Compute (Bridgend) Ltd

SELCOPY C++ Version (SLC) New Features

Release 3.10

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

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

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

Contents

SELCOPY C++ Version (SLC) Release 3.10 New Features 1
Documentation Notes 1

Overview 2
Platforms 2
Background 2
Future Schedule 2
Program Name 2
SELCOPY C++ Version Advantages 2
SELCOPY BAL Version Advantages 2
Recent History 3

Important Changes 4
Keyword Changes 4
OPT OPTION OPTIONS reserved words 4
ENVFAIL=CANCEL for Command Line Args 5
POS PARM for Command Line Args 5
POS DSN change for VM CMS only 6
FLAG EOD change for VM CMS only 6
Shared Library for SLCCALL 7
Changed Linkage 7
*< Notation 7

New Features and Other Changes 8
Micro Focus Cobol support for Windows platform 8
ALTX=n for Alternate Index No for MFC files 8
TYPE=C and mixed TYPE arithmetic 8
INCLUDE files - PATH is searched 9
Long file id 9
CMS Notation for a filename 9
Record Format in Summary 10
WORKAREA/WORKA 10
POS ARG for Command Line Args 10
2 blank lines following Command Line Control Statements 10
OPTION PRTSUM=n 11
Continuation Records 12
STACK command to mimick CMS STACK feature 12
DIR for CMS includes SFS path where approp 12
UTIME statement supported for CMS and syntax enhanced 12
PAGEWIDTH for PRINT file may exceed 132 12
DATAWIDTH for PRINT file may exceed 100 13
POS UXREASCD introduced for CMS 13
XV statement supported for UNIX, PC and CMS 13
Ctrl-C handled to give quiet termination 14
Abnormal termination handled 14
Command line option -NOTRAP introduced 14
F=CARD has DSN reported in Summary 14
SLEEP statement supported for CMS 14
FLUSH statement support for TAPE output 15
NEWBLK param to give equivalent of FLUSH 15
CSV data via COMPRESS/EXPAND statements 15
LEFT,RIGHT,CENTRE for adjusting a field 16
SUMPRT is synonym for PRTSUM 16
OPTION CALLTYPE=DIRECT|VIA_SLCCALL 17
OPTION LIBNAME="libnam" 17
CVDATE statement for converting date formats 18
CVPP statement for changing size of a p.d. field 19
POS SCALE introduced 20
FLAG EOMBR/EODIR/EODSK supported 20
Floating Point Conversion Char <-> Float (Native) 21
Floating Point Conversion BIN <-> HEX (Base 2<->16) 21
Floating Point Default Style 22
Floating Point Arithmetic 22
Floating Point Literals 23
Floating Point Conversion Char (Exponent) --> Float 23
Floating Point Conversion - Rogue Lengths 23
GEN statement supports variable RANGE @LOW,@HIGH 24
CVCH supports FORMAT'd conversion to Printable Hex 24
CVCC|CVCZ|CVZC conversion statements supported 24
RECFM=V3 supported for FTP Block Mode files 25
CVxx statements tolerate conflicting TYPE codes 25
FORMAT string for CVxx may be dynamic 26
SQL DataBase Access with ODBC 26
SSN = SubSystemName 26
USER = the_id_of_the_authorized_user 27
ODBCPASS (OPASS) = the_user_password 27
OPTION CBLSQLOG [= fileid] 27

Contents

New Features and Other Changes
HEADER parameter for ODBC tables....28
POS FHDR - Special Position....28
Column Names as fields....29
READ - Type 1....29
READ - Type 2....30
READ - Type 3....30
ODBC Operation....30
UPDATE of current row....30
DELETE of current row....30
Prepared INSERT....31
DIRTYPE for DIR and DIRDATA input....32
RAW parameter for DIR and DIRDATA input....33
Pointer and Keyword Arithmetic....33
Error messages for IF-type statements....34
DCL statement....34
CASEI for Case Insensitive Compare....36
POS DATE information extended....36
Numeric literals allowed in quotes with punctuation....37
SEP char in comment respected even if in quotes....37
Multiple fields on a PRINT statement....37
Multiple fields on a PRINT statement with hex FMAT....38
HEX offsets supported....38
MVS DD SELCNAM and SELCMSG supported....39
MVS Option to specify alternate CBLNAME....39
MVS Operator Message switches in CBLNAME obeyed....39
MVS SELCOPY/i LIST Window as an input file....40
MVS: UPD,DEL,INS totals in summary....42
MVS: Use SLC's SYSTEM command to issue a TSO command....43
MVS: Treat %ABC% as a rexx variable....43
MVS: DIR input off HFS....43
OPTION TRAP introduced as synonym for OPTION ABTRAP....43
DCL var assignment from @ptr....43
WNT: File Sharing....44
NOT*FOUND*OR*EMPTY message changed....45
MVS: CALL statement for Assembler routines....45
Synchronize release number for Product Suite....45
The CHOP statement....46
Testing a field for numeric....46
Use of negation on an IF statement....47
Windows - File Redirection with Single Quotes....47
PRINT TYPE=S corrections to match original version....47
CMS fix for CLOSE statement....48
OPTION ERRLIM=nn....48
&DCLvar usage to refer to the address of DCLvar....48
DCL var assignment from Compound Source....49
MVS System Symbols supported....49
OPTION PRINTABLE=hexstr UNPRINTABLE=hexstr....50
COMPRESS and EXPAND using DCL vars and no WORKLEN....50
TRAN statement supports HITS parameter....51
DO statement with Parameters for Sub-Rtn....51
DSNPFX=NO for MVS....52
MVS: Oversized SELCOPY.MSG file now tolerated....53
MVS: Trailing blanks on SYSPRINT eliminated....53
Pagination error fixed....53
CMS: Reading off an attached VSE disk....53
CMS: RECFM=U on CMS minidisks....54
MVS: Abended with 2 LIST input files....54
CMS: Recursion Loop fixed....55
Other Fixes....55

Messages....56
ERROR Messages - Control Statement Analysis....56
ERROR Messages - Selection Time....60
WARNING and Information Messages in Summary....62

SELCOPY C++ Version (SLC) Release 3.10 New Features

Documentation Notes

Information in this new feature document applies to the multi-platform C++ version of the mainframe SELCOPY Batch product. The C++ version will be known as SLC and this document details changes introduced to SELCOPY (SLC) 3.10 Build 004 since SELCOPY (SLC) 2.08 Build 387.

The **SELCOPY C++ version** for z/OS (MVS) and z/VM (CMS) operating systems is provided as an executable load module, SLC, which is included as part of the **SELCOPY Product Suite**, available for download and install from the CBL web site **SELCOPY Downloads** page (URL: http://www.cbl.com/selcdl.html). Corrective service is provided in the form of new build levels via z/OS SMP/E SYSMODs or a VM/CMS VMARC software update bundle.

The **SELCOPY C++ version** for IBM i, UNIX and Windows operating systems is provided as downloadable .TGZ and .ZIP archive files from the CBL web site **SELCOPY Downloads** page.

The following publication documents operation of the SELCOPY program for both the mainframe BAL (Basic Assembler Language) version and the C++ version. It is available in Adobe Acrobat PDF format from the CBL web site **SELCOPY Documentation** page:

- SELCOPY 2.0x User Manual

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

The following generic terms are used throughout this document:

MVS	-	IBM z/OS, OS/390, MVS/ESA, MVS/XA, MVS/SP, OS.
CMS	-	IBM z/VM, VM/ESA, VM/XA, VM/SP.
AIX	-	IBM AIX
DEC	-	HP Tru64
HPX	-	HP HP-UX
LNX	-	Linux x86 (RHEL or SLES)
LNZ	-	z/Linux (RHEL or SLES)
SCO	-	UnXis (SCO) UnixWare
SUN	-	SUN Sparc Solaris
WNT	-	MicroSoft Windows x86 (NT, Vista, 7, Server 2008)
AS/400	-	IBM i, i5/OS, OS/400
UNIX	-	AIX, DEC, HPX, LNX, LNZ, SCO and SUN.
PC	-	x86 servers or workstations running MicroSoft MS-DOS or Windows.
ALL	-	AIX, DEC, HPX, LNX, LNZ, SCO, SUN, MVS, CMS, AS/400 and WNT.

Overview

Overview on the evolution of the C++ version of the SELCOPY Batch utility product.

Platforms

This document applies to the following platforms:

- UNIX. (For supported platforms please see Documentation Notes above.
- MicroSoft Windows.
- IBM Mainframe z/OS (MVS) and z/VM (CMS).

Background

SELCOPY for the IBM mainframe, first released in 1971, is written in BAL (Basic Assembler Language) and is ongoing.

SELCOPY for the various UNIX and PC platforms and the, first released in 1996, is written in C++ for the benefit of portability between platforms. It has essentially the same syntax as the BAL version.

Certain features in the C++ version now more than justify its release to mainframe users, giving them many benefits, in particular, the ability to read lists from the SELCOPY Interactive environment.

Future Schedule

The BAL version for the mainframe will ultimately be phased out, but not until the C++ version has been adapted to call BAL subroutines for all the critical, cpu-intensive, parts of the product. Thus the raw power of the BAL version will be retained, with the C++ overhead apparant only in the control card analysis at start up.

Program Name

The IBM Mainframe C++ version will be known as **SLC**.
The IBM Mainframe BAL version will remain as **SELCOPY**.
The IBM i, UNIX and Windows C++ versions may remain as SELCOPY, or be renamed to SLC, depending on the installation's preference.

SELCOPY C++ Version Advantages

- Reads lists as provided by SELCOPY/i. e.g. Vols, DataSets, Members, Queues, Allocs.
- Command line invocation can provide all control statements on the PARM field.
- Case insensitive compare.
- Reverse scan.
- CSV (Comma Separated Variables) support.
- TYPE=C and mixed TYPE arithmetic.
- INCLUDE statement.
- CVDATE statement for converting date formats.
- HEX offsets supported.
- DECLARE variables
- Multiple fields on a PRINT statement.

SELCOPY BAL Version Advantages

- Raw speed.
- DB2, IMS, ADABAS support.

Recent History

This document incorporates all new features up to SELCOPY C++ Version 3.10 Build Level 004 that have been introduced since SELCOPY C++ Version 2.08 Build Level 387 and have not yet been included in the SELCOPY manual.

For ease of reference, all new features declared in the SELCOPY C++ Version 3.00 New Features document (published as "SNF300c.txt"), have been repeated in this (3.10) document.

"SNF300c.txt" may therefore be disregarded, as it only covers new features up to SELCOPY C++ Version 3.00 Build Level 007. "SNF300c.txt" was initially published on CBL's FTP server ftp://ftp.cbl.com/All on 2011/06/24.

Platform	SELCOPY (SLC) Release	Build Level	Latest Change	Publish Date	Comments
MVS Mainframe	3.00	001	2010/12/01 23:16	2010/12/16	GA. The SLC program.
WNT Windows	3.00	002	2010/12/22 22:05		Internal only.
WNT Windows	3.00	003	2011/02/01 16:36		Internal only.
WNT Windows	3.00	004	2011/03/01 22:32	2011/03/01	GA. (ftp.cbl.com/wnt)
CMS Mainframe	3.00	005	2011/03/06 19:15		Restricted Dist.
WNT Windows	3.00	006	2011/05/25 22:28	2011/06/08	GA. (ftp.cbl.com)
LNX Linux	3.00	007	2011/06/17 11:38	2011/06/22	GA. (ftp.cbl.com/lnx)
ALL SNF300c	3.00	007	2011/06/22 15.41	2011/06/24	GA. (ftp.cbl.com/all) Documentation for all supported platforms.
CMS Mainframe	3.00	008	2011/10/30 16:10		Restricted Dist.
WNT Windows	3.00	008	2012/01/10 22.48		Internal only.
MVS Mainframe	3.00	009	2012/03/04 21:28	2012/03/15	GA.
WNT Windows	3.00	009	2012/03/05 18:48		Internal only.
WNT Windows	3.10	001	2012/04/01 16:48		Internal only.
MVS Mainframe	3.10	001	2012/04/10 17:51	2012/04/12	GA. (sysmod)
WNT Windows	3.10	002	2012/05/29 12:09		Internal only.
CMS Mainframe	3.10	002	2012/05/29 12:10		Restricted Dist.
MVS Mainframe	3.10	003	2012/06/29 16:22		Internal only.
WNT Windows	3.10	003	2012/06/30 17:44		Internal only.
MVS Mainframe	3.10	004	2012/09/21 17:10	2012/09/24	GA. (sysmod)
CMS Mainframe	3.10	004	2012/09/21 17:10	2012/09/24	GA.
WNT Windows	3.10	004	2012/09/21 17:10	2012/09/24	GA.

Important Changes

There are 8 important changes which can give different results from the previous releases of SELCOPY C++ on UNIX, Windows and CMS systems:

- Keyword Changes
- OPT OPTION OPTIONS reserved words
- ENVFAIL=CANCEL for Command Line Args
- POS PARM for Command Line Args
- POS DSN for VM/CMS
- Shared Library for SLCCALL
- FLAG EOD change for VM/CMS only
- *< Notation

2009/01/23 s208_917

Keyword Changes

The keywords UXRETCD, UXRETSYS, UXRETVSAM and UXREASCD have been introduced, each refering to SELCOPY's internal storage where a copy of the appropriate value is held as a 4-byte binary field, in Big Endian format regardless of platform.

The keywords RETCD, RETSYS, RETVSAM and REASCD and their documented synonyms are still supported, but are now always treated as values (numbers) irrespective of context.

For example, previous releases of SELCOPY treated the keyword RETCD in two different ways depending on the type of statement using it. e.g.

```
                                * --- Old meaning --------------------------
RETCD = 22                      * Sets the Return Code value to 22.
if RETCD > 8                    * Checks the Return Code value.
then print from RETCD           * Printed from POS UXRETCD.
then print from UXRETCD         * ERROR 007 due to POS UXRETCD not supported.
cvbc 4 at RETCD  to 101 fmt=z99 * Used 4 bytes at POS UXRETCD.
```

This difference has now been removed and all references to the keyword, RETCD, or to its synonyms, RC and RETCODE, are treated as numbers which, in keeping with other SELCOPY syntax, may be used as an absolute value or as a position within the current input record area or within the workarea if WORKLEN coded. e.g. (New meaning.)

```
                                * --- New meaning --------------------------------
RETCD = 22                      * Same as before. Sets the Ret Code value to 22.
if RETCD > 8                    * Same as before. Checks the Return Code value.
then print from RETCD           * Changed.  Prints from POS 9 assuming SELCOPY's
                                *           current Return Code setting is 9.
then print from UXRETCD         * New.      Prints from UXRETCD as defined above.
cvbc 4 at RETCD  to 101 fmt=z99 * Changed.  Uses POS 9 in wkarea if curr retcd = 9.
cvbc 4 at UXRETCD to 101 fmt=z99 * New.     Uses SELCOPY's internal storage where
                        * the Big Endian version of the current Return Code is held.
```

This rationalization then allows support for the use of RETCD as an operand on arithmetic statements. See "Pointer and Keyword Arithmetic " below.

Note:

1. User modification of the 4 bytes at POS UXRETCD will no longer affect the Return Code used for subsequent references to RETCD, or for the value displayed at the end of SELCOPY's summary. SELCOPY's own internal value is always used.

In order to modify SELCOPY's internal Return Code, the keyword RETCD must be used as the destination field of a statement. e.g.

```
RETCD = 22                      * Sets the Return Code value to 22.
mult 11  by 2  into RETCD       * Sets the Return Code value to 22.
```

Support for the POS keywords, PGNO and its synonym PAGECOUNT, has been withdrawn. Please use POS UXPGNO instead in order to refer to SELCOPY's internal storage holding the current page number for the PRINT file as a 4-byte Packed Decimal field. e.g.

```
cvpc 4 at PGNO   to 3 at 50     * Will give:  ERROR 007  INVALID NUMERIC PARAM
cvpc 4 at UXPGNO to 3 at 50     * Will work ok.
```

2005/03/16 s208_438

OPT OPTION OPTIONS reserved words

The control words OPT, OPTION and OPTIONS have been added to the list of reserved words and may not be used as labels.

See below: `ERROR 182 RESERVED WORD MAY NOT BE USED AS A LABEL`

2005/07/20 s208_614

ENVFAIL=CANCEL for Command Line Args

OPT ENVFAIL=CANCEL

Reference to a command line argument that has not been provided on the command line invoking SELCOPY will no longer cause ERROR 152 unless OPTION ENVFAIL=CANCEL is set.

The default ENVFAIL setting is ENVFAIL=SAME, thus it is possible that a SELCOPY invokation that would have cancelled on previous releases, will now run without an error message.

OPT NOENVVAR

Use of the OPTION NOENVVAR statement will now also apply to any command line argument keywords, %1, %2 etc, which may be used in the Control Statements, regardless of whether any command line arguments have been provided on the command line invoking SELCOPY.

2005/07/28 s208_614

POS PARM for Command Line Args

On earlier releases of selcopy, POS PARM referred to system storage where each argument is null terminated.

POS PARM, synonym PARMS, for AS/400, UNIX and PC, refers to a copy of the string of parameters coded by the user on the command line invoking SELCOPY.

Minimum length allocated for the PARM string is 80 bytes, padded with blanks, and terminated by X'00' at POS PARM+81.

If the PARM data exceeds 80 bytes, no blank padding is done, but the string is still terminated by X'00', immediately following the last token. Thus, the length of the PARM data may be ascertained by scanning for the X'00'.

Where possible, POS PARM data will now be identical to the command line arguments as supplied by the user, with white space intact, and without each argument being terminated by X'00'.

Note:

1. Command line options such as -ctl=fname -lst=fname and -log=fname for I/O redirection are excluded, together with any white space following them.

   Also excluded are any SELCOPY Control statements that may follow the PARM data.

See also POS ARG, which refers to the full arg string, is described below.

The example below illustrates both POS PARM and POS ARG usage, as well as the ability to reference a PARM field using %1, %2 .. %9 syntax, even though the %n is within quotes.

Also illustrated is the ability to issue a SYSTEM command, synonym SYS, to invoke another instance of SELCOPY to run a separate job, writing its listing file to a different fileid.

The 2nd listing file is not shown below as it is so trivial. However, the -ctl=fileid option could have been used to invoke a file of non-trivial control statements instead of just printing and logging to the terminal the literal "slc2 output.".

The w=2222 option is required in order to allow setting the current LRECL to a value greater than 80.

The spurious comments, "* -lst=x.x" and "* xxx -lst=zzz" were included simply to show that valid redirection syntax is not obeyed if part of a comment.

