*pos1* must be a number greater than or equal to 1 and less than or equal to the maximum length of the data records (or length of the expanded records in the case of a TABLE format view.)

For all display formats, the string is treated as being non-numeric and the search is actioned on the character representation of the record data.

*pos2* The last position of a range of positions within the data record to be searched.

if *pos1* is greater than *pos2* in the data record, then the values are swapped.

If *pos2* is greater than the maximum length of the data records (or length of the expanded records in the case of a TABLE format view) then *pos2* is set equal to the maximum (or expanded) record length.

Default is *pos1* so defining a search range of width 1.

**#ALL** Search all field columns in the current TABLE format display.

*field\_col* An individual field column to be searched within a TABLE format display.

The field column may be identified by its reference number (e.g. #6) or its name (e.g. JobID). If a field name is used, enclosing parentheses are **mandatory**. Field names are automatically converted to a field reference and Referencing the same field column more than once will not cause an error.

If the field is an array, then all elements of the array are searched. To search an individual element of an array, the element occurrence should be specified in parentheses as a subscript to the field reference/name. e.g. #13(3) for the 3rd element of a single dimension array. An entry must exist for each dimension of the array. e.g. RoomSize(6,2,4) - a three dimensional array.

Specification of multiple field columns must either be enclosed in parentheses and/or separated by commas. The field columns may be specified in any order, however, the data is always searched from the first column in the display to the last.

Field column search specifications may be a combination of individual field columns and columns ranges. e.g. (JobID #6 Tax\_Reference #12 #15:#20).

A search is only performed on field columns that are selected for display. Therefore, any field column specified on the EXCLUDE command that has not been selected for display, will be ignored.

*field\_col1:field\_col2* The first and last fields of a range of field columns to be searched within a TABLE format display.

*field\_col1* and *field\_col2* must be separated by ":" (colon) and if *field\_col1* occurs after *field\_col2* in the data record, then the values are swapped. *field\_col1* and *field\_col2* have the same specifications as *field\_col*.

Specification of multiple field column ranges must either be enclosed in parentheses and/or separated by commas. Field column search specifications may be a combination of individual field columns and field columns ranges. e.g. (FirstName:LastName, #2, Salary:Bonus, EmpNo, #10:#15). If field names are used, enclosing parentheses are **mandatory**.

`.name1`

A label name identifying the first record of a range of data records to be searched. The preceding "." (dot) is mandatory. Default is `.ZFIRST`.

`.name2`

A label name identifying the last record of a range of data records to be searched. The preceding "." (dot) is mandatory. If `.name2` occurs before `.name1` in the display, then the order is reversed. If EXCLUDE PREV is executed and `.name1` is specified, the default is `.ZFIRST`. Otherwise the default is `.ZLAST`.

## Examples:

`exclude last 100`

Exclude the last record in the display (of the default record type) that contains a numeric field with value 100 or a character field containing the string "100".

`ex c'Spec' prefix (#10:#20, TitleTrack)`

Exclude the first record in the display (of the default record type) that contains the exact string "Spec" as a word prefix within any of the selected character fields.

`x all`

Exclude all records of the default record type.

## See Also:

CHANGE  
FIND  
RFIND  
SELECT

---

## EXTRACT

### Description:

See [SET/QUERY/EXTRACT Options](#).

---

## FIND

### Syntax:

```

>>+- Find +-+ string -----+ +- NEXT --+ +- CHARS --+
|         |         |         |         |         |
+- / -----+ +- ALL ---+ +- PREFIX --+ +- EX --+
|         |         |         |         |         |
+- FIRST --+ +- SUFFIX --+ +- NX --+
|         |         |         |         |         |
+- LAST --+ +- WORD ---+ +- X ---+
|         |         |         |         |         |
+- PREV --+

+- #ALL -----+ +- .ZFIRST ---- .ZLAST --+
>+-----+-----+-----+-----+-----+-----+<<
+- pos1 -----+ +- .name1 -----+
|         |         |         |         |         |
|         +- pos2 --+         +- .name2 --+
|         |         |         |         |         |
|         +-----+-----+-----+
|         |         |         |         |
|         +- , -----+
|         v
+- ( +- field_col -----+ ) +-
|         |         |         |         |
+- field_col1:field_col2 -----+

```

## Description:

Search data records in the current edit or browse view for the specified character string or numeric value. Only data records that are of the **default record type** (visible and EXCLUDED records) are included in the search. Records groups that are of a different record type or are NOTSELECTED or SUPPRESSED, are excluded from the search.

If the specified occurrence (all, first, last, next or previous) of the search string or numeric value is found within a record, then:

1. If the record is EXCLUDED, it is made visible.
2. The cursor is positioned at the beginning of the string or numeric field.
3. If necessary, scrolling occurs to display the found data.

All occurrences of the search string or numeric value are highlighted in the text. (Enter the **RESET FIND** command to turn off the highlighting.)

Use of FIND with no parameters is equivalent to **RFIND** (assigned to function key **PF5** by default) and repeats the last find command executed, including all its specified parameters.

The **FORMAT** of the SDE display affects the execution of FIND.

## FORMAT CHAR and HEX

A character compare for the supplied search string is performed against entire, unformatted data records.

*field\_col* and #ALL parameters are not applicable to these display formats and are ignored.

## FORMAT SINGLE and TABLE

The search string is compared against **individual fields** that have been selected for display in the expanded data record. (See **SELECT**)

Fields are searched from left to right in the order that they appear in the display. This is true regardless of the order in which field columns are specified on the FIND command, or the order in which fields are encountered within the expanded record.

If a field column is specified on the FIND command that is not part of the display (e.g. an expanded group field or a field that has been removed from display by a SELECT command), then an error message is returned.

For character data type fields, a string compare is performed. For numeric data type fields (binary, packed decimal, floating point, zoned, etc.), then the following will occur:

1. If *pos1*, *pos2* positional parameters are **not** specified, the search string is interpreted as a signed numeric value and an arithmetic compare for equality is performed against the field's formatted numeric value.

The length and data type of the numeric field, and the number of digits in the search value are not significant. e.g.

```
FIND 67
```

Finds numeric fields with value "67" (e.g. "0067", "67.00", "0.0670E+03") and character fields containing the string "67" (e.g. "167 Baker Street").

If the search string is non-numeric, then numeric data type fields are not searched. Therefore, to bypass searching numeric data type fields, explicitly set the search string to be character or hex using 'string', C'string' or X'string' formats respectively. e.g.

```
FIND '67'
FIND C'67'
FIND X'F6F7'
```

Finds only character fields containing the string "67".

2. If *pos1*, *pos2* positional parameters are specified, the search string is interpreted as being a character string and a string compare is performed on the character representation of numeric data.

Although the range of positions may span a number of fields, the search is still performed against individual fields within the range. i.e. a match for the search string will not be found if the data spans a field boundary.

A match for the search string on just part of the character representation of a numeric field is considered to be a successful hit and the whole numeric field is highlighted. e.g.

```
FIND 476 21 100
```

Finds character and numeric fields where character representation of the data contains the string "476". (e.g. a zoned decimal field with value "14760" or "-4762")

Where a character string or numeric value is found in a numeric field, the entire field is highlighted and the cursor positioned at the start of the field.

## Parameters:

*string*

The FIND search string. The search string may be any of the following:

- ◊ An unquoted numeric value. The search string is treated as a numeric value when a numeric field is searched in a SINGLE or TABLE format view. In all other cases a numeric search string is treated as a character string.
- ◊ An unquoted character string containing no commas or blanks. The search for the character string will be case-insensitive so that uppercase and lowercase characters are treated as being the same.
- ◊ A character string enclosed in single (') or double (") quotation marks. The search string may contain embedded commas and blanks and the character string will be case-insensitive.

Two adjacent quotation mark characters that are embedded in a search string which is enclosed by the same quotation mark characters, will be treated as a single occurrence of the character. e.g.

```
FIND 'Jim O''Brien'
```

Find the character string "Jim O'Brien".

- ◊ A character string enclosed in single (') or double (") quotation marks with the prefix (or suffix) C. This is equivalent to specifying a quoted search string but that the string search will be case-sensitive. (e.g. C'Book')
- ◊ A hexadecimal string enclosed in single (') or double (") quotation marks with the prefix (or suffix) X.

If a no search string is specified, the default search string used is that specified on the most recent FIND, CHANGE, or EXCLUDE command executed against the current file. **FIND ALL** with no search string is a special case and is equivalent to executing RESET FIND EXCLUDED.

ALL

If none of the records to be scanned are EXCLUDED or NX is specified, then FIND ALL is the same as FIND FIRST except that a message is displayed providing the number of occurrences of the string/value found in the data. Unlike FIND FIRST, **all** excluded records that contain an occurrence of the string/value are made visible if NX is not specified. FIND ALL with no other parameters is equivalent to RESET FIND EXCLUDED.

FIRST

Search forwards from the top of the file data (i.e. the first position of the first data record) to find the first occurrence of the string.

LAST

Search backwards from the bottom of the file data (i.e. the last position of the last data record) to find the last occurrence of the string.

NEXT

Search forwards from the current cursor location to find the next occurrence of the string. If the cursor is not within the window's data display area, the search begins at the first position of the first visible or excluded record within the display area that is of the default record type.

PREV

Search backwards from the current cursor location to find the previous occurrence of the string. If the cursor is not within the window's data display area, the backwards search begins at the first position of the first visible or excluded record within the display area that is of the default record type.

CHARS

For non-numeric search strings only, CHARS indicates that a successful match occurs if the search string is found anywhere within the data being searched.

PREFIX

For non-numeric search strings only, PREFIX indicates that a successful match only occurs if the search string is found at the start of a word within the data being searched. i.e. the matched text must precede an alphanumeric character and either be preceded by a non-alphanumeric character or be at the start of a line or field.