The "-lst=selc.lst2" is not obeyed for the parent SELCOPY because it is within a quoted string.

```
selcopy  -lst=selc.lst1 xxParm1  'xxParm2' !opt w=2222 dw=60 \
!pr 'Param 1 is "%1" and Param 2 is "%2"' * Can be referenced with %n syntax. \
!if p parm,parm+400 = x'00' !t l=@-parm !t pr fr parm \
!if  p arg, arg+400 = x'00' !t l=@-arg !t pr fr arg \
!sys 'selcopy -lst=selc.lst2 !plog "slc2 output." * -lst=x.x !e'  \
!e * xxx -lst=zzz
```

This will produce the following report on the "selc.lst1" file:

```
SELCOPY/WNT 2.08 at Compute Bridgend - Wales - (pw=94)  -djh-      2010/05/11 21:24   PAGE   1
------------------------------------------------------------      ----------------   --------

         EQU %1   xxParm1
         EQU %2   'xxParm2'


         opt w=2222 dw=60

     1.  pr 'Param 1 is "%1" and Param 2 is "%2"' * Can be referenced with %n syntax.

         if p parm,parm+400 = x'00'
     2.  t l=@-parm
     3.  t pr fr parm

         if  p arg, arg+400 = x'00'
     4.  t l=@-arg
     5.  t pr fr arg

     6.  sys 'slc208  -lst=selc.lst2 !plog "slc2 output." * -lst=x.x !e'

         e * xxx -lst=zzz


    INPUT   SEL SEL                                                                 RECORD
    RECNO   TOT ID.          1         2         3         4         5         6  LENGTH
    -----   --- --- ....,....0...,....0...,....0...,....0...,....0...,....0  ------
        0     1   1 Param 1 is "XXPARM1" and Param 2 is "xxParm2"                    80
        0     1   3 xxParm1  'xxParm2'                                              80

        0     1   5 -lst=selc.lst1 xxParm1  'xxParm2' !opt w=2222 dw=60 !pr 'Par    319
                    am 1 is "%1" and Param 2 is "%2"' * Can be referenced with %
                    n syntax. !if p parm,parm+400 = x'00' !t l=@-parm !t pr fr p
                    arm !if  p arg, arg+400 = x'00' !t l=@-arg !t pr fr arg !sys
                     'selcopy -lst=selc.lst2 !plog "slc2 output." * -lst=x.x !e'
                      !e * xxx -lst=zzz
                    ....,....1...,....2...,....3...,....4...,....5...,....6

SUMMARY..
 SEL-ID      SELTOT      FILE     BLKSIZE  LRECL           FSIZE   CI    DSN
 ------      ------      ----     -------  -----           -----   --    ---
    1----6        1


    ** SELCOPY/WNT 2.08.922  Licensed by Compute (Bridgend) Ltd  +44 (1656) 652222 & 656466 **
                                    ** Expiry: 01 Feb 2011 **
```

2006/01/16 s208_859

POS DSN change for VM CMS only

POS DSN for SELCOPY 2.08 running under VM/CMS, will refer to a formatted copy of the Data Set Name, where blanks are removed and '.' is used as a separator. e.g.

```
  'ABCDNAME.EXECTYPE.A1'
  'XYZ.LST.G2          '
  'XYZ.LST.G2          '
   ....,....1....,....2
```

whereas previous releases gave:

```
  'ABCDNAMEEXECTYPEA1'
  'XYZ     LST     G2'
   ....,....1....,...
```

2007/09/22 s208_906

FLAG EOD change for VM CMS only

FLAG EOD is no longer treated as FLAG EODISK, but as FLAG EODIR.

For UNIX and PC platforms, the FLAG statement previously never supported FLAG EOD.

For the mainframe, only the CMS platform supported FLAG EOD.

Please refer to the "FLAG EOMBR/EODIR/EODSK supported" heading below for a detailed description.

2006/06/06 s208_883

Shared Library for SLCCALL

The name of the Shared Library searched for CALL statements has been changed from "slccall.xxx" to "libselc.xxx", where "xxx" varies according to the platform on which SELCOPY is run. e.g.

```
"LIBSELC.DLL"    * For Windows platforms.
"libselc.so"     * For most UNIX platforms, including AIX 4.2 and above.
"libselc.sl"     * For HP-UX.
"libselc.a"      * For AIX 4.15.
```

It is no longer mandatory to have the shared library on your system, because it is only loaded when your SELCOPY Control Statements include the use of the CALL statement, for calling your user written sub-routine(s).

If you do not use the CALL statement, you no longer need the libselc.xxx shared library on your system at all.

If the CALL statement is used, the libselc.xxx shared library no longer needs to be located in the fixed location /usr/selcopy for UNIX, but may be placed in any directory mentioned in the PATH for Windows, or in LD_LIBRARY_PATH for UNIX (LIBPATH for AIX) or in a system directory such as /lib or /usr/lib.

It is the user's responsibility to create the shared library containing the CALL'ed routines by compiling and linking the appropriate C or C++ source code. The slccall.c and slccall.h files are supplied with the SELCOPY product as an example.

Changed Linkage

SELCOPY will no longer call the interface routine, slccall, but instead will call the named routine directly, thereby reducing overhead.

The slccall function provided on earlier code samples is now effectively obsolete, but is still included in the current code sample to enable continued use of the legacy technique.

A single argument is passed by SELCOPY to the called routine.

The single argument is a pointer to the first pointer in an array of pointers to char.

The array of pointers is always terminated with a null pointer, allowing the called routine to recognize a variable number of array elements.

Thus, every called routine must be defined as:

```
int  routine_name (char **parms);
```

and casts or manipulation should be used to interpret data that is not char.

Please refer to slccall.c for more info.

See also OPTION LIBNAME and OPTION CALLTYPE, described below, for forcing use of the legacy technique.

2007/06/11 s208_903

*< Notation

The check for "*<" notation, defining comment data for the rest of the line, ignoring any further separator chars, will be obeyed if the "*<" is found in pos 1,2 of the **logical** line, rather than being restricted to pos 1,2 of the **physical** line. e.g.

```
....,....1....,....2....,....3....,....4....,....5....,....6....,.
*<    print 'aaa'  !pr 'bbb'   !pr 'ccc'   * Whole line is comment.
   *< print 'xxx'  !pr 'yyy'   !pr 'zzz'   * Whole line is now comment, but
   * on prev releases, yyy and zzz still got printed.
```

An added benefit is that it is now possible to keep the 1st statement on a line active, but comment out all subsequent statements on the same line using "*<" at the logical start of the 2nd statement on the line. e.g.

```
print 'aaa' * Comment. !  *< pr 'bbb' !pr 'ccc'   * Only aaa gets printed.
```

Note:

1. Changing the "`* Comment.`" to "`*< Comment.`" has no effect, because the "*<" is not in logical pos 1,2 of the statement, so it simply gets treated as a normal comment.

   However, changing the "`* Comment.`" to "`! *< Comment.`" would have an effect.

See also section below, "SEP char in comment respected even if in quotes".

New Features and Other Changes

Features that enhance or add new operation to the SELCOPY C++ Version without affecting existing job streams.

2006/03/24 s208_880

Micro Focus Cobol support for Windows platform.

While VSAM-type file organizations are supported natively on IBM mainframe platforms, on other platforms, VSAM-type file organizations are currently only supported when the host machine has Micro Focus Cobol (MFC) proprietary software installed.

Customers with Micro Focus Cobol software on a Windows platform may use the full range of VSAM I/O statements as documented in the SELCOPY User Manual for VSAM on the IBM Mainframe.

Note:

1. For SELCOPY to process Micro Focus Cobol files, it is necessary that the user has a valid current licence for "Net Express", or equivalent product from Micro Focus, on the machine being used.

   Micro Focus Cobol software is available from:

   ```
   Micro Focus International Limited,
   9420 Key West Avenue,
   Rockville,
   Maryland 20850.

   http://www.microfocus.com/
   ```

OPTION MFC

To indicate to SELCOPY that VSAM keywords on its control statements are to be interpreted as meaning use Micro Focus Cobol processing, the MFC option must be supplied to SELCOPY on an OPTION statement, either within the current control statements, or in the selcopy.nam file.

OPTION MFC will also cause RECFM=MFV files to be treated as real MFC files.

OPTION MFC is ignored for IBM mainframe platforms, all of which support IBM's original VSAM.

It is recommended that OPTION MFC is placed in the selcopy.nam file, to give system wide effect. e.g. At the DOS command prompt:

```
set  MFCFIL=c:\mfc\mfcmast.fil
selcopy   -ctl=mfcdemo.ctl
```

Where the file "mfcdemo.ctl" contains:

```
option mfc           * Treat VSAM keywords as MFC equivalents.
read %MFCFIL% vsam   * Gets fileid from system envvar, MFCFIL.
print  stopaft=22
```

2008/07/03 s208_913

ALTX=n for Alternate Index No for MFC files

To read a Micro Focus Cobol KSDS using an alternate index, the ALTX parameter is used. KEYIX and AIX are synonyms.

The ALTX agument must be numeric, indicating which alternate index to use. ALTX=0 is the default, meaning use the Prime Key. e.g.

```
read c:\mfc\mfcmast.fil KSDS ALTX=2    * Use the 2nd Alternate Index.
```

2005/11/24 s208_836

TYPE=C and mixed TYPE arithmetic.

Support for TYPE=C character arithmetic is introduced.

Arithmetic data types may be mixed at will on arithmetic statements by coding a TYPE parameter following each field concerned.

If the TYPE parameter is not coded on all fields, the data type first mentioned will become the default type for all unqualified fields on that statement.

If no TYPE parameter is coded, the default data type for all fields is TYPE=P (Packed Decimal), as has always been the case. e.g.

```
** c:\djh\cc\slc\ctl\ssif25 **#    L=001 --- 2005/11/24 23:32:48  (L07)
             * IF statement with mixed string types - 2005/11/24
opt noban dw 80  w 222
  in card   fill=' '  !lrecl 80
  cvcp 10 at 01   to 4 at 21   * to Packed Decimal.
  cvcb 10 at 11   to 3 at 31   * to Binary.

  if 4 at 21 ty p    =   3 at 31 ty b    !t p 41 = 'a=b'
  if 4 at 21 ty p    >   3 at 31 ty b    !t p 41 = 'a>b'
  if 4 at 21 ty p    <   3 at 31 ty b    !t p 41 = 'a<b'

  p 61 = xxxx
  add 3 at 31 ty b    to 4 at 21 ty p   into 4 at 51 ty c
  add 4 at 01 ty c    to 4 at 11 ty c   into 4 at 61 ty z

  cvch  4 at 21   to 21        * Convert to readable hex.
  cvch  3 at 31   to 31
* cvch  4 at 51   to 51
  pr
  e   **
123       456
-6        +6
1234      1234
1         0
-0        +0
0         -1
/*
```

2004/08/06 s208_388

INCLUDE files - PATH is searched

If no path information is given for an INC file, and the file is not found on the current directory, then all directories defined in the PATH environment variable are searched for the required file. e.g.

```
  inc   xyz.equates      * Will be found if on curr dir or on any dir on the PATH.
  inc ./xyz.equates      * Will be found only if it is on the current directory.
```

Also applies to the command line argument "-ctl" defining the Control Card input file, but not to the "-lst" argument which is an output file. e.g.

```
  selcopy   -ctl=xyz.ctl    -lst=xyz.listing    arg1 arg2 ...
                        * The file "xyz.ctl" will be found if on curr dir or on any dir on the PATH.
                        * The file "xyz.listing" will be written to the curr dir.
```

2004/12/10 s208_419

Long file id

The DSN reported in the summary is now chopped to fit on the summary within the 132 byte line, taking as many additional lines as is required. e.g.

```
SUMMARY..
 SEL-ID      SELTOT      FILE     BLKSIZE  LRECL           FSIZE   CI    DSN
 ------      ------      ----     -------  -----           -----   --    ---
    1             1 READ stuvwxyz   2048   167 U               1       C:\tmp\160_byte_filename_for_G
                                                                       xxxx_Gxxxx_of_DDDDD_2004-12-08
                                                                       _sq11471_padlen7_Padding_Lengt
                                                                       h-29-up-to-here-_ABCDEFGHIJKLM
                                                                       NOPQRSTUVWXYZ---60---abcdefghi
                                                                       jklmnopqrstuvwxyz
```

2004/12/14 s208_430

CMS Notation for a filename

For UNIX, all filenames are used as is, with no conversion, but for WNT, PCD and OS2, earlier versions of SELCOPY recognised and unconditionally converted a CMS notation filename to the local platform notation. This made it impossible to reference a filename with 2 or 3 tokens separated by blanks or dots, if it happened to be also valid as a CMS name. e.g.

```
  "  ABCD  XYX  C "   became   "C:ABCD.XYZ" on a Windows machine.
  "ABCD.XYX.C"   also became   "C:ABCD.XYZ".
```

Recognition of CMS Notation for a filename will in future only be the default for CMS and PCD platforms.

Where CMS Notation is required for platforms other than CMS and PCD, the CMS keyword can now be used on the READ statement for the file. e.g.

```
READ   " ABC  DEF  G "   CMS     * Force CMS Notation, if possible.
```

The keyword BFS (Byte File System) may be used on the CMS platform to override the default action of checking for and converting a CMS notation filename. Thus all files on the BFS are also readable. e.g.

```
READ   " ABC  DEF  G "   BFS     * Force BFS Notation, keeping fname intact.
```

2004/12/14 s208_430

Record Format in Summary.

The record format (RECFM) of a file is no longer reported as just F, V or U. If the data is blocked, then FB or VB is reported. RECFM=U (Undefined) cannot be blocked. This change is made in order to match SELCOPY on the mainframe.

2005/03/20 s208_441

WORKAREA/WORKA.

The keywords WORKAREA and WORKA have been introduced as synonymns of WORKLEN. e.g.

```
OPTION          WORKAREA=22000
READ   ABCFILE  WORKAREA=22000
```

2005/07/20 s208_614

POS ARG for Command Line Args

POS ARG, synonym ARGS, for AS/400, UNIX and PC, refers to a copy of the string of all arguments coded by the user on the command line invoking SELCOPY.

The POS ARG string will include:

1. All arguments for options such as -ctl=fname -lst=fname and -log=fname for I/O redirection.
2. All POS PARM data.
3. All SELCOPY Control statements that may follow the PARM data, including the separator characters (!).

Minimum length allocated for the ARG string is 80 bytes, padded with blanks, and terminated by X'00' at POS ARG+81.

If the ARG string exceeds 80 bytes, no blank padding is done, but the string is still terminated by X'00', immediately following the last token. Thus, the length of the ARG data may be ascertained by scanning for the X'00'.

Where possible, POS ARG data will now be identical to the full command line arguments as supplied by the user, with white space intact, and without each argument being terminated with X'00'. e.g.

```
selcopy  -ctl user.statements  -lst=listing.file  abcdef  ghijkl  !pr from args !e
```

Should print:

```
....,....1....,....2....,....3....,....4....,....5....,....6....,....7....,....8
-ctl user.statements  -lst=listing.file  abcdef  ghijkl  !pr from args !e
```

See also POS PARM, which refers to the parameters only, is described above.

A complete example of POS ARG and POS PARM usage is also given in the description of POS PARM.

2005/08/01 s208_626

2 blank lines following Command Line Control Statements

When Control Statements are passed to SELCOPY on the command line, with additional Control Statements taken from a redirected input file, then 2 blank lines are printed following the command line control statements, before printing the control statements for the redirected input file.

Previous releases printed the control statements from the redirected input file immediately after the command line control statements.

2005/04/11 s208_486

OPTION PRTSUM=n

The PRTSUM keyword, with its optional numeric argument, may be coded on an OPTION statement in the control cards, or in "selcopy.nam", to control the amount of diagnostic information reported in the summary.

PRTSUM
If the optional argument is omitted, PRTSUM=1 is assumed.

PRTSUM=0
Synonym of NOPSUM (also NOPTOT), where the printing of the Summary is suppressed.

PRTSUM=1
The Summary is printed in the same format as on pre-2.09 releases, when the PRTSUM option was not supported. e.g.

```
SUMMARY..
 SEL-ID      SELTOT      FILE     BLKSIZE  LRECL           FSIZE   CI    DSN
 ------      ------      ----     -------  -----           -----   --    ---
    1            22 READ SSDYN07      20    10 FB              3       C:\djh\cc\slc\SSDYN07.INP
    2             3
    3            22
    4             3
    5             1
    6             1
    7----9        3
```

PRTSUM=2 (Default)
PRTSUM=2 is the default for Rel 2.09. It is the same as PRTSUM=1, but additionally:

1. All User Labels are also reported in the summary, with a blank line generated preceding each label.
2. Unconditional RETURN statements are identified with the string "=ret=", unless *>Comment is coded on the RET statement which is given priority. e.g.

```
SUMMARY..
 SEL-ID      SELTOT      FILE     BLKSIZE  LRECL           FSIZE   CI    DSN
 ------      ------      ----     -------  -----           -----   --    ---
                     =rdrtn=
    1            22 READ SSDYN07      20    10 FB              3       C:\djh\cc\slc\SSDYN07.INP
    2             3
    3            22 =ret=

                     =xxrtn=
    4             3
    5             1
    6             1
    7----9        3 =ret=
```

PRTSUM=3
The same as PRTSUM=2, but additionally:

1. An individual line is printed in the summary for each statement. Multiple statements sharing the same selection total are no longer combined into a single summary line.
2. IF, AND and OR statements are reported in the summary, as well as NOW, THEN and ELSE statements.
3. All statements are identified by their Operation Word, and to accommodate this extra information, the FILE column and all columns to the right of it, are shifted 6 bytes to the right.
4. Comments on all statements are reported on the summary line, regardless of whether the *> notation were used or not.

   Comments on statements in the summary are aligned between the FSIZE and CI columns, preserving the leading asterisk, and ignoring the spacing on the original.

   Comments on labels are also reported, aligned close to the LRECL column, preserving the leading asterisk, and ignoring the original spacing. e.g.

```
SUMMARY..
 SEL-ID      SELTOT            FILE     BLKSIZE  LRECL           FSIZE   CI    DSN
 ------      ------            ----     -------  -----           -----   --    ---
                     =rdrtn=                        * Comment data on a label.
    1            22   READ   SSDYN07        20    10 FB              3       C:\djh\cc\slc\SSDYN07.INP
    6            74   READ     FILE1      2048   128 U              75       C:\djh\cc\slc\LST\SSPARM11
    7            75   @l1=                                              * Comment data on a statement.
                      if                                               * Comment data.
    2             3     t do                                           * Comment data.
    3            22 =ret=                                              * Return from a subrtn.

                     =xxrtn=                        * Comment data on a label.
    4             3   pr                                               * Comment data on a statement.
                      if                                               * Comment data.
    5             1     t move                                         * Comment data.
    6             1     t move                                         * Comment data.
    7             3   move                                             * Comment data.
```

```
            8             3   pr                                          * Comment data.
            9             3 =ret=                                         * Return from a subrtn.

                          =xyz_label=                 * Just a fabricated example.
                            if                                            * If something or other.
                             a
          269             0     t mod                                     * Change something.
          270             0     t rc=                                     * Set return code.
```

2005/10/02 s208_677

Continuation Records

Support for the use of the '\' (BackSlash) character to indicate that a control statement is continued on to the next line is now fully functional in the same way as on the Mainframe version of SELCOPY.

2006/01/20 s208_862

STACK command to mimick CMS STACK feature

When not running on the CMS platform, the STACK statement will mimick the CMS STACK feature by saving stacked lines in dynamic storage.

At EOJ the saved lines will be written out to the file "selc.stk" on the current directory.

2006/09/15 s208_891

DIR for CMS includes SFS path where approp

DIR input from SLC running under CMS will report the full SFS (Shared File System) path when the input DSN is a directory on CMS's SFS.

2006/10/09 s208_891

UTIME statement supported for CMS and syntax enhanced

The UTIME statement is now supported for the CMS platform and enhanced to allow use of the DSN parameter in conjunction with the FILE parameter.

The revised syntax is:

```
 _____________________________________________________________________________
|       |                                  |                                  |
|       | [[ [FILE=] fnam ] DSN= ]         |                                  |
|       |                                  |                'yyyy/mm/dd hh.MM.ss' |
| UTIME |                       fileid     | [ FTIME = ]      n2 AT p3        |
|       |                      n1 AT p1    |                   p3,p4          |
|       |                       p1,p2      |                                  |
|_______|__________________________________|__________________________________|
```

The original syntax, which will continue to be supported on later releases, was:

```
 _____________________________________________________________________________
|       |                                  |                                  |
|       |                       fileid     |                'yyyy/mm/dd hh.MM.ss' |
| UTIME |    [ FILE = ]        n1 AT p1    | [ FTIME = ]      n2 AT p3        |
|       |                       p1,p2      |                   p3,p4          |
|_______|__________________________________|__________________________________|
```

2006/09/15 s208_891

PAGEWIDTH for PRINT file may exceed 132

The maximum value for PAGEWIDTH has been increased to 156 bytes.
The minimum value for PAGEWIDTH remains at 66 bytes. e.g.

```
opt  dw=10    pw=66      * Both valid at min values.
opt  dw=4096  pw=156     * Both valid at max values.
```

2006/09/18 s208_891

DATAWIDTH for PRINT file may exceed 100

The maximum value for DATAWIDTH has been increased to 4096 bytes.
The minimum value for DATAWIDTH remains at 10 bytes. e.g.

```
opt  dw=10    pw=66      * Both valid at min values.
opt  dw=4096  pw=156     * Both valid at max values.
```

2006/10/09 s208_891

POS UXREASCD introduced for CMS

For use under CMS, POS UXREASCD refers to the 4-byte binary "Reason Code" returned by a CMS function, in addition to the 4-byte CMS "Return Code" which is referenced by POS UXRETSYS.

For example, the UTIME statement sets the 4-byte binary fields at both POS UXRETSYS and at POS UXREASCD.

Notes:

1. Do not confuse POS UXRETSYS with POS UXRETCD. POS UXRETCD refers to SELCOPY's Return Code which usually gets set to 8 when a CMS function call fails, unless SELCOPY's return code is already higher than 8, in which case it remains unchanged.
2. The UX fields may now be referenced as values using the following:

Value Keyword	Synonyms	POS in Storage
RETCD	RETCODE,RC	UXRETCD
REASCD		UXREASCD
RETVSAM		UXRETVSAM
RETSYS	CMSRETCD,RETXV,RETCMS	UXRETSYS

e.g.

```
* The following 3 statements all mean the same thing:
 IF RETCD > 8
 IF 4 AT UXRETCD TYPE=B > 8
 IF RC > 8
```

2006/10/24 s208_893

XV statement supported for UNIX, PC and CMS

For CMS:

The XV statement is already documented in the SELCOPY User Manual for the CMS platform on the mainframe.

The Rexx return code passed back for the XV FETCH NEXT statement is as documented in IBM's "z/VM: CMS Callable Services Reference" manual for the CSL (Callable Service Library) routine, DMSCGX. It is saved by SELCOPY as a 4-byte binary number at POS RETXV. Please refer to IBM's documentation for the full list, but the more common return code values (in decimal) are:

0	Normal completion.
200	The data returned in the var value and/or var name has been truncated. SELCOPY will also set its own RC=8.
202	REXX is not active.
206	All variables have been processed (effectively EOF).
207	Unsupported function. (This return code is issued if DMSCGX is called from within an EXEC2 environment.)

For UNIX and PC:

Since s208_893, XV FETCH is now also supported for UNIX and PC platforms, using the same syntax as for CMS.

The value fetched for a supplied variable name is obtained from the environment, resulting in the same value as would be obtained from use of '%VARNAME%' as a literal on a control statement.

2006/12/06 s208_894

Ctrl-C handled to give quiet termination

Use of Control-C or Control-Break is handled by SELCOPY and the run brought to a controlled halt, with files closed properly and the full selection summary reported as normal.

An error message to indicate that the run was terminated by the operator is reported on the terminal and on the listing file.

Return Code 44 is set by SELCOPY.

```
ERROR 566 - STOP COMMAND OR ^C ISSUED BY OPERATOR
```

2006/12/06 s208_894

Abnormal termination handled

When a failure occurs due to a coding error in the SELCOPY program, such as an Access Violation or Illegal Instruction, the interrupt is handled by SELCOPY.

An error message is written to the terminal and interrupt handling discontinued.

If possible, the run is then brought to a controlled halt, with files closed properly and the full selection summary reported as normal.

The error message reporting the failure is repeated on the listing.

Return Code 88 is set by SELCOPY.

```
**ERROR** SEGV signal intercepted.  abc.............
```

The 15 byte string abc... holds information regarding SELCOPY internals and should be passed to CBL for investigation of the problem.

2006/12/11 s208_894

Command line option -NOTRAP introduced

Interrupt handling is default for SELCOPY, but may be disabled with the command line option -NOTRAP (-NOTR is a synonym).

Use of the -NOTRAP option will revert to having ^C and ^Break and any abnormal termination event handled by the system.

2007/01/28 s208_894

F=CARD has DSN reported in Summary

If CARD input is used and the SELCOPY control statements are read from a file, the DSN (data set name) of the CARD file is reported in the summary.

2007/02/16 s208_896

SLEEP statement supported for CMS

SELCOPY's SLEEP statement under CMS or MVS uses a POSIX function.

To use the SLEEP statement it is therefore necessary to have:

1. UNIX System Services started.
2. The POSIX(ON) run-time option in effect for the program.

e.g.

```
slc  POSIX(ON) / !read abc.data !sleep 10 secs !print !end
```

In the ficticious example above, the "/" is essential, indicating to the command processor the end of system run-time options. All subsequent options or parameters are passed to the program, in this case SLC.

SLC will then read the file "abc.data" off any accessed disk, or failing that, off the BFS or HFS (Byte File System for CMS or Hierarchical File System for MVS), and take 10 seconds to process each record. Very useful !

Alternatively, a SLC job could be started which re-accessed a disk, checked for the existence of a certain file or event and did something useful, otherwise sleeps for 20 mins or whatever. It could be left running continuously in a disconnected machine. e.g.

```
selcopy POSIX(ON) / -ctl=CHECKEML.CTL  -Lst=EMAIL.LST.B
```

2007/03/01 s208_898

FLUSH statement support for TAPE output

All SELCOPY's output files are buffered using a buffer of length BLKSIZE, default=2048.

Nothing is physically written until the buffer cannot accommodate another record, at which time the buffer is written out, reset to empty, then used for the record that wouldn't fit.

The FLUSH statement allows the user to control when SELCOPY issues a physical write in order to clear the buffer.

For output to a real TAPE device, the physical write will also include the writing of an Inter-Record-Gap on the tape, so the FLUSH statement becomes very important in order to control where the IRGs occur. e.g.

```
read input.fil   * Read a data record.
wr    tape.fil   * Puts curr record in buffer.
flush tape.fil   * Physically writes and clears the buffer.
```

Note:

1. SELCOPY's OPEN=NORWD and CLOSE=NORWD will have no effect. If coded they are silently ignored.

   However, when attaching a tape to a UNIX or PC system, you may well find that an option exists for choosing whether at open/close time the tape rewinds or remains where it is.

2007/06/17 s208_903

NEWBLK param to give equivalent of FLUSH

The NEWBLK parameter may be used on a WRITE statement to give the same effect as using a FLUSH statement prior to the WRITE. e.g.

```
read input.fil          * Read a data record.
wr    tape.fil newblk   * Puts curr record at start of buffer,
                        * by first flushing the buffer if reqd.
* Thus, every record will start a new block.
* Other WRITE statements may be coded, without the NEWBLK param,
* if selective blocking is reqd.
```

2007/03/13 s208_899

CSV data via COMPRESS/EXPAND statements

Comma Separated Variables may be processed with the COMPRESS/EXPAND statements:

```
 ____________________________________________________________________________________________
|             |                                                                              |
|             |  n1 AT p1         n2 AT p2                                                   |
|             |   p1, p2     TO    p3, p4                                                    |
|             |  'litval'                                                                    |
|             |                                                                              |
|             |        FLEN=n1 n2 n3 ...    * Field lengths.  Mandatory param.               |
|             |                                                                              |
|  COMPRESS   |        DLM='x'    * Field delimiter char.  Default is ',' (Comma.)           |
|             |        DELIM                                                                 |
|             |                                                                              |
|  EXPAND     |        FILL='x'   * Fill (padding) char.   Default is ' ' (Blank.)           |
|             |        PAD                                                                   |
|             |                                                                              |
|             |        ESC='x'    * Escape char.           Default is '\' (BackSlash.)       |
|             |                                                                              |
|             |        ENC='x'    * Enclosing char.        Default is '"' (DoubleQuote.)     |
|             |                                                                              |
|             |        IFNEC      * Implies ENC.  Only enclose if necessary.  (Default.)     |
|             |        STR        * Implies ENC.  Only enclose if non-numeric string.        |
|             |        ALL        * Implies ENC.  Enclose all except null fields.            |
|             |                                                                              |
|_____________|______________________________________________________________________________|
```

CSV Notes:

1. All parameters are accepted for both COMPRESS and EXPAND statements.

   IFNEC, STR and ALL however, for the EXPAND statement, only serve to imply the ENC parameter. Provided ESC is not also coded on the EXPAND statement, IFNEC, STR and ALL are silently ignored.

2. ENC and ESC parameters are mutually exclusive.

3. If ENC and ESC parameters are both omitted for a COMPRESS statement, an error is given if the source data already contains a DLM char, nothing is written to the destination area, and LRECL=0 is returned.

4. If the destination area is not large enough to contain the resultant data, RC=8 is set, further fields are ignored and an error message is reported on SELCOPY's listing file, such as:

   CMPRS Error 09: Field#20,Len=25 DataLen=00 TotDestLen=300 Destn area overflow.

5. FLEN arguments define the lengths of the fields to be processed, in order, for the COMPRESS source or the EXPAND destination.

   At least one FLEN value must be specified and multiple FLEN arguments must be comma or blank separated.

   Where the source data exceeds that defined by the FLEN args, the last FLEN arg is used for the length of all remaining fields, thus: FLEN=30,20 means the first field is length 30, and all remaining fields are 20 bytes each.

   Where the source data does not require all the FLEN args, the surplus FLEN values are ignored.

6. The STR option on the COMPRESS statement will cause leading blanks to be disregarded before validating for numeric. A totally blank field is considered non-numeric. Leading zeros, except for junior position, are suppressed in destination.

7. If an EXPAND source field is too long, the destination field is filled with asterisks, RC=8 set and an error message given on SELCOPY's listing file with details of the first field found to be too long, such as:

```
EXPND Error 04:  Field#16,Len=03  DataLen=04  TotDestLen=300  Field too long.
```

2007/05/08 s208_902

LEFT,RIGHT,CENTRE for adjusting a field.

The basic op-codes to left-adjust, right-adjust or centralize a field are `ADJL` (synonym `LEFT`), `ADJR` (synonym `RIGHT`) and `ADJC` (synonyms `CENTRE` and `CENTER`).

The destination field is always filled, blank padded as necessary, and may fully or partly overlap the source field without loss of integrity. e.g.

```
center 14 at 1    into 20 at 1
```

The destination field may be omitted, in which case the source field is adjusted in place. e.g.

```
right  20 at 1   * Right adjust on to itself.
```

Multiple blanks in the source field are condensed into 1 blank in the destination. e.g.

```
p 101 = " xxx   yyyy zz"           *         ....,....1....,....2
left   101,114    into 1,20        * To get "xxx yyyy zz          "
```

2007/05/09 s208_902

SUMPRT is synonym for PRTSUM

For convenience and the benefit of those users who prefer to guess SELCOPY's keywords rather than look them up, SUMPRT may now be coded for PRTSUM. e.g.

```
opt  prtsum=3       * To get a full summary, identifying each statement with
                    * its opcode,  together with any comments coded.
```

May be coded as:

```
opt  sumprt=3
```

2007/05/16 s208_902

OPTION CALLTYPE=DIRECT|VIA_SLCCALL

OPT CALLTYPE may be coded in the "selcopy.nam" file, and/or in individual control statement files, to control the type of linkage used by subsequent CALL statements calling user-written routines in a run-time Shared Library.

The default call type is DIRECT and is recommended.

CALLTYPE arguments are:

DIRECT (The default, which is recommended.)
means that user-written routines in the shared library, will be called directly with just 1 argument, viz: a pointer to an array of char string pointers.

The Shared Library to be used will be:

- ◊ "libselc.so" (default for UNIX)
- ◊ "libselc.dll" (default for Windows)
- ◊ "your_chosen_name" as set with an OPTION LIBNAME statement.

VIA_SLCCALL (Legacy.)
means that user-written routines in the shared library, will NOT be called directly. Instead, the legacy linkage is used, where your user-written routine, "slccall", is called with 2 arguments:

1. A pointer to a null terminated char string holding the name of the required routine.
2. A pointer to an array of char string pointers.

Your "slccall" function must then call the required routine with whatever arguments it requires.

Use of OPT CALLTYPE=VIA_SLCCALL will always use, regardless of any changes made using an OPT LIBNAME statement, the legacy library name:

- ◊ "slccall.dll" for Windows.
- ◊ "slccall.so" for most UNIX.
- ◊ "libselc.sl" for HP UNIX.

However, the library is still loaded dynamically from any directory on the library path.

The directory, "/usr/selcopy" is no longer searched unless mentioned on the environment variable LD_LIBRARY_PATH (or LIBPATH for AIX).

Although not recommended, different CALLTYPE settings may be used for different CALL statements in the same run, by preceding the CALL statement with an OPTION CALLTYPE statement.

However, once the first CALL statement is encountered, the Shared Library is opened using the library name defined by the last OPTION LIBNAME statement, if any, otherwise the default library name for the current CALLTYPE setting is opened. Any further OPTION LIBNAME statements will have no effect.

All routines used, regardless of linkage types, must be in the same library. Only 1 shared library is supported.

See also OPTION LIBNAME below.

2007/05/16 s208_902

OPTION LIBNAME="libnam"

OPTION LIBNAME='libnam' may be used to override SELCOPY's default Shared Library name with 'libnam' and may be coded in the "selcopy.nam" file, and/or in individual control statement files.

If used, the LIBNAME option should precede all CALL statements.

The argument, "libnam", must be provided in single or double quotes. e.g.

```
OPT LIBNAME="slccall.so"   * Use the legacy libname for DIRECT calls.
```

The "libnam" may be changed repeatedly on different OPT statements, but will have no effect until the first CALL statement is encountered.

When the first CALL statement is encountered, the Shared Library is opened, using the library name applicable to the current CALLTYPE setting:

DIRECT (The default.)
When OPTION CALLTYPE=DIRECT is set, or no CALLTYPE option has been coded, the Shared Library is opened using the library name defined by the last OPTION LIBNAME statement, if any, otherwise using the default library name:

◊ "libselc.dll" for Windows.
◊ "libselc.so" for all UNIX.

VIA_SLCCALL
When OPTION CALLTYPE=VIA_SLCCALL is set, whatever has been coded on previous OPT LIBNAME statements will have no effect. The library name used is fixed at the legacy library name for your platform:

◊ "slccall.dll" for Windows.
◊ "slccall.so" for most UNIX systems.
◊ "libselc.sl" for HP UNIX.

Any further OPTION LIBNAME statements following the first CALL statement will have no effect.

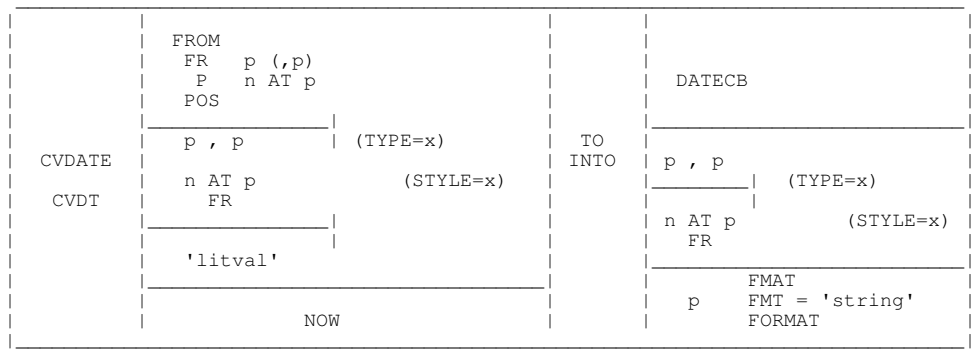
Whatever CALLTYPE is set, the shared library is still loaded dynamically, searching for it on any directory on the library path. Thus, it is no longer necessary to keep your shared library on the directory, "/usr/selcopy". If the CALL statement is not used, the shared library need not even exist.

The directory, "/usr/selcopy" is no longer searched unless mentioned on the library path environment variable, LD_LIBRARY_PATH (or LIBPATH for AIX).

See also OPTION CALLTYPE above.

2007/06/13 s208_903

CVDATE statement for converting date formats.



Style codes for the CVDATE statement:

I	International.	ccyymmdd (Default)
A	American.	mmddccyy
B	British.	ddmmccyy
J	Julian.	ccyyddd
D	Days since 1900/01/01.	nnnnnnnnn
T	TOD STCK format.	8-byte binary. (IBM Mainframe.)
N	NOW.	Current Date/Time. (Source only.)
F	FORMAT supplied.	(Destination only.)
C	DATE Control Block.	(Destination only.)

Data Types for CVDATE statement:

C	Character. (Default if TYPE not coded anywhere)
P	Packed Decimal.
U	Unsigned Packed Decimal.
B	Binary. (Big Endian.)

FORMAT keywords for CVDATE statement:

Len	Keyword	Description	Example
9	"Mmmmmmmmm"	Month Word	"January "
?	"Mmmzzzzzz"	Month Word	"January" (Blanks truncated.)
9	"Ddddddddd"	Day Word	"Sunday "
?	"Dddzzzzzz"	Day Word	"Sunday" (Blanks truncated.)

4	"ccyy"	Year	"2007"
4	"yyyy"	Year	"2007"
4	"ddth"	Day of month	" 2nd"
?	"zdth"	Day of month	"3rd"
3	"Mmm"	Month Word	"Jan"
3	"MMM"	Month Word	"JAN"
3	"Ddd"	Day Word	"Sun"
3	"DDD"	Day Word	"SUN"
3	"ddd"	Day of year	"123"
2	"ww"	Week of Year	"22"
2	"cc"	Century	"20"
2	"yy"	Year	"07"
2	"mm"	Month	"01"
2	"dd"	Day of Month.	"31"
1	"D"	Day of Week.	"1" (Sunday=1 .. Saturday=7)
1	"d"	Day of Week.	"1" (Monday=1 .... Sunday=7.)