SUFFIX

For non-numeric search strings only, SUFFIX indicates that a successful match only occurs if the search string is found at the end of a word within the data being searched. i.e. the matched text must be preceded by an alphanumeric character and either precede a non-alphanumeric character or be at the end of a line or field.

WORD

For non-numeric search strings only, WORD indicates that a successful match only occurs if the search string is found to be a complete word within the data being searched. i.e. the matched text must either be preceded by a non-alphanumeric character or be at the start of a line or field, and either precede a non-alphanumeric character or be at the end of a line or field.

EX  
X

Search EXCLUDED data records only.  
FIND does not search records that are NOTSELECTED or SUPPRESSED.



NX

Search only visible data records (i.e. not EXCLUDED).  
FIND does not search records that are NOTSELECTED or SUPPRESSED.

*pos1*

The first position of a range of positions within the data record to be searched.

For SINGLE and TABLE formats, this is a position in the expanded record. Only those fields, or parts of fields, that fall within the position range will be searched. Fields will be searched in the order that they occur within the display area.

*pos1* must be a number greater than or equal to 1 and less than or equal to the maximum length of the data records (or length of the expanded records in the case of SINGLE or TABLE format views.)

For all display formats, the string is treated as being non-numeric and the search is actioned on the character representation of the record data.

*pos2*

The last position of a range of positions within the data record to be searched.

if *pos1* is greater than *pos2* in the data record, then the values are swapped.

If *pos2* is greater than the maximum length of the data records (or length of the expanded records in the case of SINGLE or TABLE format views) then *pos2* is set equal to the maximum (or expanded) record length.

Default is *pos1* so defining a search range of width 1.

#ALL

Search all field columns in the current TABLE or SINGLE format display.

*field\_col*

An individual field column to be searched within a TABLE or SINGLE format display.

The field column may be identified by its reference number (e.g. #6) or its name (e.g. JobID). If a field name is used, enclosing parentheses are **mandatory**. Field names are automatically converted to a field reference and Referencing the same field column more than once will not cause an error.

If the field is an array, then all elements of the array are searched. To search an individual element of an array, the element occurrence should be specified in parentheses as a subscript to the field reference/name. e.g. #13(3) for the 3rd element of a single dimension array. An entry must exist for each dimension of the array. e.g. RoomSize(6,2,4) - a three dimensional array.

Specification of multiple field columns must either be enclosed in parentheses and/or separated by commas. The field columns may be specified in any order, however, the data is always searched from the first column in the display to the last.

Field column search specifications may be a combination of individual field columns and columns ranges. e.g. (JobID #6 Tax\_Reference #12 #15:#20).

A search is only performed on field columns that are selected for display. Therefore any field column specified on the FIND command that has not been selected for display, will be ignored.

*field\_col1:field\_col2*

The first and last fields of a range of field columns to be searched within a TABLE or SINGLE format display.

*field\_col1* and *field\_col2* must be separated by ":" (colon) and if *field\_col1* occurs after *field\_col2* in the data record, then the values are swapped. *field\_col1* and *field\_col2* have the same specifications as *field\_col*.

Specification of multiple field column ranges must either be enclosed in parentheses and/or separated by commas. Field column search specifications may be a combination of individual field columns and field columns ranges. e.g. (FirstName:LastName, #2, Salary:Bonus, EmpNo, #10:#15). If field names are used, enclosing parentheses are **mandatory**.

*.name1*

A label name identifying the first record of a range of data records to be searched. The preceding "." (dot) is mandatory. Default is .ZFIRST.

*.name2*

A label name identifying the last record of a range of data records to be searched. The preceding "." (dot) is mandatory. If *.name2* occurs before *.name1* in the display, then the order is reversed. If FIND PREV is executed and *.name1* is specified, the default is .ZFIRST. Otherwise the default is .ZLAST.

## Examples:

find 22.3

Search all fields in the display belonging to the default record type for the next occurrence of the search string/value 22.3. Numeric fields are tested for numeric value 22.3, character fields are tested for character string "22.3" anywhere within the field.

find all 'ask' suffix ex (DsName:ProcLib, JobStep)

Search selected character fields in the display belonging to excluded records of the default record type for all occurrences of the word suffix string "ask".

### See Also:

CHANGE  
EXCLUDE  
RFIND  
SELECT

---

## FLIP

### Syntax:

```
>>-- FLIP -----><
```

### Description:

Flip records that are of the **default record type** so that visible data records become EXCLUDED and EXCLUDED records are made visible.

### Parameters:

FLIP has no parameters.

### See Also:

LESS  
MORE  
MORE

---

## FORMAT

### Syntax:

```
>>-- FOrmat --+-- Character -----+-----><
      |
      +-- Hex -----+
      |
      +-- Single -----+
      |
      +-- Sngl -----+
      |
      +-- Tabl -----+
```

### Description:

Sets the display format for the current SDE BROWSE or EDIT window.

### Parameters:

#### Character

Multi record view with all records mapped as a single, character field with field name "Record". No data conversion is performed.

#### Hex

Same as CHARACTER with the addition that the data is also displayed in Hex below the character display. Note that the Hex display occupies an additional 2 lines of data.

#### Single

#### Sngl

Single record format with data types formatted according to the record structure. Each field occupies a separate line. Vertical scrolling transfers focus between fields. Horizontal scrolling transfers focus between records.

Table

Multi-record table format (Default) with data types formatted according to the record structure. Each field occupies a separate column. Vertical scrolling transfers focus between records. Horizontal scrolling transfers focus between fields.

## HEX

### Syntax:

```
>>-- HEX  --+-- ON  ---+----->>
          |      |
          +-- OFf  ---+
```

**Description:**

Sets the hexadecimal display format on or off. Issued without the **ON** or **OFF** parameter the display format is toggled.

If the current display format has been set using **FORMAT CHAR**, or **FORMAT HEX**, then **HEX ON** is the equivalent of issuing **FORMAT HEX**.

If the current display format has been set using **FORMAT SINGLE**, or **FORMAT TABLE**, then for each line displayed **HEX ON** will add two further lines to show the hexadecimal representation of the raw (unformatted) data below each formatted field. e.g.

SRCRecOff	SRCRecLen	SRCMbr
BN 124:4	BN 125:4	AN 137:8
<----->	<----->	<----->
152	80	APEZLINK
0009	0005	CDCEDCDD
0008	0000	17593952
144	80	SDFECMD1
0009	0005	ECCCCDCF
0000	0000	24563441

### Parameters:

ON  
OFF

HEX display format is set ON or OFF.

**INSERT**

### Syntax:

```

>>- Insert ----->
      +- record_type -+
                        |
                        | +- , -----+
                        | v             |
                        +- (-- field_col +- ) -+

>-----<
      +- , -----+
      | v         |
      +- Values(-- field value +- ) -+
                +- n_lines -+

```

**Description:**

Insert one or more new records following the focus line, optionally providing explicit values to be inserted into specified fields.

Fields within the inserted records do not require explicitly assigned values. This occurs either when no list of field values is specified or when the default record type contains more fields than there are values in the specified list of field values.

If fields are not explicitly assigned values (either because no list of field values is specified or because the default record type contains more fields than there are values in the specified list of field values) then they are assigned default values as follow:

- 0 (zero) for numeric fields.

- Nulls for bit and hex fields.
- Blanks for all other field types.

If an expression is used to identify the record type then an attempt is made to populate any fields specified in that expression with values which satisfy the expression.

INSERT with no parameters inserts one record of the current **default record type** with default values in all fields.

## Parameters:

*record\_type*

The record type of the record(s) to be inserted. If *record\_type* is not specified, then the **default record type** is used.

*field\_col*

An individual field column within a list of field columns for which a corresponding *field\_value* is to be inserted.

The field column may be identified by its reference number (e.g. #1) or its name (e.g. SeqNUM). Specification of multiple field columns must be separated by commas and enclosed in parentheses.

If the field is a **struct** or a **union**, then an error message is returned. If the field is an array, then an individual array element must be specified in parentheses as a subscript to the field. e.g. #13(3) for the 3rd element of a single dimension array. A subscript entry must exist for each dimension of the array. e.g. RoomSize(6,2,4) - an element within a three dimensional array.

If a list of field columns is not specified then as many of the set of selected fields in the current view as there are values in the list of field values, are used as the default. In this case, the order of the field columns is equal to the order in which fields are displayed in the current view.

*field\_value*

An individual field value within a list of field values to be inserted in a corresponding *field\_col* entry in the field column list.

If one or more *field\_col* entries are specified (i.e. a field list is specified), then there must be the same number of *field\_value* values to correspond with each *field\_col* entry.

If a field value contains special characters, blanks or commas, then it should be enclosed in quotes.

*n\_lines*

Number of identical records to be inserted.  
Default is 1.

## Examples:

```
insert (SeqNum, FirstName, LastName, JobTitle, Salary) Val(283, Jacob, Smith, 'Systems Engineer', 35,950.00)
```

Insert a record of the default record type populating the specified list of fields with the explicit values in the corresponding list of field values. All other fields are populated with default values.

```
insert 2 Val(18.3, "Michael O'Leary", -12.333)
```

Insert 2 records of the default record type populating the first 3 selected fields in the display with the explicit values specified in the list of field values. All other fields are populated with default values.

---

## LEFT

### Syntax:

```
>>- Left  ---+-----+-----><
|
|  -- Cursor -----+
|
|  -- CSR -----+
|
|  -- Data -----+
|
|  -- Half -----+
|
|  -- Max -----+
|
|  -- Page -----+
|
|  -- n_cols -----+
|
```

### Description:

In **multi record view**, scroll to the left the display of field and header lines belonging to records that are of the **default record type**. The display of all other visible records, header lines and shadow lines remains unchanged.

In **single record view**, LEFT will display the fields belonging to a visible data record that has a lower line number than the record that is in the current view. LEFT CURSOR/DATA/HALF/PAGE are not applicable in single record view.