Notes:

1. The FORMAT string is case-sensitive and processed left to right.
2. When blanks are truncated, the rest of the format is shifted left.
3. D and d codes are only obeyed if adjacent chars are non-alpha.

NOW

The NOW parameter may be used only as a source field and returns today's date.

DATECB

The DATECB parameter may be used only as a destination field. All date fields in the DATE Control Block are refreshed with the date supplied in the source field.

If the source field is NOW, all time fields are also refreshed, otherwise all time fields in the DATE Control Block are zeroized.

Examples

```
move 10 from now     to 120   * Illegal - NOW is not a real position.  Only for CVDATE.
move 10 from DATE-2  to 120   * Gives "yyyy/mm/dd", as in previous releases.
CVDATE   NOW    to DATECB     * Refresh DATE Control Block, as in previous releases.

CVDATE '2007/02/01'          TO 07 AT 131 STYLE=J TYPE=C   * -> '2007032'

CVDATE '2007/02/01'          TO  6 AT 131 STYLE=J TYPE=P   * -> x'0000 2007 032F'
                              * Source still defaults to TYPE=C as it's a literal.

p 1 = '2007/06/30'            * Character source.
CVDATE 10 at 1               TO 15 AT 131 STYLE=J TYPE=U   * Unsigned Packed Dec destination.
                    * Above will fail with RC=8 as no TYPE coded for source, so it
                    * defaults to the same as the destination.
CVDATE 10 at 1  TYPE=C       TO 15 AT 131 STYLE=J TYPE=U   * Works ok.  Source will
                                                           * still default to STYLE=I.

cvdate   NOW  to prec+40                fmt='d Ddd, Mmm ddth yyyy'
                              * Could give: '7 Sun, Jul  1st 2007'

CVDATE 'June 30th 2007'  STYLE=A  to DATECB  * Refresh DATE Ctl Block, zeroize time flds.

CVDATE 10 at 1  to 141  fmt='Day d Ddd Mmm ddth ccyy  Dddzzzzzz ddth Mmmzzzzzz yyyy'
                           'Day 6 Sat Jun 30th 2007  Saturday 30th June 2007      '
```

2007/06/17 s208_903

CVPP statement for changing size of a p.d. field

The CVPP statement may be used as a convenient way to copy a packed decimal field into a different size p.d. field.

If the destination field is too small, the source data is truncated to fit the destination and RC=8 is set.

No validation is performed on the individual decimal digits of the source.

2007/06/17 s208_903

POS SCALE introduced

POS SCALE refers to the bottom line of dots used internally by SELCOPY on its PRINT output, which can be useful to the user.

The size of the SCALE line is equal to the current setting of DATAWIDTH. e.g.

```
print from scale   l=22
```

gives:

```
....,....1....,....2..
```

2007/09/22 s208_906

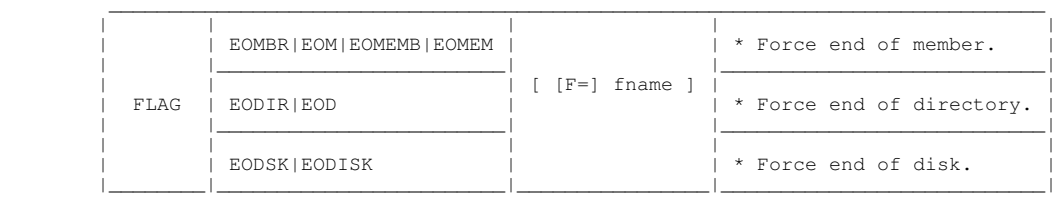
FLAG EOMBR/EODIR/EODSK supported

For UNIX and PC platforms, the FLAG statement previously only supported FLAG EOMEMB which could be used for DIRDATA input.

The FLAG statement is now supported as described for the mainframe in the SELCOPY User's Manual, with 2 exceptions:

1. FLAG EODIR is also supported on the CMS platform for the SFS (Shared File System) which offers a directory structure.
2. On the CMS platform, FLAG EOD is no longer treated as a synonym for FLAG EODISK, but as FLAG EODIR.

Syntax:



The FLAG statement may only be coded for a file which is being read with a DIR or DIRDATA parameter coded. e.g.

```
read  'CDEF:/*.txt'  dirdata  sub=4          * PC or CMS files.
read  '/mnt/l07/c/djh/test*.???'  dir sub=4  * UNIX or Open MVS or VM files.
```

FLAG EODSK
: is applicable to systems, such as CMS and the PC, that have a filesystem which supports disk identifiers.

  However, if FLAG EODSK is coded for a UNIX-type filesystem then, as well as well as bypassing the current dir, all subdirs for that dir will also be bypassed. Use of FLAG EODSK on a UNIX-type filesystem can save a lot of I/O and CPU time.

FLAG EODIR
: is applicable to systems, such as UNIX, CMS's SFS and the PC, that have a filesystem supporting a directory structure. Has no effect on a native CMS mask.

Use of the FLAG statement with DIRDATA input can be very powerful. e.g.

```
* Restrict the nesting of directory levels to 4, and for the disk
* letters C, E and G only, print the dir entry and 1st 3 records of
* every '.txt' file, but bypass all further files in a directory as
* soon as a file with 'xyz' in its name is found, and bypass all
* further files in all further directories on the current disk as soon
* as a file with 'pqr' in its name is found.

read  'CEG:/*.txt'  dirdata  sub=4

if data                * If current rec is member data.
and incount > 2
  then FLAG  EOMBR     * Force end of member for data recs.

if dir                 * If current rec is a directory entry.
thenif  pos any = 'xyz'
  then FLAG  EODIR     * Force end of current dir.

if dir                 * If current rec is a directory entry.
thenif  pos any = 'pqr'
  then FLAG  EODISK    * Force end of current disk.

print  * Print the current record, which may be a DIR entry,
       * or 1 of the 1st 3 DATA records of each file chosen.
```

2008/01/17 s208_908

Floating Point Conversion Char <-> Float (Native)

The UNIX/PC version of SELCOPY has been brought up-to-date with the mainframe version with regard to the CVCF and CVFC statements for converting to and from Character and Floating Point fields. e.g.

```
cvcf 20 at 1    to 8 at 101                * Char to Floating Point.

cvfc  8 at 101  to 1  format='z,zz9.99999-' * Floating Point to Char.
cvfc  4 at 101  to 1  format='ss,ss9.99999' * Floating sign on left.
cvfc  101,108   to 12 at 1                 * To get Zoned decimal.
```

Floating Point fields are:

- Length 4 (Single precision)
- Length 8 (Double precision)
- Length 16 (Extended precision) are not supported by SELCOPY.

By default, Floating Point fields are assumed to be in the native style of the machine architecture on which SELCOPY is running.

For character to float, the value "-0" will be stored as 0, losing the sign.

There are 2 distinct styles of encoding systems, both used widely on various architectures, but having different bit representations for the floating point fields. Both encoding systems are supported by SELCOPY.

HEX - IBM Base 16 - Hexadecimal Format:
7 bits for a HEX exponent for all (4, 8 or 16-byte) floating point fields. The integer portion of the value stored is always 0. Used predominantly on IBM mainframe platforms.

BIN - IEEE-754 Base 2 - Binary Format:
8 bits for a BIN exponent for a single (4-byte) floating point field. 11 bits for a BIN exponent for a double (8-byte) floating point field. The integer portion of the value stored is always 1, but the 1 bit is not stored in the floating point field because it can be implied. Used predominantly on UNIX and PC platforms.

Common to both:
The senior bit represents the sign: 0=Positive, 1=Negative. The exponent follows immediately after the sign. Excluding the sign and the exponent, the remaining junior bits represent the fractional part of the number.

Precision:
Only 10 places of decimal after the decimal point are returned by SELCOPY for a float to character conversion.

While it is possible to convert an 8-byte floating point variable to character format with more than 300 places of decimal, some sensible limit has to be applied.

SELCOPY is used predominantly in the commercial environment, so the limit of 10 places of decimal after the decimal point would seem more than enough.

On the integer side, to the left of the decimal point, there is no limit, other than that imposed by the user's FORMAT parameter.

If the resultant value is too large to fit in the FORMAT, the destination string is filled with asterisks and RC=8 is set.

Please refer to the main SELCOPY User Manual under the CVFC parameter description for more on precision.

2008/01/28 s208_908

Floating Point Conversion BIN <-> HEX (Base 2<->16)

Data from an alien platform, with a different style floating point encoding system, will always give problems. Treating HEX float fields as a BIN can give ridiculous values when converted to decimal representation in a formatted character string, and vice versa. But even worse, may just give the wrong answer.

To process alien floating point fields, the CVFC and CVCF statements support use of the TYPE parameter with 2 arguments, the first being F to indicate floating point, and the 2nd indicating the style of the encoding system required for the conversion. e.g.

```
cvcf 20 at 1              to 8 at 101 TYPE=F HEX      * Char to Floating Point.
cvfc  8 at 101 type=f hex  to 1  format='z,zz9.99999-' * Floating Point to Char.
```

Supported keywords for the style of Floating Point representation are:

HEX or HFP	IBM Base 16 Hexadecimal style.
BIN or BFP	IEEE-754 Base 2 Binary style.
NAT or NATIVE	The native style of the host machine, regardless of any user or installation setting of the default.

The CVFF statement may be used to convert a floating point variable from one style to another. The size of the variable may also be changed. e.g.

```
cvff  8 at 101 type=f HEX  to 8 at 201 type=f BIN      * IBM to IEEE format.
cvff  4 at 101 type=f bin  to 8 at 201 type=f hex      * IEEE to IBM format.
```

2008/01/28 s208_908

Floating Point Default Style

By default, Floating Point fields are assumed to be in the native style of the machine architecture on which SELCOPY is running.

However, the default style for Floating Point may be changed, using the DEFAULTFP parameter coded on an OPTION statement, which may be provided, by the user on an earlier control statement, or by the "selcopy.nam" file defining the installation defaults.

DEFFP and DFLTFP are synonyms for DEFAULTFP.

DEFAULTFP takes 1 mandatory argument which defines the floating point style required to be used as default. The keywords are the same as may be coded as the 2nd argument of TYPE=F. e.g.

```
OPTION  DEFFP=HEX   * Default to IBM Base 16 Hexadecimal style.
option dfltFP=BIN   * Default to IEEE-754 Base 2 Binary style.
opt defaultFP=NAT   * Default to native style.
```

Setting DEFAULTFP=NAT would of course only be necessary in order to override any OPTION statement which may have been coded in the "selcopy.nam" file. e.g.

```
          * Coded in "selcopy.nam" file.
opt noban deffp=hex * Suppress banner and default to HEX Floating Point.

          * Coded in user's control statements.
opt deffp=nat       * Revert to NATIVE Floating Point as default.
cvcf 20 at 1    to 8 at 101                 * Char to Floating Point NAT.
cvfc  8 at 101  to 1  format='z,zz9.99999-'  * NAT Floating Point to Char.
```

Caution:
Native floating point is used by SELCOPY for all its floating point arithmetic and conversion. Using an alien style as default will cost CPU time.

All new applications should be designed such that files with floating point data use the style native to the host machine or their intended target machine.

2008/01/31 s208_908

Floating Point Arithmetic

Arithmetic operations may be performed on Floating Point fields. e.g.

```
equ src1      13 at   1 type=c   * Source value 1.
equ src2      13 at  16 type=c   * Source value 2.

equ flt1       8 at 201 type=f   * Field 1 as a native float.
equ flt2       8 at 211 type=f   * Field 2 as a native float.

equ o_addn    31  fmt='sss,sss,ss9.999,999'   * Results.
equ o_mult    52  fmt='zzz,zzz,999.999,999s'  *
equ o_divn    73  fmt='zzz,zzz,zz9.999,999+'  *


add  src1    to src2   into o_addn
mult src1    by src2   into o_mult
div  src1    by src2   into o_divn

add  flt1    to flt2   into o_addn
mult flt1    by flt2   into o_mult
div  flt1    by flt2   into o_divn
```

The use of the EQU statement above is for clarity only. For instance, it would be equally acceptable to code the ADD statement as:

```
add 13 at 1 type=c   to 13 at 16 type=c   into 31 fmt='sss,sss,ss9.999,999'
```

2008/02/13 s208_909

Floating Point Literals

Any numeric literal for a source field, containing a single decimal point (period or full-stop), will be treated as a double precision (8-byte) native floating point number.

Any arithmetic operation performed using that literal will then use floating point arithmetic. e.g.

```
add 1234.56  to 6543.21012   into 201 fmt='zz,zz9.99999s' * Gives: ' 7,777.77012+'
```

If the destination field is not a floating point or a formatted character field, the result will be rounded by adding 0.5 and the fractional part will be dropped. e.g.

```
add 1234.56  to 6543.21012   into 201,204 type=p          * Gives:   x'0007,778C'
```

Numeric literals may be signed. e.g.

```
add -1234.56 to -6543.21012  into 201 fmt='ss,ss9.99999'  * Gives: '-7,777.77012'
```

Numeric literals may be enclosed in quotes. e.g.

```
add '-1234.56' to '-6543.21012' into 1 fmt='ss,ss9.99999' * Gives: '-7,777.77012'
```

Quoted numeric literals may use punctuation in the same way as described in the SELCOPY User Manual under the CVCP parameter description. e.g.

```
add '-1234.56' to '-6543.21012' into 1 fmt='ss,ss9.99999' * Gives: '-7,777.77012'
```

2008/03/24 s208_910

Floating Point Conversion Char (Exponent) --> Float

Exponent notation may be used for the source field of the CVCF statement for converting from Character to Floating Point. e.g.

```
p 1,20 = '1.23456789e-22'
cvcf 20 at 1    to 8 at 101     * Char to Floating Point.
cvcf '5.0e3'    to 8 at 201     * Literal to Floating Point.
```

Exponent notation for the CVFC statement is not supported.

2008/04/01 s208_911

Floating Point Conversion - Rogue Lengths

For compatibility with SELCOPY 2.02 on the IBM mainframe, the length of a floating point variable, used on a CVFC or CVCF statement, will be tolerated if it does not exceed 8, even if not a valid length (4 or 8) for a floating point variable as defined by the native machine architecture. e.g.

```
cvfc  3 at 1    to 8 at 101     * Rounded to 4-bytes with junior x'00'.
```

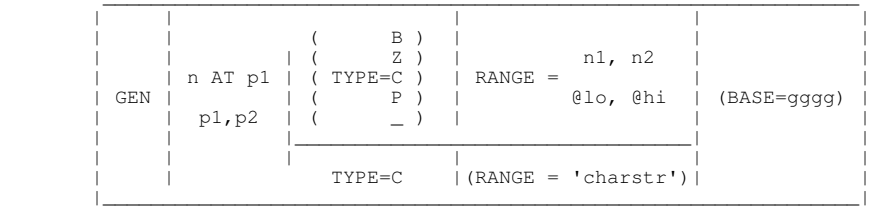
A field with a non-standard length will be either truncated from the right, or padded on the right with binary zeros

The non-standard lengths, 1, 2, and 3, are left-adjusted in 4 bytes of binary zeros, and the non-standard lengths, 5, 6 and 7, left-adjusted in 8 bytes of binary zeros.

```
<!dos  slc208 !opt w 2222   !p 1 '1.23456'   \\
              !cvcf 9 at 1   to 3 at 31      \\
              !cvcf 9 at 1   to 4 at 41      \\
              !plog ty=m  s=5         !e
<e  selc.lst
```

2008/02/06 s208_908

GEN statement supports variable RANGE @LOW,@HIGH



The RANGE parameter has been improved to allow dynamic setting of the low and high values to be generated.

Straight numeric values, n1 an n2, are still accepted as before, but the user may now supply the lower an upper values of the field using @ pointer variables which may be set under program control from elsewhere. e.g.

```
read card
@abc = 20
@xyz = 6 at 1 type=c
gen   4 at 11 type=b   range=@abc,@xyz  base='TEST'
print fr 1,14    type=d
end
123456   This is the upper limit for the RANGE reqd.
/*
```

2008/01/27 s208_908

CVCH supports FORMAT'd conversion to Printable Hex

The FORMAT parameter may be used on the CVCH statement to allow punctuation in the display of data in hex representation.

All characters in the FORMAT, except upper and lower case X, are copied unchanged to the destination field.

Upper and lower case X characters in the FORMAT are replaced in the destination field by the appropriate hexadecimal character from the source. e.g.

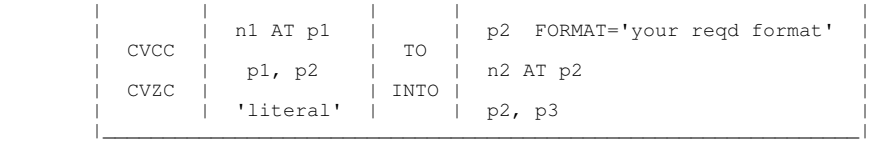
```
cvch 4 at 11   to 21 fmt='xxxx,xxxx'  * To make output more readable.

cvch 'abcd' ascii     to 1 fmt="  'xx,xx xx,xx' = ASCII code.  "
cvch 'abcd' ebcdic    to 1 fmt="  'xx,xx xx,xx' = EBCDIC code. "
```

Thus, comment information may be supplied anywhere within the format, but with the limitation that the letter X may not be used.

2008/02/10 s208_908

CVCC|CVCZ|CVZC conversion statements supported



CVCC (synonym CVZC) will extract the decimal numeric portion of the character source field, n1 AT p1, respecting the decimal point (period or full-stop) in the source if present. Without a decimal point, the source is treated as integer only.

See the SELCOPY User Manual, under the CVCP parameter description, for rules governing the extraction of numeric literals from a character source field.

The resultant decimal value is then used to fill the character destination field.

If a FORMAT parameter is supplied, the result is formatted according to the supplied format, respecting the decimal point in the format if present.

See the SELCOPY User Manual, under the FORMAT parameter description, for details.

If truncation of the fractional part of the result is necessary in order to respect the decimal point, the resultant value is rounded. e.g.

```
cvcc   '12.34567'  to 201 fmt='ss,ss,999'    * Gives:     +012
cvcc   '19.99567'  to 201 fmt='ss,sss.99'    * Gives:   +20.00 (Rounded.)
```

Without a FORMAT parameter, when destination position and length are used, instead of position and FORMAT, a default FORMAT of all 9's, for the length specified, is used and the sign is only indicated when the result is negative.

A negative result has the junior decimal digit translated to zoned, such that: a junior decimal digit of "1" is translated to "A", "2" to "B" and so on to "9" to "I". The exception is "0" which is translated to "}". e.g.

```
cvcc   '19.99567'  to 8 at 201      * Positive gives: '00000020'  (Rounded.)
cvcc  '-19.99567'  to 8 at 201      * Negative gives: '0000002}'  (Rounded.)

* The default FORMAT of all 9's has no decimal point defined, so the
* value is rounded and truncated to its integer portion only, and then
* used to fill the destination.
```

CVCZ (synonym CVZZ) operates in the same way as the CVCC statement, but does not allow use of the FORMAT parameter for the destination field.

2008/03/07 s208_909

RECFM=V3 supported for FTP Block Mode files

When FTP is used with "MODE B" to copy a mainframe platform RECFM=VB file to a UNIX or PC platform, the BDW (Block Descriptor Word) at the start of each block and the RDW (Record Descriptor Word) at the start of each logical record are stripped off and replaced with a 3-byte prefix for each logical record.

The first byte of the prefix holds flags to indicate:

128	End of data block is EOR
64	End of data block is EOF
32	Suspected errors in data block
16	Data block is a restart marker

The next 2 bytes hold the length of the logical record.

See: http://www.ietf.org/rfc/rfc0959.txt for details of FTP Transmission MODES.

Such a file can of course be read with SELCOPY using existing syntax: e.g.

```
lrecl = 3                 * Set length to read next time with EOL=NO.
read FTPFILE  eol=no      * Read the next record prefix.
lrecl = 2 at 1  type=b    * Get length of the following logical rec.
read FTPFILE              * Read the next logical record.
```

However, the above is cumbersome, so RECFM=V3 is now supported. e.g.

```
read FTPFILE   recfm=v3        * Read the next logical record.
```

Assuming that OPTION NORDW is in effect, the 3-byte prefix will be suppressed and only the logical data will be returned. The LRECL value will be that of the data only.

If the prefix information is required, the RDW parameter should be used. e.g.

```
read FTPFILE   recfm=v3  rdw  * Read the next logical record.
```

The record returned will then include the 3-byte prefix with the logical data, and the LRECL value will be that of the prefix + data.

Note:
The default for RECFM=V|V2|V3|MFV files may be RDW or NORDW depending on your installation's standards in the "selcopy.nam" file.

2008/03/17 s208_910

CVxx statements tolerate conflicting TYPE codes

The EQU statement is often used to define a data field, in order to save laborious repetition of multiple keywords, and at the same time give a more meaningful name to the field. e.g.

```
equ srce   8 at 201 type=f hex     * An 8-byte IBM floating point value.
equ dstn   8 at 211 type=f bin     * An 8-byte IEEE floating point value.
equ pr_h        301 format='Base16 Field = xxxx,xxxx xxxx,xxxx'
equ pr_b        341 format='Base2  Field = xxxx,xxxx xxxx,xxxx'
```

It is desireable therefore to use that name whenever the field is referenced, but this was not previously possible, due to the presence of the TYPE keyword.

For all CVxx statements, the restriction has been removed by giving priority to the data types defined by the xx of the CVxx statement, regardless of what may have been coded for the TYPE parameter. e.g.

```
cvff  srce   to dstn    * No problem - The TYPE codes match.
cvch  srce   to pr_h    * Diff TYPE codes. Source is not CHA and
cvch  dstn   to pr_b    * destination is not HEX, but both are accepted.
```

2008/04/09 s208_911

FORMAT string for CVxx may be dynamic

e.g.

```
equ varfmt   p 101
varfmt = 'xxxxxxxx,xxxxxxxx'   s=1
@slen = 6                   * Source length, which may vary.
@flen = @slen + @slen       * Reqd length for FORMAT.
if @slen > 4
  t @flen = @flen + 1       * Extra 1 for the comma.

cvch   @slen at 201   to 301    FMAT = @flen at varfmt
```

2008/09/23 s208_913

SQL DataBase Access with ODBC

Topic Summary:

- SSN = SubSystemName
- USER = the_id_of_the_authorized_user
- ODBCPASS = the_user_password
- OPTION CBLSQLOG [= fileid]
- HEADER parameter for ODBC tables
- POS FHDR - Special Position
- Column Names as fields
- READ - Type 1
- READ - Type 2
- READ - Type 3
- ODBC Operation
- UPDATE of current row
- DELETE of current row
- Prepared INSERT

Not Implemented for ODBC:

- `POS SQLCA`, `POS SQLDA`, `POS SQLMA`
- `POS RPL`

Support for a variety of proprietary database products has been introduced using ODBC (Open Database Connectivity).

ODBC is an API (Application Programming Interface) that SELCOPY uses to access most of the popular databases supporting SQL (Structured Query Language).

Each DBMS (DataBase Management System) has its own driver which translates the SQL statements as required, so the same commands should work on all DBMSs.

Where possible, SELCOPY's Control Card Syntax follows what is already documented in the current SELCOPY User Manual for IBM's DB2 database on the mainframe running under z/OS. Please see:

`http://www.cbl.com/sman/smfrmdb2.html`

There are of course certain omissions and differences.

SSN = SubSystemName

For database access, SSN, the Sub-System Name, is mandatory in order to identify the database. Unlike z/OS DB2, it is not restricted to 4 bytes.

The SSN may be provided, in order of priority, on:

1. The I/O statement requesting access to the database.

```
READ [fname] TAB=tablename FMAT='col7, col2, etc'   SSN=oracle1
```

2. An OPTION statement coded earlier in the current control statements.

```
OPT   SSN = ODBC_Data_Source_Name
```

3. An OPTION statement coded in the "selcopy.nam" file where installation standards are defined.

```
OPT   SSN=ODBC_Data_Source_Name
```

Note:

1. SSN refers to the "ODBC Data Source Name" as defined in the database configuration setup using, on the Windows platform for example, "Microsoft ODBC Administrator." The SSN defined here may well differ from the native SSN of the physical database to which it refers.

USER = the_id_of_the_authorized_user

For database access with ODBC, USER=userid is mandatory.

USER may be provided, in order of priority, on:

1. The I/O statement requesting access to the database.

```
READ TAB=tab1 FMT='*' SSN=oracle1  USER=joe
```

2. An OPTION statement coded earlier in the current control statements.

```
OPT   USER = JOE
```

3. An OPTION statement coded in the "selcopy.nam" file where installation standards are defined.

```
OPT   USER=Guest
```

ODBCPASS (OPASS) = the_user_password

For database access, ODBCPASS=password or its synonym, OPASS, is mandatory.

ODBCPASS may be provided, in order of priority, on:

1. The I/O statement requesting access to the database, in which case the additional synonyms of PASS, PASSWD or PASSWORD may be used.

```
READ TAB=tab1 FMT='*' PASSWD=joepass
```

2. An OPTION statement coded earlier in the current control statements.

```
OPT   OPASS = joepass
```

3. An OPTION statement coded in the "selcopy.nam" file where installation standards are defined.

```
OPT   ODBCPASS=Guestpass
```

PASS, PASSWD and PASSWORD keywords are disallowed on an OPTION statement because they conflict with their existing use in conjunction with the SITE keyword.

OPTION CBLSQLOG [= fileid]

The CBLSQLOG parameter may be used on an OPTION statement to indicate to SELCOPY that a log file is to be created containing a record of calls made to the ODBC Driver Manager.

LOGSQL and SQLLOG are synonyms for CBLSQLOG.

An optional argument, which **must** be quote delimited (single or double), may be provided to define the fileid of the log file to be created. If no argument is provided, the default fileid of "selcSQL.log" is used for the SQL LOG file.

OPTION CBLSQLOG may be supplied to SELCOPY, either within the current control statements, or in the selcopy.nam file. e.g.

```
opt  logsql='lst/SSORA16.log'
```

Note:

1. SQL error messages can sometimes be very long, resulting in truncation where shown on SELCOPY's listing output.

   However, no truncation occurs on the CBLSQLOG file or on the message written to the terminal at execution time.

HEADER parameter for ODBC tables

HD, HDR and HEAD are synonyms for HEADER on a READ TAB=tabname statement.

Use of the HEADER parameter on a READ statement for an ODBC table will cause SELCOPY to generate pseudo records 1 and 2, which will provide the user with a row of column headings for the 1st record, and a 2nd record of underlining for the header. By default, the 2 header records are not generated.

If HEADER is coded, the first 2 records returned are not genuine data records. They are:

1. A heading line consisting of the selected column names, each blank padded to the full length of the column, and, provided SEP=NO has not been coded, terminated with the separator character, which by default is set to "|".

   If sufficient blank space is available within the column width, the length of that column is also shown, for example: "COL_NAME (14) "

2. Underlining for the heading line using '-' (minus) signs for the full data width of each column and with the separator character in the appropriate position terminating each column.

Note:

1. SELCOPY treats the generated pseudo records as records 1 and 2. However, they do not exist on the database. SELCOPY has simply generated them from info obtained from the SQL server. e.g.

```
  read fnam1  tab=TABL  into 1001   upd='COLOR, DATE_SOLD, PRICE, NOTES'   HDR
  if in > 2    fnam1     * Essential check on input record number.
    t DEL fnam1          * Must not delete the header and underlining.
  pr  fr 1001  datawidth=52
```

```
INPUT   SEL SEL                                                      RECORD
RECNO   TOT ID.          1         2         3         4         5    LENGTH
-----   --- --- ....,....0...,....0...,....0...,....0...,....0..  ------
    1     1   8 COLOR     |DATE_SOLD |PRICE  |NOTES (20)          |      51
    2     2   8 ----------|----------|-------|--------------------|      51
    3     3   8 Crimson   |2008-09-09|5.55   |ISO date format     |      51
    4     4   8 Grey      |2008-12-31|395.22 |Only 1 sale         |      51
    5     5   8 Violet    |2008-09-12|14.99  |Sweet               |      51
    6     6   8 Rose      |2008-08-29|3.62   |Gentle              |      51
```

2. Do not try to update or delete the header or underline records, otherwise you get something like:

```
SQLrc=-1  SQLstate=42828  SQLcode=-510
SQLerror=[IBM][CLI Driver][DB2/NT] SQL0510N
         UPDATE or DELETE is not allowed against the specified cursor.
```

3. An empty table will show on the SELCOPY Summary as having 2 rows. In the above example, the 4th database row is shown as record 6. Thus, use of the HEADER parameter for ODBC tables can be misleading, but an alternative exists for obtaining the column names.

   See the Special Position, POS FHDR, below.

POS FHDR - Special Position

POS FHDR is already supported for MFC (Micro Focus Cobol) files.

Its use has now been extended to cover ODBC tables.

When an ODBC table is opened, SELCOPY interrogates the database for information regarding that table, from which it builds a header record in dynamic storage containing all column names selected from the table. The chosed (or default) separator character is used to separate each colum name in the header record.

Immediately following the generated header record is a record of equal length which may be used as underlining for the header record.

The underlining record consists of all minus signs, except for the separator characters which match those in the header record. e.g.

```
 1.    read fnam1  tab=djh_db2_test01   w=222  user=xx odbcpass="xx" ssn=xx
 2.    pr  from FHDR        s=1  * Header.
 3.    pr  from FHDR+LRECL  s=1  * Underlining.
 4.    pr      dw=52             * Reduce datawidth.
```

```
INPUT   SEL SEL                                                      RECORD
RECNO   TOT ID.          1         2         3         4         5    LENGTH
-----   --- --- ....,....0...,....0...,....0...,....0...,....0..  ------
    1     1   2 COLOR     |DATE_SOLD |PRICE  |NOTES (20)          |      51
    1     1   3 ----------|----------|-------|--------------------|      51
    1     1   4 Crimson   |2008-09-09|5.55   |Slash in date       |      51
    2     2   4 Grey      |2008-12-31|395.22 |Only 1 sale         |      51
    3     3   4 Violet    |2008-09-12|14.99  |Sweet               |      51
    4     4   4 Rose      |2008-08-29|3.62   |Gentle              |      51
              ....,....1...,....2...,....3...,....4...,....5..
```

The above run would of course print nothing if the table were empty. End-of-File for fnam1 would cause immediate EOJ. To print from FHDR for an empty file, an IF EOF test must be used. e.g.

```
 1.    read fnam1  tab=djh_db2_empty    w=222  user=xx odbcpass="xx" ssn=xx
 2.    pr  from FHDR        s=1  * Header.
 3.    pr  from FHDR+LRECL  s=1  * Underlining.

       if eof  fnam1
 4.      t pr '=eof='
 5.      t eoj

 6.    pr   dw=52   * Print a data row, with reduced datawidth.

INPUT   SEL SEL                                                      RECORD
RECNO   TOT ID.          1         2         3         4         5   LENGTH
-----   --- --- ....,....0....,....0....,....0....,....0....,....0..  ------
    0     1   2 SELLDAT   |COLOR     |PRICE  |NOTES (20)           |      51
    0     1   3 ----------|----------|-------|---------------------|      51
    0     1   4 =eof=                                                    51
                ....,....1....,....2....,....3....,....4....,....5..
```

Column Names as fields

When a READ or INSERT statement for an ODBC table is encountered, SELCOPY will obtain a description of each column referenced by the FMT parameter, or of all columns if the FMT parameter were omitted.

Subsequent control statements may then use any of the column names as declared fields, each having a position, a length and a data type, assuming that the INTO=n or FROM=n parameter were not used for the READ or INSERT statement.

If the INTO=n or FROM=n parameter were used, the READ or INSERT statement would refer to row data at POS=n of the workarea, in which case, any reference to a field (column) name would have to be adjusted by +n-1 in order to access the correct field. e.g.

```
opt  w=2222        ssn=xx  user=xx  odbcpass="xx"
equ  irec  1001

read fnam1  tab=test_table   into irec   upd='COLOR, NOTES'
pr  fr irec       * Print every row.

if  COLOR+irec-1 = 'Blue '       * Was irec+0.
 a  NOTES+irec-1 = 'Trendy '     * Was irec+30.
  t NOTES+irec-1 = "Modified. "
  t COLOR+irec-1 = 'Turquoise '
  t UPD fnam1   from irec
  t pr          from irec
  t space 2
```

Note:

1. Early versions of column name support gave the offset of the field, rather than the position. This has been changed in order to give the benefit of being able to to refer to the name only, instead of name+1, when the row was read into position 1 of the workarea by omitting the INTO parameter.

   Also, it was necessary to code POS or P in front of any column name that was used.

   Please download and use s208_922 (Build Level 922) or later.

READ - Type 1

Supported as documented in the SELCOPY User Manual, with the exception of the following:

1. Optionally, 2 pseudo records may be generated for a column heading row and a 2nd row of underlining for the header. See HEADER parameter below.
2. USER and PASS keywords are supported.
3. NODUP, NULLS, PFX, VLEN, CHAR, keywords are silently ignored.
4. CHAR and SEP='|' are assumed by default.
5. SEP=NO|OFF or NOSEP may be coded to suppress the separator chars. SEP='x' may be used to override the default of '|'. e.g.

```
opt          ssn=L02ORCL    user=Joe   odbcpass="xxxxxx"
read ifil    table = user_objects        into 1001                      \
             fmat  ="timestamp, namespace, subobject_name, object_name" \
             where  "temporary='Y'"   header
```

READ - Type 2

Supported as documented in the SELCOPY User Manual, with the same exceptions as listed above for READ Type 1. e.g.

```
equ sqlsel   "select *   from user_objects   where temporary='Y'"
read ifil  sql=sqlsel  user=joe opass=xxxxxx  ssn=xx       into 1001
```

READ - Type 3

Supported as documented in the SELCOPY User Manual. e.g.

```
read ifil   into 1001      * Where ifil identifies an open ODBC cursor.
```

ODBC Operation

Supported as documented in the SELCOPY User Manual for the DB2 Operation, with the exception of the following:

1. For convenience, when an ODBC statement is coded as a literal, a default STOPAFT=1 is applied to it.

   If for any reason this is not required, the default may be overridden by coding an explicit STOPAFT parameter. e.g.

   ```
   odbc  "drop  table djh_test_oracle"   s=22
   ```

UPDATE of current row

Supported as documented in the SELCOPY User Manual, with the exception of the following:

1. Column values to be updated should be supplied in character format.
2. Problems can be encountered on an Oracle database where SQLstate=34000 is returned with:

   ```
   "SQLerror=[Oracle][ODBC]Cursor Not Updateable."
   ```

   As of 2009/03/14, no solution has been found, but updates are successful on a DB2 database. e.g.

   ```
   opt  w=2222        ssn=xx  user=xx  odbcpass="xx"
   equ  irec  1001
   read fnam1  tab=test_table   into irec   upd='COLOR, NOTES'
   pr  from irec
   if    pos irec+COLOR = 'Blue '       * Was irec+0.
    and  pos irec+NOTES = 'Trendy '     * Was irec+30.
     then pos irec+NOTES = "Modified. "
     then pos irec+COLOR = 'Turquoise '
     then UPD fnam1   from irec
     then pr          from irec
     then space 2
   ```

DELETE of current row

Supported as documented in the SELCOPY User Manual, with the exception of the following:

1. Problems can be encountered on an Oracle database where SQLstate=34000 is returned with:

   ```
   "SQLerror=[Oracle][ODBC]Cursor Not Updateable."
   ```

   As of 2009/03/14, no solution has been found, but deletes are successful on a DB2 database. e.g.

   ```
   opt  user=xx  odbcpass="xx"  ssn=xx   * UserId, Passwd, Sub-System Name (DataBase).
   equ %TABL%     djh_db2_test01         * Table name.
   equ   irec      101      * Input - I/O Area.
   opt          w 2200
   read fnam1  into irec  tab=%TABL%  upd      * The UPD keyword must be last.
   pr  fr irec
   if pos irec+0 = 'Black '
     then DEL fnam1
   ```

Notes:

1. UPD with no argument coded on the READ statement will be interpreted by SELCOPY to mean that all columns are updateable. "FOR UPDATE" will be generated on the SQL SELECT statement, without the " OF col1, col2, etc" being appended.

   But, because the UPD parameter has no column list argument, it must be coded last on the READ statement.

2. The READ statement could have been coded as:

```
read fnam1  into irec  sql="select *  from %TABL%   for update"
```

Here, it is essential to define the equated table name as %TABL%, rather than just TABL, because it is used within a quoted string.

Equated names that start and end in percent signs are treated as replacements for environment variables and as such will get substituted when used, whether within quoted strings or not.