LEFT is assigned to **PF10** by default. Any characters specified on the command line when the PFKey is hit will be concatenated to the command and treated as a parameter string.

Where no parameter is specified, the scroll amount will be the value specified in the "**Scroll>**" field.

## Multi Record View Scrolling

Columns may have the HOLD flag set on by the **SELECT** command. These columns are static in the display and, like the prefix area and Lrecl column, are unaffected by LEFT and RIGHT scrolling. Columns that have the HOLD flag set off are referred to as **floating** columns.

The following interpretation of cursor location applies so that LEFT CURSOR can operate successfully on the appropriate data:

- Cursor is located at an offset within a column header line.  
The cursor position is interpreted as being at the same offset within the data record that immediately follows the group of header lines.
- Cursor is located within a column of type character (AN) but beyond the width of the character data.  
The cursor position is interpreted as being the last position of the displayed character data.
- Cursor is located immediately to the left of a column of any type.  
The cursor position is interpreted as being the first position of the data displayed within that column.

Where the last column to be displayed is not of type character (AN), the magnitude by which the display is scrolled to the left is always reduced to display all field data belonging to the last column. In addition, the display area must contain all column header text belonging to the first column and, where the first column is not of type character (AN), it must also contain all field data belonging to the first column. Therefore, to accommodate this, the magnitude by which the display is scrolled to the left may be further reduced.

Because of these adjustments, it is possible that execution of a LEFT command may result in no change to the displayed data, in which case LEFT PAGE is executed instead.

No adjustment is made for scrolling left into an offset within a character data column. i.e. where the last column of the display is of type character, the last position of the column display need not be the last character of the character data. If the last column to be displayed is of type character and the display width is too small to accommodate all the column's data and header text, the column will be truncated. However, due to the other adjustments made for scrolling left, it may be impossible for some characters within the character data to occupy the last position of the column display. In order to overcome this, the user would have to alter the display area dimensions.

Attempting to scroll left beyond the start of the record will make the first data column the first column of the display.

## Parameters:

CURSOR  
CSR

The **focus column** becomes the last column of the scrolled display.

In addition to this, if the focus column is type character (AN) and the cursor is positioned on a character that is at an offset into the focus column data, then an attempt is made to make that character occupy the last position of the scrolled display.

If any of the following conditions are true, then LEFT PAGE is executed instead:

- ◇ Cursor is positioned anywhere other than within a floating column in a visible data record.
- ◇ Focus column is **not** type AN and is already the last column of the display.
- ◇ Focus column is type AN, is already the last column of the display and cursor position is at the last position of the displayed column data.

DATA

Scroll left so that the first floating column in the current display area becomes the last column of the scrolled display. If this column is type character (AN) and the first position of the display falls on a character that is at an offset into the column, then an attempt is made to make that character occupy the last position of the scrolled display.

HALF

Scroll a number of columns so that the column situated half way along the width of the current display of floating columns, becomes the last column of the scrolled display. If this column is type character (AN) and the half display position falls on a character that is at an offset into the column, then an attempt is made to make that character occupy the last position of the scrolled display.

MAX

Scroll left to display the first floating column of data in a multi record view.

In single record view, the first visible data record is displayed.

PAGE

Scroll left so that the floating column of data to the left of the first floating column in the current display, becomes the last column of the scrolled display.

If this column is type character (AN) and the first position of the display minus one falls on a character that is at an offset into the column, then an attempt is made to make that character occupy the last position of the scrolled display.

*n\_cols*

In a multi record view, scroll left a specified number of floating columns.  
The floating column of data that is *n\_cols* to the left of the first floating column becomes the new first floating column of the scrolled display.

In single record view, the visible data record that is *n\_cols* records before the record currently in view, gets displayed.

## Examples:

*left page*

Scroll records of the default record type in a multi record view display area, left by a full display area width minus the width of any columns flagged with HOLD.

## See Also:

DOWN  
RIGHT  
UP

---

## LESS

### Syntax:

```
>>--- LESS ---- where_clause -----><
```

### Description:

Exclude any visible records that are of the **default record type** and also satisfy the specified *where\_clause* criteria. Records that are already EXCLUDED are unaffected by the LESS command. The current line is also unaffected unless the SHADOW option for EXCLUDED lines is set off and the current line is one which is excluded by the LESS command. In this case the line following the current line becomes the new current line.

If LESS is executed with no parameters, then all visible records that are of the default record type are EXCLUDED.

The record type used on a LESS command becomes the new value of the **DRECTYPE** option.

### Parameters:

*where\_clause*

*where\_clause* syntax is described fully in documentation for the **WHERE** command.

### Examples:

```
less      #14 >= 255
Exclude any records that are of the default record type where numeric field reference number 14 is greater than or equal to 255.

less      OrderDate <= PaymentReceivedDate
Exclude any records that are of the default record type where the contents of field "OrderDate" is less than or equal to the contents of field "PaymentReceivedDate".
```

## See Also:

FLIP  
MORE  
WHERE



LAST	Search backwards from the last record in the file with the selected record type to locate the last record that matches the specified <i>where_clause</i> expression.
NEXT	Search forwards from the <b>focus line</b> to locate the next record that matches the specified <i>where_clause</i> expression.
PREV	Search backwards from the <b>focus line</b> to locate the previous record that matches the specified <i>where_clause</i> expression.
EX X	Locate EXCLUDED data records only. LOCATE does not search records that are NOTSELECTED or SUPPRESSED.
NX	Locate only visible data records (i.e. not EXCLUDED). LOCATE does not search records that are NOTSELECTED or SUPPRESSED.
<i>where_clause</i>	An expression which defines search criteria used to identify the required data record(s).  The <i>where_clause</i> expression involves fields belonging to the selected record type. <i>where_clause</i> syntax is described fully in documentation for the <b>WHERE</b> command.  By default, data records of the selected record type that have been EXCLUDED are included in the locate scan and are made visible if they satisfy the <i>where_clause</i> .

## Examples:

```
locate 3
    Scroll the display area so that the line that is 3 lines below the focus line becomes the new current line.

-18
    Scroll the display area so that the line that is 18 lines above the focus line becomes the new current line.

:18
    Scroll the display area so that the visible record at line number 18 becomes the new current line.

locate prev (LastName = 'Jones' & Salary > '21950') | Dept \>> 'Tech'
    Scroll the display area so that the first record before the focus line that is of the default record type and also matches the
    where_clause expression, becomes the new current line.
```

## See Also:

DOWN  
FIND  
UP  
WHERE

---

## MORE

### Syntax:

```
>>--- MORE ---- where_clause -----><
```

### Description:

Make visible any EXCLUDED records that are of the **default record type** and also satisfy the specified *where\_clause* criteria. The current line of the display and any records that are already visible are unaffected by the MORE command.

If MORE is executed with no parameters, then all EXCLUDED records that are of the default record type are made visible.

The record type used on a MORE command becomes the new value of the **DRECTYPE** option.

### Parameters:

*where\_clause*  
*where\_clause* syntax is described fully in documentation for the **WHERE** command.



**Examples:**

```
more      #5 >> X'41'
           Make visible any EXCLUDED records that are of the default record type where field reference number 5 begins with X'41'
           (ASCII character 'A').

more      LastName = C'Evans'
           Make visible any EXCLUDED records that are of the default record type where the contents of field "LastName" is equal
           "Evans".
```

**See Also:**

FLIP  
LESS  
WHERE

---

**MACRO****Syntax:**

```
>>--- MACRO ---- macro_name ---+-----+-----><
                                |         |
                                +- parm_string -+
```

**Description:**

Force SDE to execute the REXX macro specified by *macro\_name* as opposed to an SDE command of the same name.

By default, SDE first checks a user supplied token for a recognised SDE command. If the token is not recognised as a command, an attempt is made to find a macro with the same name. Therefore, when invoking a macro, the command verb MACRO may not be necessary.

Any text specified following the macroname is passed to the macro as a parameter string. The specified macro is read into memory from disk, executed and removed from memory on completion.

If the macro cannot be located, an error message is issued and the MACRO command fails.

**Parameters:**

*macro\_name*  
Name of the macro to be executed.

If *macro\_name* is a full fileid containing file name, path, etc., then the macro is loaded from the specified location. If only a file name is specified, each directory in the macro path is searched for a matching macro name.

*parm\_string*  
Text to be passed to the macro as a parameter string.

**Examples:**

```
macro      cbl.dist.cbli.site.cble(xarc)      XXFLD 22 33
           Execute the macro XARC in library CBL.DIST.CBLI.SITE.CBLE with parameters "XXFLD 22 33".

macro      select
           Execute the user macro SELECT from a library in the macro path, not the SDE command SELECT.
```

---

**QUERY****Description:**

See [SET/QUERY/EXTRACT Options](#).

---

## QQUIT

### Syntax:

```
>>--- QQuit -----><
```

### Description:

Close an SDE edit view window without prompting to save any changes made to the data. i.e. all changes will be discarded.

### Parameters:

QQUIT has no parameters.

### See Also:

END

---

## RCHANGE

### Syntax:

```
>>-- RChange -----><
```

### Description:

Repeat the find and replace performed by the last **CHANGE** command. The RCHANGE search will be executed on records that are of the same **record type** as that used by the last CHANGE command.

Where the last CHANGE issued was CHANGE LAST or CHANGE PREV, RCHANGE will perform a CHANGE PREV operation, otherwise RCHANGE performs a CHANGE NEXT operation. i.e. Search forwards (NEXT) or backwards (PREV) from the current cursor location to find the next or previous occurrence of the string/value respectively.

If the cursor is not within the window's data display area, the forwards or backwards search begins at the first position of the first visible or excluded record within the display area that is of the record type used by the last CHANGE.

RCHANGE is assigned to function key **PF6** by default.

### Parameters:

RCHANGE has no parameters.

### See Also:

CHANGE  
EXCLUDE  
FIND  
RFIND  
Find and Replace Data

---

## REDO

### Syntax:

```
>>--- REDO -----><
```



EXCLUDED  
X

FIND

LABEL

## RFIND

### Syntax:

**Description:**

RFIND is assigned to function key **PF5** by default.

### Parameters:

### See Also:

**RIGHT**

### Syntax:

65

## Description:

In **multi record view**, scroll towards the right the display of field and header lines belonging to records that are of the **default record type**. The display of all other visible records, header lines and shadow lines remains unchanged.

In **single record view**, RIGHT will display the fields belonging to a visible data record that has a higher line number than the record that is in the current view. RIGHT CURSOR/DATA/HALF/PAGE are not applicable in single record view.

RIGHT is assigned to **PF11** by default. Any characters specified on the command line when the PFKey is hit will be concatenated to the command and treated as a parameter string.

Where no parameter is specified, the scroll amount will be the value specified in the "**Scroll>**" field.

## Multi Record View Scrolling

Columns may have the HOLD flag set on by the **SELECT** command. These columns are static in the display and, like the prefix area and Lrecl column, are unaffected by LEFT and RIGHT scrolling. Columns that have the HOLD flag set off are referred to as **floating** columns.

The following interpretation of cursor location applies so that RIGHT CURSOR can operate successfully on the appropriate data:

- Cursor is located at an offset within a column header line.  
The cursor position is interpreted as being at the same offset within the data record that immediately follows the group of header lines.
- Cursor is located within a column of type character (AN) but beyond the width of the character data.  
The cursor position is interpreted as being the last position of the displayed character data.
- Cursor is located immediately to the left of a column of any type.  
The cursor position is interpreted as being the first position of the data displayed within that column.

Where the first column to be displayed is not of type character (AN), the magnitude by which the display is scrolled to the right is always reduced to display all field data belonging to the first column. Similarly, if the last column to be displayed is not of type character and the display width is too small to accommodate all the column's data and header text, the column will not be displayed.

No adjustment is made when scrolling right into an offset within a character data column. i.e. where the first column of the display is of type character, the first position of the column display need not be the first character of the character data. If the last column to be displayed is of type character and the display width is too small to accommodate all the column's data and header text, the column will be truncated.

Attempting to scroll right beyond the last data column will make the last data column the first column of the display. Furthermore, if the last data column is of type character (AN), then scrolling beyond this column will make the last character of the data the first character of the display.

## Parameters:

CURSOR  
CSR

The **focus column** becomes the first column of the scrolled display.

In addition to this, if the focus column is type character (AN) and the cursor is positioned at an offset into the focus column, then the character data at the cursor offset will occupy the first position within the first column of the scrolled display.

If any of the following conditions are true, then RIGHT PAGE is executed instead:

- ◇ Cursor is positioned anywhere other than within a floating column in a visible data record.
- ◇ Focus column is **not** type AN and is already the first floating column of the display.
- ◇ Focus column is type AN, is already the first floating column of the display and cursor position is at the first position of the displayed column data.

DATA

Scroll right so that the last floating column of the current display area becomes the first floating column of the scrolled display.

If this column is type character (AN) and the last position of the display falls on a character that is at an offset into the column data, then the character data at that offset will occupy the first position within the first column of the scrolled display.

HALF

Scroll a number of columns so that the column situated half way along the width of the current display of floating columns, becomes the first floating column of the scrolled display.

If this column is type character (AN) and the half display position falls on a character that is at an offset into the column data, then the character data at that offset will occupy the first position within the first floating column of the scrolled display.

MAX

Scroll right to display the last column of data in a multi record view. Where the display area is able to contain all data columns in the data record, the first floating column becomes the first floating column of the scrolled display. Otherwise,

the last column of data becomes the last column of the scrolled display.

In single record view, the last visible data record is displayed.

**PAGE** Scroll right so that the column of data to the right of the last floating column of the current display area, becomes the first floating column of the scrolled display.  
If this column is type character (AN) and the last position of the display plus one falls on a character that is at an offset into the column data, then the character data at that offset will occupy the first position within the first floating column of the scrolled display.

**n\_cols** In a multi record view, scroll right a specified number of floating columns.  
The floating column of data that is *n\_cols* to the right of the first floating column becomes the new first floating column of the scrolled display.

In single record view, the visible data record that is *n\_cols* records after the record currently in view, gets displayed.

## Examples:

**right half** Scroll records of the default record type in a multi record view display area, right by half a display area width minus the width of any columns flagged with HOLD.

## See Also:

DOWN  
LEFT  
UP

---

## SAVE

### Syntax:

```
>>-- Save -----><
```

### Description:

Save the current file to disk.

### Parameters:

SAVE has no parameters.



---

## SORT

### Syntax:

```
>>-- SORT ---- KEY -----><
```

### Description:

Sort records in the SDE edit view.

SORT KEY should be issued before attempting to save changes made to a structured VSAM KSDS data set in order to avoid possible VSAM WRITE key sequence errors.

### Parameters:

KEY

Supported for VSAM KSDS data sets only, sorts records into ascending key sequence.

All records in the display (visible, EXCLUDED, NOT SELECTED and SUPPRESSED) are included in a SORT KEY operation.

---

## TOP

### Syntax:

```
>>-- Top -----><
```

### Description:

Display the first page of data.  
Equivalent to the **UP MAX** command.

### Parameters:

None.

### See Also:

**BOTTOM**  
**UP**

---

## UNDO

### Syntax:

```
>>-- UNDO -----><
```

### Description:

Undo one level of changes made to the current file.

Each change level corresponds to an update of a data field within a record.

UNDO is assigned to PF22 by default and may be executed repeatedly to undo multiple levels of changes.



The third number following "Alt=n,n;m" on the status line indicates the number of change levels. If this number is not zero, then changes may be undone using UNDO.

## Parameters:

None.

## See Also:

REDO  
UNDOING - SET/QUERY/EXTRACT  
Undo and Redo Changes

---

## UP

### Syntax:

```
>>- UP -----><
|
|-- Cursor -----|
|
|-- CSR -----|
|
|-- Data -----|
|
|-- Half -----|
|
|-- Max -----|
|
|-- Page -----|
|
|-- n_lines -----|
```

### Description:

Scroll the view of the data within the SDE window upwards towards the top of the file data.

Where no parameter is specified, the scroll amount will be the value specified in the **Scroll>** field in the top right corner of the window display.

Note that the first data record in any **multi record view** will **always** be preceded by its complete group of column header lines.

For **single record views**, the standard headers are always visible at the top of the display area and are not included as part of the scrollable text. Each non-header line within a single record view is identified as being a **field data line**.

UP and DOWN scroll commands will cause the window display area to be adjusted by a number of lines determined by the number of multi record view data records and shadow lines, or single record view field data lines. Other lines in the display area, i.e. **Header lines** and blank filler lines that precede a group of header lines or follow the "End of Data" record, are not included in this calculation.

Attempting to scroll up beyond the first entry (data record, shadow line or field data line) will make the first entry the first line of the display.

### Multi Record View Scrolling

Where the cursor is positioned at an offset within a column header line, the cursor is deemed to be located at the same offset within the data record that immediately follows the group of header lines. Therefore, UP CURSOR will operate successfully when the cursor is positioned within a header line.

The first data record in the display must be preceded by all of its header lines. To accomodate this, the magnitude by which the display is scrolled upwards may be reduced with the result that it may be impossible for the target line of the UP command to occupy the last line of the current display. The display will be scrolled upwards so that the target line is still in view though not the last line in the display. In order to overcome this, the user would have to alter the display area dimensions.

For the same reason, it is possible that execution of an UP command may result in no change to the displayed data, in which case UP PAGE is executed instead.

**Parameters:**CURSOR  
CSR

The data record, shadow line or field data line on which the cursor is positioned becomes the last line of the scrolled display.  
 If the cursor is positioned on a header line, the data record or field data line immediately following the header line becomes the last line in the display area.  
 If the cursor is positioned outside the display area or on the last line within the display area, then UP PAGE is executed instead.

DATA  
 Scroll up to display one page (display window depth) less one line of data.  
 The first data record, shadow line or field data line in the current display area becomes the last line of the scrolled display.

HALF  
 Scroll up half a page of data.  
 The data record, shadow line or field data line that is half way down the page of data in the current display area becomes the last line of the scrolled display.

MAX  
 Scroll up to display the first page of data.  
 The "Top of Data" line becomes the first line of the scrolled display.  
 Equivalent to the **TOP** command.

PAGE  
 Scroll up to display the next whole page of data.  
 The data record, shadow line or field data line before the first line of the current display area becomes the last line of the scrolled display.

*n\_lines*  
 Scroll up a specified number of lines.  
 The data record, shadow line or field data line that is *n\_lines* lines above the current line becomes the first line of the scrolled display.

**Examples:**

up page  
 Scroll the display area up one page.

**See Also:**

DOWN  
 LEFT  
 RIGHT

---

**USE****Syntax:**

```
>>- USE -- record_type -----+----->
                        |
                        +- IN +-----+ struct_name +-
                        |           |
                        +- STRUCTURE -+
```

```
>- WHEN -- filter_expression -----><
```

**Description:**

Where an individual record cannot implicitly be associated with a record type object (**RTO**) by matching on record length, e.g. where several record types exist for the same length record, then a **USE/WHEN** filter is required to match a record to its RTO.

Definition of a **USE/WHEN** filter will update the in-storage RTO identified by *record\_type*. When the Structure Definition Object (**SDO**) containing the RTO is dropped, the user will be prompted to save these changes to the the appropriate Structure Definition File (**SDF**).

## Parameters:

*record\_type*

The record type of any RTO defined within the specified SDO.

IN *struct\_name*

The name of the SDO to which the specified RTO belongs. Defaults to the SDO of the **current SDE window**.

*filter\_expression*

The filter expression consists of a logical combination of simple compare expressions. This logical combination uses parentheses and the AND, OR and NOT logical operators to make a complex compare expression.