Prepared INSERT

Supported as documented in the SELCOPY User Manual, with the exception of the following:

1. Problems can be encountered on an Oracle database where SQLstate=22008 is returned with:

```
"SQLerror=[Oracle][ODBC]Datetime field overflow."
```

It would appear that, regardless of the NLS_DATE_FORMAT setting, any DATE column to be inserted must be in ISO format. e.g.

```
SELCOPY/WNT 2.08 at Compute Bridgend - Wales - (pw=94)  -djh-      2010/02/11 17:09   PAGE   1
------------------------------------------------------------      -----------------   --------
         ** c:\djh\cc\slc\ctl\SSORA16 **    L=012 --- 2010/02/11 17:08:57  (L07)
                     * INSERT rows in ORACLE dbase - 2009/03/03 -djh-
         equ  wid    70                      * To fit line size printed.
         opt  dw=wid  pw=94   logsql='lst\SSORA16.log'
         opt  user=xx  odbcpass="xx"  ssn=xx   * UserId, Passwd, ODBC Sub-System Name.
         opt  noban  prtsum=3                  * No banner on term.  Enhanced summary.

         equ  %TABL%    djh_test_oracle
         equ  irec    1      * DBASE Input Area.
         equ  crec  201      * CARD  Input Area.
         equ  orec  401      *       Output Area for INSERT statement.
         opt  w    2200

                     * Drop and recreate ORACLE table to ensure empty.       **
                     *       Replaces =read1= loop to delete all rows.
     1.    line 1   s=1
     2.    odbc  "drop    table %TABL%"
     3.    odbc  "create  table %TABL%   \
             ( color      varchar2(10),  \
               date_sold  date,          \
               price      number,        \
               notes      varchar2(20) )"

         =populate=   * Loop to insert rows in table.                         **
          --------
     4.    space 2  s=1
     5.    pr   '=populate=  Loop to insert rows in table.'     s=1
     6.    read card into crec  fill

           if eof card
     7.      t close fnam3
     8.      t goto readv

           if p crec = '*'      * Ignore comment records.
     9.      t goto populate

    10.    do build_orec
    11.    INSERT fnam3  from orec  tab=%TABL%      * No FMT, will default to all columns.
    12.    goto populate

         =readv=      * Loop to verify all rows inserted.                     **
          -----
    13.    space 2  s=1
    14.    pr   '=readv=  * Loop to verify all rows inserted.'  s=1
    15.    read fnamv  tab=%TABL%   into irec    HDR

           if eof fnamv
    16.      t pr '=EOF='
    17.      t do cvxxtest
    18.      t eoj

    19.    pr  l=wid  fr irec    s=222
    20.    goto readv

         =build_orec=   * Sub-Rtn *                                           **
          ----------
               * Needed as a subrtn so that references to COLOR, DATE_SOLD, PRICE and NOTES occur
               * after the reference to INSERT fnam3 which defines the column positions.
    21.    pos orec+NOTES-1     = 20 at crec+60       * Len=20
    22.    pos orec+PRICE-1     = 20 at crec+40       * Len=17
    23.    pos orec+DATE_SOLD-1 = 20 at crec+20       * Len=19
    24.    pos orec+COLOR-1     = 20 at crec+00       * Len=10
    25.    pr  l=wid  s=222          fr orec
    26.  =ret=

         =cvxxtest=     * Sub-Rtn *  Check QFD lengths with CVxx and Arit.    **
          --------
    27.    p orec,orec+wid-1 = ' '      * Blank it out.         ....,....1
    28.    add  price+irec-1  to price+irec-1   into       orec+00  fmt='z,zz9.99'
```

```
    29.    cvch color+irec-1                    to 20 at orec+10  fmt='xxxx,xxxx xxxx,xxxx xxxx'
    30.    space 1
    31.    pr  l=wid  fr orec    s=222
    32.  =ret=
         end

SELCOPY/WNT 2.08 at Compute Bridgend - Wales - (pw=94)  -djh-      2010/02/11 17:09   PAGE   2
------------------------------------------------------------      ----------------   --------
    INPUT   SEL SEL                                                                    RECORD
    RECNO   TOT ID.          1         2         3         4         5         6         7  LENGTH
    -----   --- --- ....,....0....,....0....,....0....,....0....,....0....,....0....,....0  ------
        0     1   5 =populate=  Loop to insert rows in table.                              80
        2     1  25 Crimson    2009-08-07           5.55           Must be ISO date fmt      80
        3     2  25 Grey       2008-12-31         395.22           Only 1 sale               71
        4     3  25 Violet     2008-09-12          14.99           Sweet                     65
        5     4  25 Rose       2008-08-29           3.62           Gentle                    66


        5     1  14 =readv=  * Loop to verify all rows inserted.                           66
        1     1  19 COLOR     |DATE_SOLD (19)     |PRICE (17)       |NOTES (20)          |     70
        2     2  19 ----------|-------------------|-----------------|--------------------|     70
        3     3  19 Crimson   |2009-08-07 00:00:00|5.55             |Must be ISO date fmt|     70
        4     4  19 Grey      |2008-12-31 00:00:00|395.22           |Only 1 sale         |     70
        5     5  19 Violet    |2008-09-12 00:00:00|14.99            |Sweet               |     70
        6     6  19 Rose      |2008-08-29 00:00:00|3.62             |Gentle              |     70
        6     1  16 =EOF=                                                                 70

        6     1  31     7.24  526F,7365 2020,2020 2020                                   70
                    ....,....1....,....2....,....3....,....4....,....5....,....6....,....7

SUMMARY..
 SEL-ID      SELTOT            FILE     BLKSIZE  LRECL           FSIZE   CI    DSN
 ------      ------            ----     -------  -----           -----   --    ---
    1             1   line
    2             1   ODBC                                                    drop table DJH_TEST_ORACLE
    3             1   ODBC                                                    create table DJH_TEST_ORACLE ( co
                                                                              lor varchar2(10),  ...

                    =populate=                    * Loop to insert rows in table.                    **
    4             1     space
    5             1   pr
    6             5   READ     CARD       2048    80 U               5       C:\djh\cc\slc\ctl\ssora16
                      if
    7             1     t CLOS fnam3              70 F               0       DJH_TEST_ORACLE
    8             1     t goto readv
                      if                                              * Ignore comment records.
    9             1     t goto populate
   10             4   do   build_orec
   11             4   INS      fnam3              70 F               0       DJH_TEST_ORACLE
                                                                     * No FMT, will default to all columns.
   12             4   goto populate

                    =readv=                       * Loop to verify all rows inserted.                **
   13             1     space
   14             1   pr
   15             6   READ     FNAMV              70 F               6       DJH_TEST_ORACLE
                      if
   16             1     t pr
   17             1     t do   cvxxtest
   18             1     t eoj
   19             6   pr
   20             6   goto readv

                    =build_orec=                  * Sub-Rtn *                                        **
   21             4   mod                                             * Len=20
   22             4   mod                                             * Len=17
   23             4   mod                                             * Len=19
   24             4   mod                                             * Len=10
   25             4   pr
   26             4 =ret=

                    =cvxxtest=                    * Sub-Rtn *  Check QFD lengths with CVxx and Arit.    **
   27             1   mod                                             * Blank it out.         ....,....1
   28             1   add
   29             1   cvch
   30             1   space
   31             1   pr
   32             1 =ret=

    ** SELCOPY/WNT 2.08.922  Licensed by Compute (Bridgend) Ltd  +44 (1656) 652222 & 656466 **
                                ** Expiry: 01 Feb 2011 **
```

2008/09/26 s208_914

DIRTYPE for DIR and DIRDATA input

For Windows, DIR and DIRDATA input may use the DIRTYPE parameter to control the selection of directory entries based on the file Attributes reqd.

DIRTYPE may be coded on an OPTION statement, or on a READ statement qualified with either a DIR or DIRDATA parameter

The argument for DIRTYPE may be the keyword ALL, or any subset or combination of the string ".ADHNRSY" where the meanings are:

Code	File Attribute required
.	The "." and ".." entries if present.
R	Read-only.
H	Hidden.
S	System.
V	Volume-Id.
D	Subdirectory.
A	Archive. (Amended file.)
Y	Symbolic Link. (UNIX only.)
N	Normal. (No attribute flags set.)
ALL	All directory records, regardless of type.

Blanks and commas are treated as punctuation and ignored, provided the DIRTYPE argument is enclosed in quotes.

TYPEDIR is a synonym for DIRTYPE.

On a READ statement for a DIR or DIRDATA file, the keyword, TYPE, is also treated as a synonym for DIRTYPE. e.g.

```
OPTION  DIRTYPE = 'N, R, A"      * This is the default.

READ  "%SLC%/tst/*" DIR DIRTYPE=A           * Amended files only. No subdirs.
READ  "c:/abc/*"    DIR TYPE=.ADHNRSY SUB=6 * Spec'd codes. Nest 6 levels of subdir.
READ  "c:/abc/*"    DIR TYPE=ALL      SUB   * All codes, anything, all subdirs.
```

2008/09/26 s208_914

RAW parameter for DIR and DIRDATA input

When reading a directory, if a DIR record is read that has an unrecognized DIRTYPE, it may be useful to display the actual contents of the physical directory entry without having it interpreted and formatted by SELCOPY.

The RAW parameter on the READ statement for the directory allows this. e.g.

```
READ  "%SLC%/tst/*"  DIR  DIRTYPE=ALL  RAW  * No formatting.
PRINT   TYPE=D   STOPAFT=4                   * Print in dump format.
```

2009/01/22 s208_917

Pointer and Keyword Arithmetic

The following are now handled as Keyword AT pointers, as already documented in the "Keyword Changes" section (IMPORTANT) above:

Keyword AT pointers	The POS in storage
DIFF	UXADIFF
RETCD,RETCODE,RC	UXRETCD
REASCD	UXREASCD
RETVSAM	UXRETVSAM
RETCMS,CMSRETCD,RETXV,RETSYS	UXRETSYS

Arithmetic operations may now be performed on both AT pointers and Keyword pointers. e.g.

```
** c:\djh\cc\slc\ctl\SSAT29 **     L=009 --- 2009/03/28 11:57:16  (L07)
             * @ ptr as destination of an arith statement - 2008/12/05
opt w 222  dw=30  noban  * prtsum=3
equ sp1   space 1
  @3 = 3   !@p = 2  !lrecl = 26  !line 1
  p 1 = 'abcdefghijklmnopqrstuvwxyz7890'!pr !pr ' Alphabet L=26.'           !sp1
  diff = 26             !pr fr diff         !pr ' Should show "z7890".'     !sp1
  add 22  to retcd      !pr fr rc           !pr ' Should show "vwxyz7890".' !sp1
  mult @p by 3          !p @p = '6'     !pr !pr ' Pos 6 should be 6.'       !sp1
  add @3  to @p         !p @p = '9'     !pr !pr ' Pos 9 should be 9.'       !sp1
  add 1   to L                          !pr !pr ' Lrecl should be 27.'      !sp1
  sub 5   fr @p         !p @p = '4'     !pr !pr ' Pos 4 should be 4.'       !sp1
  add 3   to L                          !pr !pr ' Lrecl should be 30.'      !sp1
  add 5   to 5  into @j !p @j = '#'     !pr !pr ' Pos10 should be #.'       !sp1
  add 1   to @j into @j !p @j = '+'     !pr !pr ' Pos11 should be +.'       !sp1
```

2009/01/26 s208_917

Error messages for IF-type statements

For an IF-type statement, the selection id reported for an error message is that of the next THEN statement.

To assist in identifying that the error concerns an IF, AND or OR statement, the selection id is reported with only 1 minus sign.

For example, assuming an error on an IF statement preceding a THEN statement which is Selid 42, the following messages are reported:

```
(SEL  -42)   **# SELECTION TIME ERROR 523 **#   F=john     - FILENAME NOT FOUND OR CONFLICT
**#WARNING**# (SEL  -42)       52 = RETURN CODE FROM SELCOPY
```

Previous releases reported:

```
(SEL---42)   **# SELECTION TIME ERROR 523 **#   F=john     - FILENAME NOT FOUND OR CONFLICT
**#WARNING**# (SEL---42)       52 = RETURN CODE FROM SELCOPY
```

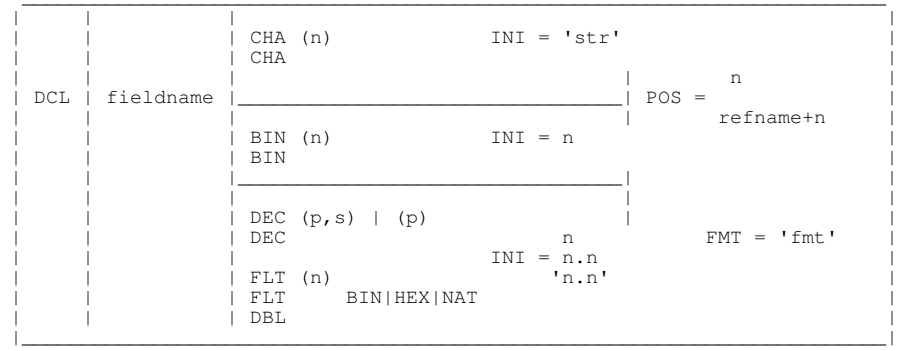
2009/03/28 s208_917

DCL statement

The DCL statement, synonym DECLARE, has been introduced for declaring fields within a dynamic area of storage having a position, a length and a data type, without a requirement to use WORKLEN to specify a workarea.

If all working fields are declared with the DCL statement, no WORKLEN is required.

WORKLEN may however also be used in conjuction with DCL statements to specify a traditional workarea for use as before. The WORKLEN area will be allocated in a separate dynamic storage area and DCL fields may be declared within it.



fieldname

The name for a DCL field must start with an alpha character and consist of alphanumeric characters.

Data types	Keyword	Synonyms	Default Precision	Default Size	Max
Character	CHA	C,CHAR	(1)	1	any
Binary	BIN	B	(4)	4	8
Packed Decimal	DEC	D	(5,0)	3	16
Floating Point	FLT	F,FP,FLOAT	(4)	4	8
Floating Point	DBL	DOUBLE	(8)	8	8

FLOAT and DOUBLE are treated as synonymous apart from when the precision is omitted resulting in the default precision of 4 and 8 respectively.

Field width, Precision and Scale

CHA, BIN and FLT fields optionally have a precision, (n), following the data type, where n is the field width in bytes. The brackets are required.

DEC fields optionally have a precision and scale, (p,s), following the data type, where the first number, p, is the precision which defines the number of decimal digits to be allocated for the field, including the digits after the decimal point, and where the second number, s, is the scale which defines the number of digits to be treated as after the decimal point. The brackets are required. The comma may be replaced with blank(s).

If scale, s, is omitted, a scale of 0 is used and p will then define the maximum number of digits for a decimal integer.

DEC fields are stored with 2 decimal digits held in each byte, with the junior half of the junior byte reserved for the sign. Thus the field width in bytes will be: (precision/2 +1)

Style of Floating Point representation

HEX or HFP	IBM Base 16 Hexadecimal style.
BIN or BFP	IEEE-754 Base 2 Binary style.
NAT or NATIVE	The native style of the host machine, regardless of any user or installation setting of the default.

If omitted, NATIVE floating point style is used.

INI parameter

By default, CHAR fields are initialized to blank and arithmetic fields to zero. The optional INI parameter, synonym INIT, may be used to override this.

For CHAR fields, the INIT argument must be enclosed in single or double quotes.

For arithmetic fields, a decimal point (dot, period or fullstop) in the INI argument is significant and is respected fully for a FLOAT field and matched according to the scale of a DEC field. Only 1 dot for the decimal point is permitted.

For arithmetic fields, the argument may be enclosed in quotes, but quotes are only required when commas or spaces are used for punctuation. The dot for the decimal point will still be respected.

POS parameter

POS=n may be coded to indicate that the DCL statement is to use storage within the traditional WORKLEN area at POS=n.

POS=refname may be coded to indicate that the DCL statement is to redefine (overlay) storage already allocated, starting at the storage for a previously coded DCL statement for the field name, refname.

POS=refname+n will give an offset of n on the referenced DCL storage. The offset may be negative, but n must be numeric.

FMT parameter

The FMT parameter, synonyms FORMAT and FMAT, on a DCL statement may be used to define the character format of an arithmetic field.

FMT on CHA fields:

If the DCL field is CHA, FMT defines the field width and the location of the decimal point defines the scale. The (n) may therefore be omitted. If (n) is coded, it must match the width of the FMT argument.

FMT on a CHA field allows a formatted assignment to be made.

FMT on Arith fields:

Arithmetic DCL fields may have a FMT parameter coded to define the formatting required when the field is the subject of a PRINT statement. The FMT used does not need to match the defined scale. When printing, if FMT is not defined on the DCL or on the PRINT statement, a default format of FMT='SS,SSS,SSS,SS9' is used for integer fields, or FMT='SS,SSS,SSS,SS9.9999' for any field with a scale. e.g.

```
DCL  ABC  CHA        INI  1234.56      FMT='ss,ss9.999'
DCL  PQR  DEC(8,2)   INI  654321.78    FMT='sss,sss,ss9.999'
PRINT    "PQR value is:"   PQR       "   ABC=" ABC   "   Ok?"
```

would give the print line:

```
....,....1....,....2....,....3....,....4....,....5....,....6
PQR value is:   +654,321.780   ABC=+1234.560   Ok?
```

Examples

```
DCL   irec   CHA  (100)              * Field in DYN area.  (No POS coded.)
DCL   rec2   cha  ( 50)              * Field in DYN area.  (No POS coded.)
DCL   ABC    Char  (10)   POS 1001   * Field at POS 1001 in WORKLEN area.
DCL   totc   CHAR FMAT='zz,zz9.99'   * Field length omitted.

dcl   tot1   DEC                     * Default: (5,0) needing 3 bytes.
dcl   tot2   dec   (5)               * Same as (5,0) needing 3 bytes.
dcl   tot3   dec   (9,3)             * 5 bytes, with 3 places of decimal.
dcl   tot4   d     (7,2)             * 4 bytes, with 2 places of decimal.
dcl   tot5   d    ( 31, 6 )          * 16 bytes (max) with 6 places of decimal.

dcl   totb1  bin                     * Default: (4) needing 4 bytes.
dcl   totb2  bin   (6)               * 6-byte binary field.
dcl   totb3  bin   (8)               * 8-byte (max) binary field.

dcl   totf1  float                   * Default: (4) needing 4 bytes.
dcl   totf2  flt   (4)               * Same as default FLOAT.
dcl   totf3  double                  * Default: (8) needing 8 bytes.
dcl   totf4  float (8)               * Same as default DOUBLE.
dcl   totf5  double(4)               * Same as default FLOAT.


dcl   fld1   DEC   (19,6)            * Field in DYN area. (10 bytes.)
DCL   fld2   CHA   (90)              * Field in DYN area.
dcl   irec   char  (100)  pos=fld1   * Field in DYN area overlaying fld1 and fld2.
```

Usage Examples

```
         read abcfil  into irec   * ERROR 546 if input rec exceeds size of DCL var, irec.
                                  * Prev rec residue not cleared unless FILL coded.
         irec = 'ABC'             * Pos 4 up to end of irec will be blank padded.
         irec+20 = 'ABC'          * Pos 1 to 20 of irec will be unchanged.
                                  * Pos 21 to 23 of irec will be set to 'ABC'.
                                  * Pos 24 up to end of irec will be padded with
                                  * the FILL char, which by default is blank.

         rec2 = irec              * Contents of irec will be copied to rec2, truncated to
                                  * fit rec2, with no error given.
         irec = rec2              * Contents of rec2 will be copied to rec2 and the
                                  * remainder of irec padded with the FILL char.

         if irec = rec2           * The shorter field is padded with the FILL char.

         tot5 = totf4             * Arithmetic assignment follows normal conversion
                                  * rules.  RC=8 is possible if too large for destination.

         if tot5 = totf4          * Arith comparison follows normal conversion rules.

         @xyz = irec              * Sets @xyz to point at the 1st byte of irec.
         @xyz = irec+20           * Sets @xyz to point at pos 21 of irec.

         @abc = tot1              * Sets @abc to the VALUE held in the arithmetic
                                  * field, tot1.  It does NOT point to it.

         add tot1 to totb1   into totf5    * Sets totf5 to the value tot1 + totb1.
         totf5 = tot1 + totb1              * Does the same thing.
         add tot1 to totb1   into @tot     * Sets @tot  to the value tot1 + totb1.
         @tot = tot1 + totb1               * Does the same thing.
                                  * RC=8 is possible if too large for an @ ptr value
                                  * which is held as a 4 byte binary number.

         if irec = 'x'  ptr=@x    * Scans irec for 'x' and sets the pointer @x to
                                  * the 1st 'x' found. If no 'x' found, @x is set to NULL.
                 * Not suitable for arithmetic DCL vars, but if really necessary, use
                 * POS=refname on a DCL for a CHA var overlaying the same storage, or
                 * use &DCLvar to refer to the address of DCLvar.  See &DCLvar below.
```

2009/06/19 s208_918

CASEI for Case Insensitive Compare

e.g.

```
** c:\djh\cc\slc\ctl\ssif26 **     L=006 --- 2009/07/16 16:42:17  (L07)
            * IF statement with CASEI for Case Insensitive string compare - 2009/06/17
opt noban dw 60  prtsum=3   w 222
  lrecl = 60
       * ....,....1....,....2....,....3....,....4....,....5....,....6
  p 1 = "abcdefghi j(lmnopqr st)vwxyz  ABcdefghi"
  line 1
  print

  if p 01   =  ABCD   casei         !t pr ' EQ  - Correct'         !el pr 'Error: abcd <> ABCD'
  if p 11   = 'J(LM'  casei         !t pr ' EQ  - Correct'         !el pr 'Error: j(lm <> J(LM'
  if p 21   = 'ST)V'  casei         !t pr ' EQ  - Correct'         !el pr 'Error: st)v <> ST)V'
  if p 1,26 = 'ST)V'  casei         !t pr from @,L                 !el pr 'Error: ST)V not in range.'
  if p 1,26 = 'ST)V'  casei step=10 !t pr from @,L                 !el pr 'Error: ST)V not in step.'
  if 'AbCd' = 'Abcd'                !t pr 'Error: Should be diff.' !el pr 'Diff - Correct'
  if 'AbCd' = 'Abcd'  casei         !t pr ' EQ  - Correct'         !el pr 'Error: Should be same.'
  if 4 at 1 = 3 at 31 casei         !t pr 'Error: Should be diff.' !el pr 'Diff - Correct'
  pr from DIFF
  pr 'Above should print from "defghi j(lm" ... etc ...'

e **
```

2009/07/31 s208_918

POS DATE information extended

The following offsets are subject to change until formally released in the SELCOPY User Manual.