The literal logical operator names may be used or their equivalent symbols:

AND	& (ampersand)
OR	! (broken bar),   (vertical line)
NOT	~ (tilde), ¬ (not sign), \ (backslash)

A simple compare expression has the syntax:

```
>>-- columnname --+-----+----- = --+-- value --><
                  |         |         |
                  +-- ~ --+   +--- < ---+
                  |         |         |
                  +-- ¬ --+   +--- > ---+
                  |         |         |
                  +-- \ --+   +--- <= ---+
                  |         |         |
                  + NOT +     +--- >= ---+
                  |         |         |
                  |         |         |
                  +--- << ---+
                  |         |         |
                  +--- >> ---+
```

For numeric columns, comparisons are arithmetic.

For character columns, comparisons are case insensitive with blank padding on the left.

For character columns, the compare value can be delimited with single or double quotes. It only needs quotes if the value contains any of the logical operator characters or blanks, or if it happens to be the same as a column name.

The comparison operators are:

=	Equals.
<	Less than.
>	Greater than.
<=	Less than or equal.
>=	Greater than or equal.
<<	Contains. This operator applies to string columns and returns TRUE if the column contains the value as a substring.
>>	Begins. This operator applies to string columns and returns TRUE if the column starts with the value.

The comparisons can be negated with any of the valid ways of saying NOT.

## Examples:

The following filter expression would match records for which col1 is equal to 'A' and either col2 contains 'C' as a substring or numeric col3 is greater than 4.

```
use Type1Rec when col1 = 'A' and (col2 << 'C' or col3 > 4)
```

**Notes:** The quotes around the literal strings are not needed unless there are columns in the list with name A or C. Blanks separating the elements of the expression are optional.

## VIEW

### Syntax:

```
>>-- View --+-----+----->>
|           |           |
|           *           |
|         +--+ , -++    |
|         |      |      |
|       +---+-----+   |
|       |               |
|       V               |
|     record_type       |
+-----+-----+
```

**Description:**

Select the **record types** associated with data records to be made visible within an SDE BROWSE or EDIT window.

Where a non-generic record type is specified, the **DIRECTYPE** setting is updated to be the first *record\_type* parameter specified on the VIEW command.

If no parameters are specified, then the **default record type** is used.

Where more than one record type is selected in a **multi record view**, the appropriate column headings are displayed at each change of record type.

In a single-record view, horizontal scrolling occurs between visible records only.

VIEW is assigned to **PF4** by default.

### Parameters:

```
record_type
```

The record type of any RTO defined within the current structure definition object (SDO). If *record\_type* is not specified, the default record type is used.

\* (asterisk)

Generic record type indicating that records of all record types within the SDO are to be made visible. If supplied, this must be the only parameter.

### Examples:

```
<view CompUnit,Source,Instruction
<y *
```

## WHERE

### Syntax:

```
>>--- WWhere --- where_clause -----><
```

**Description:**

Display only those records in the display that are of the **default record type** and match the specified *where\_clause* criteria.

Records of the default record type that have been EXCLUDED are included in the WHERE search and are made visible if the *where\_clause* is satisfied. If WHERE is executed with no parameters, then all records that are of the default record type are made visible.

The first record that matches the criteria becomes the current line and all records that do not match the criteria become EXCLUDED. If no records are matched, the Top of Data line becomes the current line.

In **single record view**, the display area will be scrolled to the first data record that matches the *where clause*. Subsequent scrolling left and right through the records will display only those records that have not been excluded by the WHERE command.

The record type used on a WHERE command becomes the new value of the **DRECTYPE** option.

### Parameters:

*where\_clause*

A logical combination of one or more simple filter expressions that define search criteria used to select the required data records. This logical combination uses the following logical operators to make a complex compare expression.

( )	Left and Right Brackets (parentheses).
&	Logical AND symbol (ampersand).
	Logical OR symbol (vertical line).

```

+-----+ & +-----+
|               |
v             +---+ +---+
>> +-----+ filter_expr -----+----->
    |         |                   |         |
    +---+ ( +---+                +---+ ) +---+

```

**Notes:**

1. To avoid confusion, it is recommended that parentheses should be used where more than two filter expressions are specified in order to specify logical AND/OR precedence.
2. Parentheses must be balanced so that there are an equal number of left and right parentheses exist in the where clause.

### ***filter\_expr* Syntax:**

```
>>-- field_col1 -----+-----+-----<op> -----+--- field_col2 -+-----<
      |~|
      | |
      +-+~+-+
      | \ -+
      +- \ -+
```

***filter\_expr* Description:**

Test the contents of *field col1* against a test value specified by *field col2* or *string* to establish a TRUE or FALSE condition.

Where *field\_col1* is a numeric field, the test value must be numeric and an arithmetic compare is performed. Similarly, if *field\_col1* is a character field (data type AN), then test value is treated as a character string with blank padding on the right.

If the data type of the test value is not compatible with *field col1* then an error will be returned.

### ***filter\_expr* Parameters:**

```
field_coll
```

A field column which is defined within the default record type. The contents of the field are to be tested for this *filter expr*.

The field column may be identified by its reference number (e.g. #12) or its name (e.g. BirthDate). A field name will automatically be converted to its field reference.

If the field is a **struct**, then the field is treated as a character string which is a concatenation of all the fields that comprise the structure. If the field is an array, then an error message is returned. However, an individual element of an array may be tested by specifying the occurrence of the element in parentheses as a subscript to the field reference/name. e.g. #13(3) for the 3rd element of a single dimension array. An entry must exist for each dimension of the array. e.g. RoomSize(6,2,4) - a three dimensional array.

The symbols "~" (tilde), "!" (not sign) and "\" (backslash) represent the logical NOT operator and reverses the TRUE or FALSE condition established by the comparison operator <op>.

<op>

Comparison operator specified as one of the following:

=	<b>Equals.</b>
<>	<b>Not Equals.</b>
<	<b>Less than.</b>
>	<b>Greater than.</b>
<=	<b>Less than or equals.</b>
>=	<b>Greater than or equals.</b>
<<	<b>Contains.</b> This operator applies to character columns only and returns TRUE if <i>string</i> is a sub-string of <i>field_col1</i> .
>>	<b>Begins.</b> This operator applies to character columns only and returns TRUE if <i>string</i> is a sub-string at the start of <i>field_col1</i> .

*field\_col2*

A field column which is defined within the default record type. The contents of this field will be tested against the contents of *field\_col1*.

*field\_col2* supports the field specification as *field\_col1*.

For character data the comparison will be case sensitive.

*string*

The string argument to be tested against the contents of *field\_col1*.

The search string may be any of the following:

- ◊ An un-quoted numeric value containing no commas. Supported only when *field\_col1* is a numeric field.
- ◊ A character string enclosed in single (') or double (") quotation marks. Supported only when *field\_col1* is a character field.

The string may contain embedded commas and blanks and the character string will be case-insensitive. Two adjacent quotation mark characters that are embedded in a search string which is enclosed by the same quotation mark characters, will be treated as a single occurrence of the character. e.g.

```
WHERE LASTNAME = 'O'Driscoll'
```

Test the character field LASTNAME for the string "O'DRISCOLL".

- A character string enclosed in single (') or double (") quotation marks with the prefix (or suffix) C. This is equivalent to specifying a quoted search string but that the string search will be case-sensitive. (e.g. C'Cable') Supported only when *field\_col1* is a character field.
- A hexadecimal string enclosed in single (') or double (") quotation marks with the prefix (or suffix) X. Supported only when *field\_col1* is a character field.

## Examples:

```
where LastName = C'Jones'
```

For records that are of the default record type, display only those records for which the field "LastName" is equal to "Jones" in mixed case.

```
where LastName = C'Jones' & Salary > 26000
```

As above but filter the data records further so that the numeric field "Salary" is greater than 26,000.

```
where #12(2,7) \<< 'E77'
```

Display records that are of the default record type and where the two dimensional character array element belonging to field reference number 12 does not contain the string "E77".

```
where PostCode >> "SW1" & ( DispatchDate = OrderDate | ( Quantity > 30 & Cost < 2155.53 ) )
```

Display records that are of the default record type where character field "PostCode" begins with string "SW1" **AND** one of the following conditions are true:

1. The contents of fields "DispatchDate" and "OrderDate" are equal.
2. The numeric field "Quantity" is greater than 30 **AND** the numeric field "Cost" is less than 2155.53.

**See Also:**

FLIP  
LESS  
LOCATE  
MORE

---

**ZOOM****Syntax:**

```
>>-- ZOOM -----><
      |             |
      +-- In  ----+
      |             |
      +-- Out  ----+
```

**Description:**

Switches the display format between **single record view** and multi record view. When switching to single record view the ZOOM command operates on the **default record**.

If no parameter is specified, then the display format toggles between the single and multi record view.

ZOOM is assigned to **PF2** by default.

**Parameters:**

IN	Switch to single record view mode. With FORMAT TABLE in effect this is the equivalent of issuing the command FORMAT SINGLE.
OUT	Switch to multi record view mode. With FORMAT SINGLE in effect this is the equivalent of issuing the command FORMAT TABLE.

# SET/QUERY/EXTRACT

## Syntax:

```
>>-+-----+----- option_name ----- value -----><
    |         |
    +- SET -----+

>>--- Query ----- option_name -----><

          +-----+
          |         |
          v         |
>>--- EXtract ---+--- /option_name ---+-----><
```

## Description:

Structured Data Environment options may set, and their current values queried or extracted into stem-variables for use in REXX macros using the SET, QUERY and EXTRACT commands respectively. Additional operations supported by EXTRACT, allow the user to extract information about individual records and also extract record text.

SDE options may be initialised via the following methods:

1. The (SData) section of the CBLiINI file which is processed at CBLi startup.
2. The SDEPROF macro which is executed at the start of every SDE session.
3. The SDE CLI command SET, executed from the command line or an SDE macro.
4. The "Edit Options" entry of the "Options" drop down menu on the parent MDI window menu bar.

SET options take effect at the following different levels:

Global	The option affects all SDE edited files.
File	The option may be set differently for each SDE edited file.
View	The option may be set differently for each SDE window view of the same file.

## Parameters:

*option\_name*  
The SDE environment option(s). For EXTRACT, multiple options maybe requested at once by separating each with a blank or "/" (forward slash).

*value*  
For SET, the new value to be assigned for *option\_name*.

## Supported Options:

DRECTYPE, FIELD, FOCUS, MSGLINE, MULTIPOINT, POINT, PREFIX, REFERENCE, REGION, SCALE, SHADOW, TYPE, UNDOING, UNNAMED, USING, VALUE, WRAP

## Examples:

```
< sd query shad
< sd shad off all
< 'extract /reference/scale/type/'
```



## DRECTYPE - SET/QUERY/EXTRACT Option

### Syntax:

```
>>+-----+ DREctype --- record_type -----><
    |         |
    +- SET -----+

>>--- Query ----- DREctype -----><

>>--- EXtract --- /DREctype/ -----><
```

### Description:

This option controls the **record type** to be used as the **default record type** if CBLi SDE is unable to determine the default record type from the **focus line**.

The initial value of DRECTYPE is set when the SDE window is opened via an **EDIT** or **BROWSE** commands. In addition to the SET DRECTYPE command, DRECTYPE may be dynamically set following a **VIEW**, **WHERE**, **MORE** or **LESS** command or any command that requires a record type.

See section **Default Record Type** for further information.

SET DRECTYPE takes effect at the View level.

### SET Value:

*record\_type*

The record type of any RTO defined within the current structure definition object (**SDO**).

### QUERY Response:

The record type that is the current setting of the DRECTYPE option.

### EXTRACT Rexx variables:

```
directype.0          1
directype.1          The record type that is the current setting of the DRECTYPE option.
```

## FIELD - EXTRACT Option

### Syntax:

```
>>--- EXtract --- /FIELD/ -----><
```

### Description:

For use in macros, EXTRACT FIELD obtains the characteristics (field reference number, field data type and field name) of each field column header belonging to the **default record type**.

Field data type may be one of the following:

BINTEGER
BIT
CHARACTER
DECIMAL
FIXED
FLOATBIN

FLOATHEX
HEXADECIMAL
INTEGER
STRUCTURE
UNION
VARCHAR
XVARCHAR
ZONED

Values are obtained for each displayed field column in the order in which they appear in the display. Therefore, the **SELECT** command will influence the EXTRACT FIELD values.

### EXTRACT Rexx variables:

`field.0` Number of currently selected fields belonging to the default record type.

`field.i` The blank delimited field reference number (#nn), field data type and field name (including any array element subscripts in parentheses) of the *i*th field in the display.

### See Also:

EXTRACT FOCUS  
EXTRACT VALUE

---

## FOCUS - EXTRACT Option

### Syntax:

```
>>--- EXtract --- /FOCUS/ -----><
```

### Description:

For use in macros, EXTRACT FOCUS obtains information about the **focus field**.

### EXTRACT Rexx variables:

<code>focus.0</code>	7
<code>focus.1</code>	Line number of the first record in the focus line <b>record group</b> .
<code>focus.2</code>	Line number of the last record in the focus line record group.
<code>focus.3</code>	Shadow state of the focus line. (VISIBLE, EXCLUDED, SUPPRESSED or NOTSELECTED)
<code>focus.4</code>	Description of the focus line record group. (DATA, TOF, EOF, SHADOW or BOUNDS)
<code>focus.5</code>	Record type of the focus line record group.
<code>focus.6</code>	Focus field name.
<code>focus.7</code>	Position of cursor within the focus field.

### See Also:

EXTRACT FIELD  
EXTRACT VALUE

## MSGLINE - SET/QUERY/EXTRACT Option

### Syntax:

```
>>+-----+ MSGLine -- ON -- line_num ---+-----+><
    |      |      |      |                  |      |
    +- SET  +-----+      +- lines ---+-----+
                                |      |
                                +--- OVERLAY ---+

>>--- Query ----- MSGLine -----><

>>--- EXtract --- /MSGLine/ -----><
```

### Description:

This option controls the location and number of lines used to display messages in the SDE display window and also display properties of the first message line. The line number is specified as being relative to the top, middle or bottom of the data display area.

SET MSGLINE takes effect at the View level.

### SET Value:

#### *line\_num*

Line number of first message line relative to the top, middle or bottom of the document window.

Line numbers relative to the top of the document window are specified as positive integers. e.g. 2, 8

Line numbers relative to the middle of the document window are specified as offsets from M. e.g. M+1, M-3

Line numbers relative to the bottom of the document window are specified as negative integers. e.g. -6, -10

By default *line\_num* is set to 1.

#### *lines*

The number of lines that message text may occupy within the display area.

If the message text exceeds the number of message lines then an SDE Message List window is opened to display the message text instead.

If *lines* is not specified, the number of message lines last defined in the current CBL view, is unchanged.

By default *lines* is set to 5.

#### OVERLAY

Specifies that the first message line should overlay a line normally used to display a line of data. If omitted, the first message line will be reserved for message display only. Second and subsequent message lines always overlay data lines.

Initially OVERLAY is set on.

### QUERY Response:

The current setting of the MSGLINE option, **ON** or **OFF** followed by location of the first message line, the number of message lines and whether OVERLAY is in effect.

### EXTRACT Rexx variables:

msgline.0	4
msgline.1	The current setting of the MSGLINE option, <b>ON</b> or <b>OFF</b> .
msgline.2	The current position of the first message line.
msgline.3	The number of message lines.
msgline.4	"OVERLAY", if overlay is in effect.

## MULTIPOINT - SET/QUERY/EXTRACT Option

### Syntax:

```
>>+-----+----- MULTIPoint ---+-- ON ---+-----><
    |         |         |         |         |
    +- SET -----+         +- OFF ---+

>>--- Query ----- MULTIPoint -----><

>>--- EXtract --- /MULTIPoint/ -----><
```

### Description:

This option controls whether or not more than one label name may be assigned to a line in the file display via the **SET POINT** command or by overtyping a name in the line's prefix area.

SET MULTIPOINT takes effect at the File level.

### SET Value:

ON Allow multiple label names on a single line.

OFF Do not allow multiple label names on a single line. (Default)

### QUERY Response:

The current setting of the MULTIPOINT option, **ON** or **OFF**.

### EXTRACT Rexx variables:

```
multipoint.0 1
multipoint.1 The current setting of the MULTIPOINT option, ON or OFF.
```

## POINT - SET/QUERY/EXTRACT Option

### Syntax:

```
>>+-----+----- Point --- .name -----><
    |         |         |         |         |
    +- SET -----+         +- OFF -----+

>>--- Query -----+----- Point -----><
                    |         |
                    +--- Point* -----+

>>--- EXtract ---+--- /Point/ -----><
                    |         |
                    +--- /Point*/ -----><
```

### Description:

Assign or unassign a label name to the **focus line** (data record or shadow line) for subsequent use as a line target. The assigned label name is displayed in the line's prefix area.

If **MULTIPOINT** is set on, then a line may be assigned more than one label name.

The same label name may not be assigned to more than one line in the current file. Where a label name is already assigned to a line in the current file, it is unassigned from that line and reassigned to the focus line.

A label name remains assigned to a line even if the line record number changes due to record inserts.

SET POINT takes effect at the File level.

## SET Value:

**.name** Label name of a new label to be assigned to the focus line or an existing label to be unassigned. The preceding "." (dot) is mandatory.

**OFF** Unassign the specified label name from any of the lines in the file display.

## QUERY Response:

**Point** Displays the focus line number and all label names currently allocated to the focus line. If no names are allocated to the focus line (line number n), the following message is returned:

SDE010I POINT No points assigned to line n

**Point\*** Displays the line number and associated label names of all named lines in the current file.

## EXTRACT Rexx variables:

<b>Point</b>	<b>point.0</b>	0 if focus line is not a named line; otherwise, 1.
	<b>point.1</b>	Line number and label name(s) assigned to the focus line.
<b>Point*</b>	<b>point.0</b>	Number of named lines.
	<b>point.i</b>	Line number and associated label name(s) of the ith named line in the current file.

---

## PREFIX - SET/QUERY/EXTRACT Option

### Syntax:

```

>>+-----+-- PREFIX --+-- ON --+-- Left --+-- 6 -----+
|         |         |         |         |         |
+- SET ----+         +- OFF --+-- Right --+-- n_bytes --+

>>--- Query ----- PREFIX -----><

>>--- EXtract --- /PREFIX/ -----><

```

### Description:

PREFIX defines whether the prefix area is displayed, whether it is displayed on the left of the window view or on the right and the number of columns to use.

The prefix displays the record number, and is also a space where line-sensitive **prefix commands** may be entered.

The **PREFIX** option operates at the **view-level**.

## SET Value:

**ON** Display of the PREFIX area is set ON (Default).

**OFF** Display of the PREFIX area is set OFF.

**LEFT** The PREFIX area is set to display on the record's left-hand side (Default).

RIGHT

The PREFIX area is set to display on the record's right-hand side. (i.e. it is, in fact, a suffix)

*n\_bytes*

The length of the prefix area. Min=1, Max=8, Default=6.

### QUERY Response:

The current setting of the PREFIX option, **ON** or **OFF**, followed by **LEFT** or **RIGHT**, followed by the length **n\_bytes**.

### EXTRACT Rexx variables:

prefix.0	3
prefix.1	The current setting of the PREFIX option, <b>ON</b> or <b>OFF</b> .
prefix.2	The current position of the PREFIX option, <b>LEFT</b> or <b>RIGHT</b> .
prefix.3	The current length of the PREFIX area, a number from 1-8.