POS DATE-36	4 Binary.	No of Seconds since 1970/01/01 00:00:00
POS DATE-32	4 Binary.	No of Millisecs after the second.
POS DATE+68	4 Binary.	No of Microsecs after the second.
POS DATE+72	4 Binary.	Elapsed Millisecs since start of program.

The tenths of seconds value at POS DATE+18 on previous PC and UNIX releases has always been 0. It is now set using the value from an appropriate system call.

However, on certain systems, the value has been found to be unreliable. Please therefore avoid depending on it.

For timing exercises, it is better to use the Elapsed Milliseconds since the start of the program in the 4-byte binary field at POS DATE+72.

2009/08/04 s208_918

Numeric literals allowed in quotes with punctuation

For convenience and clarity, the punctuation characters, blank and comma only, are allowed in an arithmetic literal, provided the literal is enclosed in single or double quotes. e.g.

```
add '1,234,567'  to 4 at 2001 type=b   * Is the same as:

add  1234567     to 4 at 2001 type=b
```

2009/08/07 s208_918

SEP char in comment respected even if in quotes

The risk of an undetected misinterpretation of control statement syntax has been removed by respecting the SEP char within comment data, regardless of quotes. e.g.

```
if p 1 = 'x'
or p 1 = 'y'
  t print 'ok'
el  print 'err'  * What's this ?   !t goto get
```

Previous releases treated the SEP char (!) as part of the comment, because the quote preceding it masked the SEP char. No warning was given and the GOTO GET statement was ignored.

Note that the statement:

```
print 'What a farce !'
```

remains valid. The SEP char is not part of a comment.

However,

```
print 'Oh dear.'  * 'What a farce!'
```

will be invalid because the SEP char in the comment will be obeyed, leaving the residue, just a single quote, as the next statement, which will cause an error due to an unmatched quote.

To avoid the problem altogether, code

```
opt sep=off       * Disable SEP char processing, or use a diff char.
print 'Oh dear.'  * 'What a farce!'
opt sep='!'       * Reset SEP char processing as reqd.
```

2009/09/03 s208_918

Multiple fields on a PRINT statement

The PRINT, LOG and PLOG statements have been enhanced to support multiple data arguments on a single statement.

Thus, in combination with the DCL statement, a line of output may consist of numerous DCL fields interspersed with literals.

Literals and fields are concatenated to form a single output line, with no additional spacing between them. If spacing is required, it must be provided within the literal values, in the FMT argument, or as an additional literal. e.g.

```
DCL  ABC  DEC(8,2)   INI  654321.78    FMT='sss,sss,ss9.999'
DCL  XYZ  CHA        INI  1234.56      FMT='ss,ss9.999'
PRINT    "ABC value is:"   ABC       "   XYZ=" XYZ    "   Ok?"
```

would give the print line:

```
....,....1....,....2....,....3....,....4....,....5....,....6
ABC value is:   +654,321.780   XYZ=+1,234.560   Ok?
```

A different FMT parameter and argument may be coded on the PRINT statement in order to override that coded on the DCL statement.

The \ character below means that the statement is continued on the next line. e.g.

```
DCL  ABC  DEC(8,2)   INI  654321.78    FMT='sss,sss,ss9.999'
DCL  XYZ  CHA        INI  1234.56      FMT='ss,ss9.999'
PRINT    "ABC value is:"   ABC FMT='s99,999,999.99 '           \
         "   XYZ="          XYZ FMT='99,999.990'       "   Ok?"
```

would give the print line:

```
....,....1....,....2....,....3....,....4....,....5....,....6
ABC value is:+00,654,321.78    XYZ=01,234.560   Ok?
```

2009/09/17 s208_919

Multiple fields on a PRINT statement with hex FMAT

With the advent of multiple fields on a PRINT statement, it soon became desirable to print a field in hex, rather than interpreted in numeric form.

Formatting an arithmetic field in hex is now supported. SELCOPY will differentiate between decimal numeric and hex notation according to the content of the FMAT argument. e.g.

```
dcl DEC72           DEC(7,2)    INI=76543.21
pr       'DEC72 var in decimal    = '    DEC72  FMT='s99,999.99'
pr       "DEC72 var in hex        = X'"  DEC72  FMT='xxxx,'   "'"
```

would give the print lines:

```
....,....1....,....2....,....3....,....4
DEC72 var in decimal    = +76,543.21
DEC72 var in hex        = X'7654,321C'
```

Note that the FMAT arg for the hex print is too short for the size of the source field, DEC72. For conversion to a hex string only, when the format is exhausted, it wraps and continues at the beginning again.

Thus, for a larger field where FMT='xxxx,xxxx,xxxx,xxxx,xxxx' is required, it is sufficient to code FMT='xxxx,' to achieve the same result.

Spaces are also tolerated, so FMT='xxxx,xxxx ' might be useful for giving better readability.

2009/09/22 s208_919

HEX offsets supported

Hex offsets are now supported. SELCOPY will recognize a hex number by its prefix of 0x or 0X, having borrowed the syntax from UNIX. e.g.

```
pos 0x385  = '<--- Pos 901. (0x385)'
pos 0x3B1  = '<--- Pos 945. (0x3B1)'
pr  fr 0x0385   len=0x003F+3           * From 901, Length 66.
```

would give the print line:

```
....,....1....,....2....,....3....,....4....,....5....,....6....,.
<--- Pos 901. (0x385)                      <--- Pos 945. (0x3B1)
```

The syntax X'xxxx' has not been supported as a number for use as an offset or length because it has always been treated as a literal in selcopy's existing syntax. e.g.

```
pr  fr x'C1C2C3'        * On an EBCDIC machine.
pr  fr x'414243'        * On an ASCII  machine.
```

would give the print line:

```
....,....1....,....2....,....3....,....4....,....5....,....6....,.
ABC
```

2009/10/07 s208_919

MVS DD SELCNAM and SELCMSG supported

Checking of the DD names, SELCNAM and SELCMSG, has been introduced for the MVS version of SELCOPY 2.08.

If a DD statement for SELCNAM exists, it will be used instead of the DSN information held in CBLNAME.

SELCNAM may be allocated to DUMMY, in which case the DSN information held in CBLNAME for the SELCOPY.NAM file will be ignored.

If a DD statement for SELCMSG exists, it will be used instead of the DSN information held in CBLNAME.

If no DD statement for SELCMSG exists, the DSN information held in CBLNAME for the hlq.SELCOPY.NAM file will be used instead, with the last 3 characters, "NAM", modified to "MSG".

If no hlq.SELCOPY.NAM file is defined in CBLNAME, or if no CBLNAME load module exists, the DSN "SELCOPY.xxx" or "userid.SELCOPY.xxx" will be used, where the "xxx" is either "NAM" or "MSG" as appropriate.

2009/11/06 s208_920

MVS Option to specify alternate CBLNAME

The command line option "-NAM=altnam" has been introduced for the MVS version of SELCOPY 2.08, allowing the MVS user to specify an alternate load module to replace the default of CBLNAME.

The load module, CBLNAME, is still loaded first, if it exists, in order to establish the user's chosen SEP char if different from the default of "!" (Exclamation Mark).

Then, if the option -NAM=xxx is coded in the MVS parm string, the alternate load module, xxx, is loaded and used as a replacement for the CBLNAME already in storage. e.g.

For TSO:

```
call 'your.loadlib(slc)' '/-NAM=CBLNAME2 -ctl=your.input.ctl -lst=your.selc.lst' asis
```

where:

/	Indicates to Language Environment the end of run-time options in the parm field. (There aren't any here.)
-nam=	Indicates to SELCOPY the load module to replace CBLNAME.
-ctl=	Indicates to SELCOPY the Control Card input file to replace SYSIN.
-lst=	Indicates to SELCOPY the SYSPRINT dsn to be written.
asis	Preventsuppercasing of the parm string. Not necessary here as there are no control statements following the options. The options themselves are case insensitive.

For Batch:

```
// EXEC PGM=SLC,PARM='/-nam=cblname2 -ctl=your.input.ctl -lst=your.selc.lst'
```

Note:

1. If options are used to specify -ctl or -lst, any ALLOC or DD statement for SYSIN or SYSPRINT respectively is ignored.

2009/11/06 s208_920

MVS Operator Message switches in CBLNAME obeyed

The following CBLNAME switches are for controlling messages to the operator or to the TSO terminal:

CBLSMSG at **CBLNAME+X'55'** - Messages to Operator.		
CBLSMNC	0x01	ON = Suppress Control Card error message.
CBLSMNS	0x02	ON = Suppress Select Time error message.
CBLSBAN	0x04	ON = Log the Banner line at startup.

For s208_920/MVS, these switches are now obeyed.

Default:

By default, the CBLSMSG byte in CBLNAME is set to X'00', resulting in no banner info logged for normal error free execution. However, a Control Card or Select Time error will issue a message to the log file together with the banner line giving timestamp info.

Alternative:

SELCOPY's banner line may be suppressed or enabled with an "OPTION" statement in your selcopy.nam file, or in your SELCOPY control statements.

```
OPTION   NOBANNER|NOBAN|BANNER|BAN     * BANNER or not.
```

2010/01/24 s208_921

MVS SELCOPY/i LIST Window as an input file

Any LIST window, as currently available on the Interactive version of SELCOPY, SELCOPY/i, may now be read as an input file by SELCOPY under batch or TSO.

Syntax:

```
READ  [fname]  LIST  = 'any s/i list window command'   \
               [SEL   = 'reqd column names']            \
               [WHERE = 'any s/i filter command']       \
               [SORT  = 'any s/i column name']       [HEADER]
```

Batch Example:

```
// EXEC PGM=SLC,PARM='/ !read LIST = 'LD CBL.VVC.**'  !pr l=80 dw=80 s=5  !e'
```

or

```
// EXEC PGM=SLC
//SYSIN DD *
 read LIST = 'LD CBL.VVC.**'
 pr l=80 dw=80  s=5
```

Would give the SYSPRINT output:

```
INPUT   SEL SEL                                                                    RECORD
RECNO   TOT ID.          1         2         3         4         5         6         7         8  LENGTH
-----   --- --- ....,....0...,....0...,....0...,....0...,....0...,....0...,....0...,....0...,....0  ------
    1     1   2 USERCAT.CBLCAT                                 0                        0     0    328
    2     2   2 CBL.VVC.CTL                                    1 CBLM01 A PO  FB       80 23440    328
    3     3   2 CBL.VVC.CTL.PDS                                1 CBLM04 A PO  FB       80 23440    328
    4     4   2 CBL.VVC.DEF.CBLV                               1 CBLM06 A PO  FB       80 23440    328
    5     5   2 CBL.VVC.DEF.VMST                               1 CBLM04 A PO  FB       80 23440    328
                ....,....1...,....2....,....3...,....4....,....5....,....6....,....7....,....8
```

The / in the parm field indicates to LangEnv the end of run-time options. No run-time options for Lang Env were used here, and the parm string does not have a / within it, so the / could have been omitted.

Because SYSPRINT has no DD statement, the file 'user.SELC.LST' will be created or overwritten with RECFM=VB,LRECL=1024 as default, where user is your RACF userid.

TSO Example:

```
slc   / !read LIST='LD CBL.VVC.**'  !pr l=80 dw=80 s=5  !e
```

or

```
call 'your.loadlib(slc)'  '!read LIST="LD CBL.VVC.**"  !pr l=80 dw=80 s=5  !e' asis
```

Would give the same SYSPRINT output as shown above for batch.

As for batch, the / in the parm indicates to LangEnv the end of run-time options.

If SYSPRINT has not been allocated, the file 'user.SELC.LST' will be created or overwritten with RECFM=VB,LRECL=1024 as default, where user is your TSO userid.

When SLC is invoked directly as a program, everything following the program name is treated as the parm field and passed unmodified to SLC.

When SLC is invoked from a TSO CALL statement, both the fully qualified program name and the parm field must be enclosed in single quotes. Hence double quotes are used to delimit SLC's LIST argument, although 2 consecutive single quotes could have been used instead. Omitting the quotes on the program name would result in the user's TSO userid being prefixed to the name.

The asis argument on the CALL statement will prevent uppercasing of the parm string which is passed to SLC. Could be omitted here as there are no case sensitive control statements.

HEADER parameter for MVS LIST input

HD, HDR and HEAD are synonyms for HEADER on a READ LIST='.. ..' statement.

Use of the HEADER parameter on a READ statement for a LIST will cause SLC to generate pseudo records 1 and 2, which will provide the user with:

1. A heading line consisting of the selected column names, each blank padded to the full length of the column and terminated with a blank.
2. Underlining for the heading line using dots, commas and numerics to provide a scale for the full data width of each column.

By default, the 2 header records are not generated. e.g.

```
 1.  read LIST 'LL CBL.VVC.CTL'  HDR   where 'cursize>36'
 2.  pr     l=64 dw=64

INPUT   SEL SEL                                                                 RECORD
RECNO   TOT ID.          1         2         3         4         5         6      LENGTH
-----   --- --- ....,....0....,....0....,....0....,....0....,....0....,....0....  ------
    1     1   2 Member   Alias VV MM Created    LastMod          CurSize IniSize     87
    2     2   2 ....,... .     .. .. ....,....1 ....,....1....,.  ....,.  ....,.     87
    3     3   2 BBIREP01 N      1  6 1997/03/26 2003/03/24 16:59      38      38     87
    4     4   2 BBPWTEST N      1  9 2008/04/03 2008/04/03 11:17      38      34     87
    5     5   2 BIG      N      1  1 2003/05/30 2004/01/16 13:01    4004    4005     87
    6     6   2 IKEY01   N      1  4 2000/12/31 2004/01/15 18:11     136     140     87
    7     7   2 IKEY03   N      1  2 1998/05/19 2003/03/26 14:31     113     110     87
                ....,....1....,....2....,....3....,....4....,....5....,....6....
SUMMARY..
 SEL-ID      SELTOT      FILE     BLKSIZE  LRECL           FSIZE   CI    DSN
 ------      ------      ----     -------  -----           -----   --    ---
    1             7 READ DEFAULTF   2048       U               7       LL CBL.VVC.CTL
    2             7
```

POS FHDR for MVS LIST input

In the above example, the Selection Total is shown as 7, rather than 5, so the use of the HEADER parameter for LIST input can be misleading.

POS FHDR is an alternative for obtaining the column names. e.g.

```
 1.  read LIST 'LL CBL.VVC.CTL'  HDR   where 'cursize>36'
 2.  pr  from FHDR        s=1  * Header.
 3.  pr  from FHDR+LRECL  s=1  * The Scale line used for Underlining.
 4.  pr     l=64 dw=64
```

Then, the Selection Total would be shown as 5, the true value.

Reference to Column data as a field

If an MVS LIST window has been read as a file into the default position, POS=1, then the column names returned by SLCLIST in the heading line may be used as if they had been declared with a DCL statement having a fixed position, a length and a data type.

If read using the INTO=n parameter, then reference to each field must use an offset on the field name of +n-1.

Existing SELCOPY keywords, such as LRECL and DATE, which are returned by SLCLIST as column names, may not be used. If used, the original SELCOPY keyword meaning will take preference, as illustrated in the example below.

Numeric fields, such as DsnPcu and Blksz, are treated as Zoned Decimal, TYPE=Z. All other fields are Character, TYPE=C.