---

## REFERENCE - SET/QUERY/EXTRACT Option

### Syntax:

```
>>+-----+-- REFerence --+-- ON ---+-----><
    |               |      |      |
    +- SET -----+      +- OFF ---+
>>--- Query ----- REFerence -----><
>>--- EXtract --- /REFerence/ -----><
```

### Description:

This option controls the display of the reference-numbers (**#nn**) occupying a row in the field column-headings for SD edit/browse.

The **REFERENCE** option operates at the **view-level**.

### SET Value:

ON  
Reference number display is set ON.

OFF  
Reference number display is set OFF.

### QUERY Response:

The current setting of the REFERENCE option, **ON** or **OFF**.

### EXTRACT Rexx variables:

reference.0	1
reference.1	The current setting of the REFERENCE option, <b>ON</b> or <b>OFF</b> .

## REGION - QUERY/EXTRACT Option

### Syntax:

```
>>--- Query ----- REGion -----><

>>--- EXtract --- /REGion/ -----><
```

### Description:

QUERY/EXTRACT REGION provides information about the private area region for the current MVS TCB.

**REGION** is not an option for the SET command.

### QUERY Response:

The **address**, **total amount** and the amount of storage **unused** are displayed as hexadecimal numbers, first for storage below and secondly for storage above the 16 megabyte line. e.g.

```
SDE079I REGION Below 16M A=00006000 L=008FA000
U=007E2000; Above 16M A=11600000 L=6EA00000
U=6D6CF000
```

### EXTRACT Rexx variables:

region.0	6
region.1	The below-the-line address.
region.2	The below-the-line total length.
region.3	The below-the-line unused length.
region.4	The above-the-line address.
region.5	The above-the-line total length.
region.6	The above-the-line unused length.

## SCALE - SET/QUERY/EXTRACT Option

### Syntax:

```

                                     +-- - (minus) --+ +-- + (plus) --+
>>+-----+-----+ SCALE --+-- ON ---+-----+-----+-----+><
   |         |         |         |         |         |         |
+- SET -----+         +- OFF --+ +----- c1 -----+ +----- c2 -----+

>>--- Query ----- SCALE -----><

>>--- EXtract --- /SCALE/ -----><
```

### Description:

This option controls the display of the scale (<---,---1), occupying a row in the field column-headings for SD edit/browse.

The **SCALE** option operates at the **view-level**.

### SET Value:

ON  
Scale display is set ON.

OFF  
Scale display is set OFF.

**c1**  
The format character used for intervals of 1 byte.  
Default is the minus-sign (-).  
e.g. "SET SCALE ON . +" results in '< . . . + . . . 1.'

**c2**  
The format character used for intervals of 5 bytes.  
Default is the plus-sign (+).  
e.g. "SET SCALE ON - ," results in '<-----, -----1-'

## QUERY Response:

The current setting of the SCALE option, **ON** or **OFF**, followed by **c1** and **c2**.

## EXTRACT Rexx variables:

scale.0	3
scale.1	The current setting of the SCALE option, <b>ON</b> or <b>OFF</b> .
scale.2	The current setting of the SCALE option, <b>c1</b> .
scale.3	The current setting of the SCALE option, <b>c2</b> .

---

## SHADOW - SET/QUERY/EXTRACT Option

### Syntax:

```

>>+-----+-----+ SHADow  +---+ ON  +---+ +-----+-----+><
    |         |         |         |         |         |         |
    +- SET  +-----+         +---+ OFF +---+ +---+ ALL  +-----+
                                |         |         |         |
                                +---+ EXcluded+-----+
                                |         |         |         |
                                +---+ NOTselected +---+
                                |         |         |         |
                                +---+ SUPressed  +-----+

>>--- Query  ----- SHADow -----><

>>--- EXTract  --- /SHADow/ -----><

```

### Description:

This option controls the display of shadow lines used to indicate the presence of records not displayed due to one of three reasons.

**EXCLUDED** identifies a **record group** that has been excluded by the user using selective line editing techniques e.g. using the **WHERE** or **EXCLUDE** commands.

**NOTSELECTED** identifies a record group that has not been assigned a **record type**. Reasons for this are:

1. The length of the data record does not fit that required to match any of the **RTO** definitions.
2. Fields within the data record do not satisfy the criteria associated with any of the available RTO definitions.  
(RTO criteria are defined as part of the RTO via a **USE WHEN** clause.)
3. Fields within the data record contain invalid data when an RTO definition is applied.

**SUPPRESSED** identifies a record group of a record type that has not been selected for view. (See the **VIEW** command.)

The **SHADOW** option operates at the **view** level.

### SET Value:

**ON**  
Shadow display is set ON for the types that follow.

**OFF**



Shadow display is set OFF for the types that follow.

ALL

The setting affects all types of shadow lines.

EXcluded

The setting affects shadow lines for EXCLUDED records.

NOTselected

The setting affects shadow lines for NOTSELECTED records.

SUPressed

The setting affects shadow lines for SUPPRESSED records.

## QUERY Response:

The current setting of the SHADOW options, **SHADOW EXCLUDED**, **SHADOW NOTSELECTED** and **SHADOW SUPPRESSED**, each followed by **ON** or **OFF**.

## EXTRACT Rexx variables:

shadow.0	6
shadow.1	The literal <b>EXCLUDED</b>
shadow.2	The <b>ON</b> or <b>OFF</b> setting for EXCLUDED shadow lines.
shadow.3	The literal <b>NOTSELECTED</b>
shadow.4	The <b>ON</b> or <b>OFF</b> setting for NOTSELECTED shadow lines.
shadow.5	The literal <b>SUPPRESSED</b>
shadow.6	The <b>ON</b> or <b>OFF</b> setting for SUPPRESSED shadow lines.

---

## TYPE - SET/QUERY/EXTRACT Option

### Syntax:

```
>>+-----+ Type ---+ ON ---+-----><
    |         |         |         |
    +- SET ----+         +- OFF --+
>>--- Query ----- Type -----><
>>--- EXtract --- /Type/ -----><
```

### Description:

This option controls the display of the data-type/position/length (**xx ppp:ll**), occupying a row in the field column-headings for SDE edit/browse.

Possible values for the data-type **xx** are described separately under the description of **Multi-Record Views**.

**ppp** and **ll** indicate the position and length of the field within the unformatted record.

An example would be '**AN 111:30**' indicating an **Alpha-Numeric** character field of **length 30 bytes** starting at **position 111**.

The **TYPE** option operates at the **view-level**.

### SET Value:

ON  
OFF

Set Data-type/position/length display ON or OFF.

**QUERY Response:**

The current setting of the TYPE option, **ON** or **OFF**.

**EXTRACT Rextx variables:**

```
type.0          1
type.1          The current setting of the TYPE option, ON or OFF.
```

---

**UNDOING - SET/QUERY/EXTRACT Option****Syntax:**

```
>>-+-----+-- UNDOING --+- ON --+-+-----+-----+><
                        |   |   |
                        +- OFF -+ +- n_levels --+-----+-----+
                                   |   |
                                   +- n_kbytes --+-----+-----+><

>>--- Query ----- UNDOING -----><

>>--- EXtract --- /UNDOING/ -----><
```

**Description:**

UNDOING defines whether the UNDO (and REDO) facility is enabled for SDE, the number of change levels that SDE will attempt to maintain and the maximum amount of storage SDE can allocate in order to store this information.

The third number following "Alt=" on the status line displays the current number of stored change levels.

The change level count is incremented by any command for which the UNDO operation is supported. i.e. any command that changes the data in the display area or the flag bits of a line. Flag bits include a line's excluded and suppressed indicators.

Multiple changes made to a file as a result of a macro execution are considered to be one change level only.

SDE is informed of any changes to the 3270 terminal when an Attention ID (AID) is generated (e.g. on hitting the Enter key or any of the PF Keys). It is only then that changes to the file are committed to the copy of the file data in SDE storage and the change level is updated. Therefore, where changes have been made to text on multiple lines, SDE has no indication as to the order in which the lines were changed and so assigns a change level to each updated line in ascending order of line number.

The **UNDOING** option operates at the **file-level**.

**SET Value:**

```
ON
OFF
```

Set UNDOING ON or OFF.  
Default is ON.

```
n_levels
```

Number of change levels maintained by SDE for the file. If this value is exceeded, SDE drops the oldest undoable change level.  
Default is 100.

```
n_kbytes
```

Maximum amount of storage (KB) that may be obtained by SDE for storing undo information for the file.  
Default is 64K.

**QUERY Response:**

The current setting of the UNDOING option, **ON** or **OFF** followed by number of change levels and maximum storage allocation in KBytes.

**EXTRACT Rexx variables:**

undoing.0	3
undoing.1	The <b>ON</b> or <b>OFF</b> setting for UNDO.
undoing.2	The number of change levels.
undoing.3	The maximum storage allocation in KBytes.

**UNNAMED - SET/QUERY/EXTRACT Option****Syntax:**

```

>>+-----+ UNNamed --+-- ON ---+-----><
    |         |         |         |
    +- SET -----+         +- OFF --+
>>--- Query ----- UNNamed -----><
>>--- EXtract --- /UNNamed/ -----><

```

**Description:**

This option controls whether unnamed fields appear in the display. Note that COBOL **FILLER** fields are treated as unnamed.

With UNNAMED=ON in effect unnamed fields will appear as a normal column of data in table type display, except that the field-name column heading will appear blank (or FILLER).

The **UNNAMED** option operates at the **view** level, and affects all record types.

**SET Value:**

ON  
OFF

Set display of UNNAMED fields ON or OFF.

**QUERY Response:**

The current setting of the UNNAMED option, **ON** or **OFF**.

**EXTRACT Rexx variables:**

unnamed.0	1
unnamed.1	The current setting of the UNNAMED option, <b>ON</b> or <b>OFF</b> .

**USING - QUERY/EXTRACT Option****Syntax:**

```

>>--- Query ----- USING -----><
>>--- EXtract --- /USING/ -----><

```

**Description:**

QUERY/EXTRACT USING reports the structure name (**SDO**) and **record type** of the **default record** within the **current SDE window** view.

The **USING** option operates at the **view** level.

**QUERY Response:**

Message SDE079I stating the structure name and record type. e.g.

```
SDE079I USING Structure CBL.CBLI.SDO(DIRAMEMP) Record type EMP
```

**EXTRACT Rexx variables:**

using.0	2
using.1	The structure name used in the current SDE window view.
using.2	The default record type.

**VALUE - EXTRACT Option****Syntax:**

```
>>--- EXtract --- /VALUE/ -----><
```

**Description:**

For use in macros, EXTRACT VALUE obtains the values of each field belonging to the **default record type**.

Values are obtained for each displayed field column in the order in which they appear in the display. Therefore, the **SELECT** command will influence the values returned by EXTRACT VALUE.

**EXTRACT Rexx variables:**

value.0	Number of currently selected fields belonging to the default record type.
value.i	The character representation of the <i>i</i> th field value in the display.

**See Also:**

**EXTRACT FIELD**  
**EXTRACT FOCUS**

**WRAP - SET/QUERY/EXTRACT Option****Syntax:**

```
>>+-----+--- WRap ---+--- ON ---+-----><
|         |         |         |         |
+- SET ----+         +--- OFF ---+

>>--- Query ----- WRap -----><

>>--- EXtract --- /WRap/ -----><
```

**Description:**

WRAP defines whether a **LOCATE where clause** search wraps around the end of the range of displayable records to continue searching until either the condition is found or the original focus line is encountered. i.e. when WRAP is set ON, searching forwards through the data continues from the Top of Data after End of Data has been reached. Likewise, searching backwards through the data continues from the End of Data when Top of Data has been reached.

Where WRAP is OFF and the End of Data or Top of Data is reached, then the following message is returned:

```
SDE176W Top/End of file or record range reached.
```

SET WRAP takes effect at the View level.

**SET Value:**

ON Allow LOCATE searches to wrap.

OFF Do not allow LOCATE searches to wrap.

**QUERY Response:**

The current setting of the WRAP options, **ON** or **OFF**.

**EXTRACT Rexx variables:**

wrap.0	1
wrap.1	The current setting of the WRAP options, <b>ON</b> or <b>OFF</b> .

## Prefix Area (Line) Commands

The following commands can be entered in the prefix area of an SDE browse or edit window:

<i>.name</i>	Set a line pointer (line name).
A	Make this line the target for a move or copy (move or copy lines After this line).
B	Make this line the target for a move or copy (move or copy lines Before this line).
C[n] CC	Mark a line or a block of lines for copying.
D[n] DD	Mark a line or a block of lines for deletion.
F[n]	Show the first n of a number of excluded lines.
I[n]	Insert a new line or a block of n new lines. If display format is TABLE, then numeric/bit fields are initialised to zero and character fields to blank, otherwise the whole record is initialised to blank.
L[n]	Show the last n of a number of excluded lines.
M[n] MM	Mark a line or a block of lines for moving.
R[n] RR[n]	Replicate (duplicate) a line or a block of lines n times.
X[n] XX	Mark a line or a block of lines for exclusion from the display.

# Function Keys

3270 Program Function Keys (PFKeys) may be assigned to SDE commands.

The default SDE function keys may be tailored in the **(SDE)** section of the SYSTEM and USER CBLiINI files.

Following startup of CBLi, individual SDE PFKeys may be assigned new definitions using the CBLi **KEYS** command.

The CBLi **KEYS** command may also be used to open the **Function Keys window** to display, and optionally update, the current function key settings.

The SDE program default function keys are:

PF1	insert	Insert a new record following the focus line.
PF2	macro SdZoomW	Open a new window in ZOOMed-mode.
PF3	end	Quit the file (you will be prompted to save any changes).
PF4	view	View only records of the type defined by the cursor position.
PF5	rfind	Locate search string defined by last FIND or CHANGE command.
PF6	rchange	Repeat the change requested by the last CHANGE command.
PF7	up	Scroll the window display upwards by an amount determined by the scroll field or specified on the command line.
PF8	down	Scroll the window display downwards by an amount determined by the scroll field or specified on the command line.
PF9	MDINext	Place focus on the next MDI child window. (Edit view or list.)
PF10	left	Scroll the window display to the left by an amount determined by the scroll field or specified on the command line.
PF11	right	Scroll the window display to the right by an amount determined by the scroll field or specified on the command line.
PF12	retrieve -	Retrieve the last command to the command line.
PF13	delete	Delete the focus line.
PF14	macro SdSel	Create a temporary CMX file containing an SDE SELECT command so allowing the user to choose the field columns to be displayed.
PF15		
PF16		
PF17		
PF18		
PF19		
PF20		
PF21		
PF22		
PF23		
PF24		

Note that, if the following conditions are true, the contents of the command line is concatenated to the definition of the function key and the result executed as a single command.

- The function key command is not set to execute before processing any user screen inputs.  
This is determined by the status of the **BEFORE** field for the PFKey definition in the Function Keys window.
- The concatenation of the PFKey function and the command line contents does not result in an invalid ISPF scroll command (UP, DOWN, LEFT, RIGHT.)  
Where an invalid scroll command is the result, then if the first parameter on the scroll command begins with a numeric character, then an error is returned. Otherwise the contents of the command line are executed before the PFKey function.

# Glossary

The following is a glossary of terms used in this document.

## CLI (Command Line Interface)

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

## CBL

A powerful text editor that runs as an MDI application under CBLi. CBL 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.

## Current Column

The first field column visible within the current display area, belonging to records of the **default record type**. The current column references the same field within all data records that are of the default record type.

## Current Line

The first **record group** (data record or shadow line) displayed within the current display area view.

## Current Record Group

The **record group** occupying the **current line**.

## Current SDE Window

The **SDE Edit View** which received focus last.

The current SDE window prevails even if the current focus window is not an SDE view. The concept of a current SDE window is important when executing CBL/SDE editor REXX macros or when executing SDE commands from a CBL edit view via the **SDATA** command.

## Default Record

The focus line if it contains a visible data record, otherwise the first visible data record following the focus line that is of the **default record type**.

## Default Record Type

The default record type is defined as being the **record type** of the **focus line** if the focus line is a visible or EXCLUDED record group shadow line, otherwise it is the record type defined by **DIRECTYPE** setting.

See section **Default Record Type**.

## Focus Column

The field column on which the cursor is positioned within the focus line.

If the focus line is **not** a visible or EXCLUDED record group, or if the cursor is positioned outside the display area (e.g. the command line) or within the prefix area, the focus column is defined as being the **current column**. The focus column references the same field within all data records that are of the same record type as the focus line.

## Focus Field

The individual field within the **focus line** referenced by the **focus column**.

## Focus Line

The line within the display area on which the cursor is positioned.

If the cursor is positioned on a **header line**, the focus line is the first line that immediately follows the header line. If the cursor is positioned outside the display area (e.g. the command line), then the focus line is defined as being the **current line**.

## Focus Record Group

The **record group** occupying the **focus line**.

## CBLiINI

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.

## Header Line

A line within the SDE window display area that contains column header information for the **record group** that follows. A header line constitutes one line within a group of header lines which scroll with the record data display.

## 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.

**Record Group**

A record group is one or more data records that is represented by a single line in the SDE window display.

A visible data record may be considered to be a record group of one record whereas a shadow line may be a record group of one or more consecutive records of the same record type.

**Record Type**

Term referring to the name assigned to a record type object (RTO) on execution of the **CREATE STRUCTURE** command.

This name is the highest level field name identifier for a record mapping in a COBOL or PL1 copybook or in a CBLi SDE data definition clause.

**RTO (Record Type Object)**

A single record type definition within an **SDO**.

RTO definitions are generated via the **CREATE STRUCTURE** command which uses record structures defined with CBLi's own SDE structure definition syntax or defined from within an existing COBOL or PL1 copybook.

An RTO (and hence the SDO) may subsequently be altered (e.g. via the **USE** command) and saved to an **SDF**.

**SD (Structured Data)**

Structured Data refers to data within file records that have a pre-defined structure. See **structured records**.

SD is also the minimum abbreviation for **SDATA**, the CBLi CLI command used to prefix any of CBLi's **SDE** CLI commands when executed from within a CBLi edit view.

**SDF (Structure Definition File)**

A disk file (sequential data set or PDS/PDSE member) containing a saved Structure Definition Object (**SDO**).

If not already in storage, an SDF gets loaded (creating an SDO) during execution of an **EDIT** or **BROWSE** command in order to apply a structure to records processed by CBLi's SDE.

If an SDO is altered during an SDE edit session and is not flagged as being temporary, then the user will be prompted to save the SDO to an SDF. An SDO may also be saved to an SDF automatically during a **CREATE STRUCTURE** command.

**SDE (Structured Data Environment)**

Structured Data Environment that runs under CBLi and includes MDI display windows, processing options and CLI commands.

SDE enables users to browse, edit, update, copy and compare data in records that are mapped by a CBLi defined structure or COBOL, PL1 copybook.

**SDE Edit View**

An SDE MDI document window that contains a display of structured data. If the same file is displayed in multiple windows, then the user has multiple SDE edit (or CBLi text edit) views of the file.

**SDO (Structure Definition Object)**

An in-storage structure definition consisting of one or more record type objects (**RTO**) that map records in a structured file.

An SDO is created by a **CREATE STRUCTURE** command or an **EDIT** or **BROWSE** command if the **SDF** is not already loaded in storage.

**Structured Data Set**

A data set containing **structured records**.

**Structured Records**

Records that consist of one or more data fields, each with a defined field start position, length and data type.