Data movement, comparison and arithmetic statements may be used in the same way as for a DCL variable. e.g.

```
        ** c:\djh\cc\slc\ctl\SSLL05 **     L=004 --- 2010/01/28 21:42:52  (L07)
        equ wid  80
        opt noban  pw=wid+30  dw=wid   w=2222

        dcl blkfact  flt   * Work field for calculating Blocking factor.

     1.    read  LIST  'LD   CBL.VVC.DEF.**' \
              sel   'dsnpcu pri alu sec nxt   org blksz lrecl recfm entry'  * Reqd columns.

           if in 1
     2.      t pr fr fhdr     l=wid         * Heading line with column names.
     3.      t pr fr fhdr+L   l=wid         * Scale line.

           if alu = ' '
     4.                           t gg    * Don't want catalogs.
           if   DsnPcu > 30
     5.                           t gg    * Acceptable.  Ignore these.

     6.    pr  l=wid  fr 1     s=22         * Over-allocated.
     7.    div blksz   by lrecl              into blkfact   * Wrong. Uses Selc's LRECL value.
     8.    pr  '         Blocking Factor ='    blkfact fmat=zzz9.99  '  Wrong.'    s=1

     9.    div blksz   by 6 at blksz+6 ty=z  into blkfact   * Need the LRECL value from the LIST.
                               * Gives:  **01 RETCD=8**  due to zero divide on rec 9.
    10.    pr  '         Blocking Factor ='    blkfact fmat=zzz9.99  '  Correct.'  s=1

        e **
```

```
    INPUT   SEL SEL                                                                                  RECORD
    RECNO   TOT ID.          1         2         3         4         5         6         7         8  LENGTH
    -----   --- --- ....,....0...,....0...,....0...,....0...,....0...,....0...,....0...,....0...,....0  ------
        1     1   2 DsnPcu   Pri Alu   Sec Nxt Org Blksz Lrecl RecFm Entry                               93
        1     1   3    ... ....., .   ....., ... ... ....., ....., ....., ....,....1...,....2....,....3.     93
        3     1   6     20     1 C       9   1 PO  23440    80 FB    CBL.VVC.DEF.VMST                    93
        3     1   8          Blocking Factor = 252.04  Wrong.                                            93
        3     1  10          Blocking Factor = 293.00  Correct.                                          93
        4     2   6     20     1 C       9   1 PO  23440    80 FB    CBL.VVC.DEF.V20A                    93
        7     3   6     20     1 C      10   1 PO  23440    80 FB    CBL.VVC.DEF.V200                    93
        9     4   6     20     1 C      10   1 PO      0     0       CBL.VVC.DEF.V212                    93
       10     5   6     20     1 C      10   1 PO  23440    80 FB    CBL.VVC.DEF.V22B                    93
       11     6   6     20     1 C      10   1 PO  23440    80 FB    CBL.VVC.DEF.V97N                    93
       12     7   6     20     1 C      10   1 PO  23440    80 FB    CBL.VVC.DEF.V98N                    93
       13     8   6     20     1 C      10   1 PO  23440    80 FB    CBL.VVC.DEF.V980                    93
       14     9   6      6     1 C       2   1 PO  23440    80 FB    CBL.VVC.DEF.V980Z07                 93
       15    10   6      6     1 C       2   1 PO  23440    80 FB    CBL.VVC.DEF.V980Z08                 93
       16    11   6      6     1 C       2   1 PO  23440    80 FB    CBL.VVC.DEF.V980Z09                 93
       17    12   6     20     1 C      10   1 PO  23440    80 FB    CBL.VVC.DEF.V99A                    93
                  ....,....1...,....2...,....3...,....4...,....5...,....6...,....7...,....8

SUMMARY..
 SEL-ID      SELTOT      FILE     BLKSIZE  LRECL           FSIZE   CI    DSN
 ------      ------      ----     -------  -----           -----   --    ---
    1            18 READ DEFAULTF   2048       U              18       LD   CBL.VVC.DEF.**
    2----3        1
    4             2
    5             4
    6----7       12
    8             1
    9            12      (** 01 RETCD=8 **)
   10             1

** WARNING ** (SEL----9)        8 = RETURN CODE FROM SELCOPY

    ** SELCOPY/MVS 2.08.921  Licensed by Compute (Bridgend) Ltd  +44 (1656) 652222 & 656466 **
                                  ** Expiry: 20 Jul 2010 **
```

2010/05/04 s208_922

MVS: UPD,DEL,INS totals in summary

For ease of reference, the update, delete and insert totals are now reported in the summary along with the summary entry for the READ statement. e.g.

```
SUMMARY..
 SEL-ID      SELTOT      FILE     BLKSIZE  LRECL           FSIZE   CI    DSN
 ------      ------      ----     -------  -----           -----   --    ---
    1----2        1
    3            12
    4            11 READ TESTK       100    71 U              11       DJH.TEST.KSDS
                                                                2 UPD
                                                                1 DEL
                                                                3 INS
                                  *EOF*NOT*REACHED*
```

Unlike the BAL version, the FSIZE column will reflect the number of records read, rather than the total no of records in the file, which can be misleading if VSAM Keyed Reads were used. This will be corrected at a later date, as well as supporting the BWD and FWD parameters for VSAM input.

2010/05/10 s208_922

MVS: Use SLC's SYSTEM command to issue a TSO command

e.g.

```
sys   "delete 'userid.abc.test.file' "                  s=1
tso   "alloc  shr reuse f(abc) dsn('userid.mast.file')"  s=1
```

TSO and SYS are synonyms for SYSTEM.

You may even use SLC's SYSTEM cmd to invoke another instance of SLC itself. However, care should be taken to use a command line directive to put the listing file to a different DSN.

```
tso   "slc / -lst=selc.lst2.tmp  !read abc !print s=4 !end"  s=1
```

2010/06/13 s208_922

MVS: Treat %ABC% as a rexx variable

REXX variables may be referenced in the SLC control statements by enclosing the variable name in percent signs.

Substitution of the reference will be made, even if within a quoted literal string.

2010/06/17 s208_922

MVS: DIR input off HFS.

Directory entries may be read off a mounted Hierachical File System using the DIR parameter on a READ statement.

Similarly, directory entries on the Byte File System (BFS) for CMS may also be read.

2010/07/06 s208_922

OPTION TRAP introduced as synonym for OPTION ABTRAP

The keyword, TRAP, is now recognized on an OPTION statement as a synonym for the original keyword, ABTRAP, meaning trap any abnormal end situation and terminate the job quietly, suppressing the default system action of producing diagnostic information.

Also, the keyword, NOTRAP, is recognized on an OPTION statement as a synonym for ABTRAP=OFF.

See also "*Command line option -NOTRAP*"

2010/10/06 s208_923

DCL var assignment from @ptr.

The value held in an @ pointer may be assigned to an arithmetic DCL variable and vice versa.

Also, an @ pointer may be used as the source for a PRINT statement, with an optional FORMAT parameter, in the same way as for an arithmetic DCL variable. e.g.

```
SELCOPY/WNT 3.00 at Compute Bridgend - Wales - (pw=94)  -djh-      2011/05/16 15:39   PAGE   1
------------------------------------------------------------      ----------------   --------

        ** c:\djh\cc\slc\ctl\SSPR02 ***    L=006 --- 2011/05/16 15:39:35  (L07)
                     * Print the value of an @ ptr - 2010/10/02
        opt  datawidth=60
        dcl  binvar  bin(4)  ini 123     fmt='+9,999'
        *            bin(4)  ini 456.789   * ERR 202 - DCL INIT VALUE FOR BIN MUST BE INTEGER ONLY
        dcl  binv2   bin(4)  ini 456       * INI=456.0 would also fail with ERR 202.
        dcl  chavar  cha(6)  ini 'abcdef'

     1.    pr binvar                   * Uses the FMT='+9,999' from the DCL statement.
     2.    pr binvar  fmt='xxxx,'      * The FMT string wraps until source exhausted.
     3.    pr binvar  fmt='9999'       * Would give ERR 122 if a dot used in the FMT.
     4.    pr binv2                    * Uses the default FMT='SS,SSS,SSS,SS9'
     5.    space 1
     6.    @a = 22
     7.    pr  'Value of @A with default FMT is ......... '   @a    * No FMT used.
     8.    pr  'Value of @A with FMT=999 is ............. '   @a  fmt='999'
     9.    pr  'Value of binvar with FMT in DCL is ...... '   binvar
    10.    pr  'Value of binv2 with no FMT is ........... '   binv2
    11.    space 1
    12.    binvar = 8
    13.    pr  'Value of binvar after "binvar = 8" ...... '   binvar
    14.    binvar = @a
    15.    pr  'Value of binvar after "binvar = @a" ..... '   binvar
    16.    space 1
    17.    pr  'Value of chavar as initialized .......... "'  chavar  '"'
    18.    chavar = 33
    19.    pr  'Value of chavar after "chavar = 33" ..... "'  chavar  '"'

    INPUT   SEL SEL                                                                  RECORD
    RECNO   TOT ID.          1         2         3         4         5         6     LENGTH
    -----   --- --- ....,....0....,....0....,....0....,....0....,....0....,....0     ------
        0     1   1 +0,123                                                             80
        0     1   2 0000,007B                                                          80
        0     1   3 0123                                                               80
        0     1   4           +456                                                     80

        0     1   7 Value of @A with default FMT is .........            +22         80
        0     1   8 Value of @A with FMT=999 is ..............  022                  80
        0     1   9 Value of binvar with FMT in DCL is ......  +0,123                80
        0     1  10 Value of binv2 with no FMT is ...........            +456        80

        0     1  13 Value of binvar after "binvar = 8" ......  +0,008                80
        0     1  15 Value of binvar after "binvar = @a" .....  +0,022                80

        0     1  17 Value of chavar as initialized ..........  "abcdef"              80
        0     1  19 Value of chavar after "chavar = 33" .....  "33    "              80
                     ....,....1....,....2....,....3....,....4....,....5....,....6
SUMMARY..
 SEL-ID      SELTOT      FILE     BLKSIZE  LRECL           FSIZE   CI    DSN
 ------      ------      ----     -------  -----           -----   --    ---
    1---19        1

    ** SELCOPY/WNT 3.00.006  Licensed by Compute (Bridgend) Ltd  +44 (1656) 652222 & 656466 **
                                  ** Expiry: 02 Mar 2013 **
```

2010/10/24 s208_924

WNT: File Sharing

File sharing for Windows platforms erroneously allowed both READ and WRITE access to all users, which gave problems when multiple concurrent users accessed the same file.

Multiple access to the same file is now properly restricted to Read-Only access.

READ access (without use of UPDATE)
Permitted, provided no other process has the file open for WRITE or UPDATE. Other processes, also having the file open for READ, may exist.

If another process has the file open for WRITE or UPDATE, the READ access will be denied and the job terminated with ERROR 537.

WRITE access (this includes READ for UPDATE)
Only permitted if the file is not in use by any other process, for either READ or WRITE.

If any other process has the file open for anything at all, access will be denied and the job terminated with ERROR 538 for WRITE, or ERROR 537 for READ with UPDATE.

In brief, if a file is to be modified, SELCOPY will insist on exclusive access, otherwise multiple Read-Only access to the same file is allowed. e.g.

```
     ** c:\djh\cc\slc\ctl\SSLOCK01 **#  L=002 --- 2010/10/19 21:38:02  (L07)
                  * WNT: SQ11888 DBASE - File locking - 2010/09/15
               * Disallow READ if another process has the file open for WRITE.

     opt noban  dw=100  w=222   prtsum=3
     dcl irec   cha(80)
     dcl recno  bin(2)  fmt=999
               * Use of % for EQU names necessary to get substitution in literal strings.
     equ %MST%  sslock01.mst       * Master file. (9 recs.)
     equ %WRK%  sslock01.tmp       * File to be accessed by 2 diff processes at the same time.
     equ %LS2%  \tmp\x.x1          * 2ndry SLCLST file for sub-process.

1.   sys 'copy  %MST%  %WRK%'  s=1   * Ensure the temp work file exists.

               * Start a concurrent SELCOPY job to overwrite the file copied.
               * Keep busy writing by writing very slowly.
               * Note that it must use a different SLCLST file.
2.   sys 'start /I selcopy -lst %LS2%  !rd %MST% !plog !sleep 1 !wr %WRK% !e' s=1
               * sleep 1 sec after each record is enough to keep the sub-process running.

3.   sleep 2 secs    s=1   * Prevent main process from opening %WRK% too quickly,
                           * thus the sub-process will open it first.
4.   add 1  to recno

               * Must use DEFER on the read statement, to prevent immediate open.
5.   read %WRK% into irec   b=82 defer   * Should fail with "File in use.".
               * %WRK% is any old test data.  9 records is adequate.

6.   plog  'Rec ' recno ' just read = ' irec   * Will not get actioned as the
                                               * read statement above should fail.
```

2010/11/09 s208_925

NOT*FOUND*OR*EMPTY message changed.

The warning message "NOT*FOUND*OR*EMPTY" in the summary for a non-existent or zero length input file has been changed.

The ambiguity has been removed by reporting in the summary either:

```
## FILE*NOT*FOUND ##
```

or

```
## EMPTY*FILE ##
```

In both cases, RETCD=8 is set as before.

2010/11/28 s208_926

MVS: CALL statement for Assembler routines.

Assembler routines linked as a LOADMOD on any accessible load library on the system (STEPLIB, JOBLIB, LPA, Link List) may be called directly without having to be linked as a DLL with Language Environment. e.g.

```
dcl dclvar char(100)
call asmrtn     "PARM1"  'PARM2'  1234  dclvar+20  "Parm5"  ...
print 'Return Code from asmrtn='    retcd fmt=999
```

Where:

1. Parameters in quotes are passed to the assembler routine as pointers to string literals.
2. Numeric parameters are passed as pointers to the corresponding position in the work area.
3. Named DCL variables are passed as pointers to the start of the declared variable plus any offset given.

The return code from the assembler routine is then used to set SELCOPY's RETCD value, but only if higher than that already set.

2010/11/30 s300_000

Synchronize release number for Product Suite

The SELCOPY Product Suite has numerous components which, for convenience of reference, have all been adjusted to use the same release number for each global distribution.

The C++ version of SELCOPY for December 2010 is therefore at Release 3.0.

Note that on mainframe platforms, the C++ version of SELCOPY will be the SLC program.

2011/01/28 s300_003

The CHOP statement

Allows chopping a character source field into multiple destination fields, DCL vars, where the source components are delimited by 1 or more blanks within the string.

PARSE is a synonym for CHOP.

RC=8 is set if a destination field is too small to hold the appropriate component from the source field.

If a destination field is arithmetic, the source component is converted, respecting the decimal point. e.g.

```
SELCOPY/WNT 3.00 at Compute Bridgend - Wales - (pw=94)  -djh-      2011/01/28 21:31   PAGE   1
------------------------------------------------------------      ----------------   --------

         ** c:\djh\cc\slc\ctl\SSCHOP01 **   L=005 --- 2011/01/28 20:51:48  (L07)
                      * CHOP statement with arithmetic destination fields - 2011/01/19

         dcl src2      char( 60  )  ini '110119   1,234,567.85   555,666.9999777  .065'
         dcl datepaid  char( 12  )  ini 'xxxx,xxxx1'

         dcl f1        char( 10  )  ini 'xxxx,xxxx1'
         dcl f2        flt (  8  )  ini  123.48      fmt=  'z,zzz,zz9.99'
         dcl f3        dec ( 31,4)  ini  543.228     fmt='zzz,zzz,zz9'
         dcl f4        flt (  8  )  ini   32.08      fmt=      'z,zzz.zzzz'
         equ wid  80
         opt    dw=wid  prtsum=3

     1.    lrecl = wid
     2.    pr '   f1="' f1 '"   f2="' f2 '"   f3="' f3 '"   f4="' f4 '"'

     3.    CHOP src2  INTO    f1 f2 f3 f4

     4.    pr '   f1="' f1 '"   f2="' f2 '"   f3="' f3 '"   f4="' f4 '"'

     5.    space 2
     6.    cvdate f1 style=I  to datepaid fmt='yyyy/mm/dd'
     7.    pr '                  f1="'   f1        '"'
     8.    pr '            datepaid="'   datepaid '"'

         end

    INPUT   SEL SEL                                                                                  RECORD
    RECNO   TOT ID.          1         2         3         4         5         6         7         8 LENGTH
    -----   --- --- ....,....0....,....0....,....0....,....0....,....0....,....0....,....0....,....0 ------
        0     1   2    f1="xxxx,xxxx1"   f2="      123.48"   f3="        543"   f4="   32.0800"          80
        0     1   4    f1="110119    "   f2="1,234,567.85"   f3="    555,667"   f4="    0.0650"          80


        0     1   7                  f1="110119    "                                                    80
        0     1   8            datepaid="2011/01/19  "                                                    80
                    ....,....1....,....2....,....3....,....4....,....5....,....6....,....7....,....8

SUMMARY..
 SEL-ID      SELTOT            FILE     BLKSIZE  LRECL           FSIZE   CI    DSN
 ------      ------            ----     -------  -----           -----   --    ---
    1             1   LRECL=
    2             1   pr
    3             1   chop
    4             1   pr
    5             1   space
    6             1   cvdate
    7             1   pr
    8             1   pr

    ** SELCOPY/WNT 3.00.003  Licensed by Compute (Bridgend) Ltd  +44 (1656) 652222 & 656466 **
                                ** Expiry: 02 Mar 2013 **
```

2011/01/15 s300_003

Testing a field for numeric

A field is considered numeric if all the characters within it are numerics, '0' to '9', or comma, fullstop (period) or blank.

A plus or minus sign which precedes all numbers is accepted as valid numeric. e.g.

```
  if num          fld1         * Where fld1 is a character DCL var.
  IF NOT NUMERIC  fld1
  if fld1 not num
  if NUMERIC      20 AT 2001
```

2011/01/15 s300_003

Use of negation on an IF statement

The keyword, NOT, may be used following an IF, AND or OR statement in order to negate the condition being tested. e.g.

```
if not    pos 20 gt 'M'
if not    numeric   fld1
```

Note the difference:

```
if not    pos any  =  'x'     * Only if 'x' is not found in whole record.
if        pos any not 'x'     * If any char in whole record is not 'x'.
                              *      (Will probably select every record.)
```

2011/02/10 s300_004

Windows - File Redirection with Single Quotes

Windows accepts a single quote as a valid character for a filename, thus if SELCOPY is invoked with the following command:

```
selcopy  -ctl 'abc.ctl'   -lst 'xyz.lst'
```

then SELCOPY will attempt to read its control cards from the file "'abc.ctl'", which almost certainly will not exist, and will write its output listing to the file "'xyz.lst'" including the single quotes, probably not what was intended.

Alternatively, the following command would work ok, with Windows handling the double quotes:

```
selcopy  -ctl "abc.ctl"   -lst "xyz.lst"
```

SELCOPY for Windows has now been updated to disregard single quotes enclosing a filename on the command line, thereby making the first example above equally valid as the second.

If it is required to reference a file that does indeed have enclosing single quotes, the filename should be enclosed in double quotes:

```
selcopy  -ctl "'abc.ctl'"   -lst "'xyz.lst'"
```

2011/02/24 s300_004

PRINT TYPE=S corrections to match original version.

The C++ version of SELCOPY, on releases prior to s300_004, differed from the original mainframe assembler version in its processing of the PRINT TYPE=S statement in several ways, some of which have been corrected as follows, while certain other differences will remain, for the benefit of more control for the user.

Corrections to the C++ version:

1. Headings on the output listing file are now suppressed, thereby matching the original mainframe assembler version.

   For both C++ and Assembler versions: If it is required to print headings, they can be printed from POS HEAD. e.g.

   ```
   read  some.file
   if in 1
     then print type=s  from HEAD       * The heading line.
     then print type=s  from HEAD+160   * The underlining.
     then print type=s  from ' '        * A blank line.

   print type=s          * Print data recs, length 133 bytes.
   ```

2. Length written to the PRINT TYPE=S file is 133 bytes, provided that the length of the line to print is not explicitly coded on the PRINT TYPE=S statement.

3. POS UXLINE is now maintained in the same way as on the original mainframe assembler version.

4. IF LINE = n statements will now perform in the same way as on the original mainframe assembler version.

5. The underlining data for the SELCOPY header line will now be found at POS HEAD+160 which matches the original mainframe assembler version. C++ releases prior to s300_004, it was erroneously at POS HEAD+158.

Differences that persist:

1. Unlike the assembler version, the length of the output line is not fixed at 133 bytes, but variable, with trailing blanks removed.

2. For the Assembler version, the length of the output line may be coded on a PRINT TYPE=S statement, but the length is ignored and 133 bytes are written unconditionally.

   For the C++ version, if LRECL=n, LEN=n or L=n is coded on the PRINT TYPE=S statement, or the syntax "FROM n1 AT p1" or "FROM p1 p2" (without the quotes) is used, then the length written is obeyed as explicitly stated on that particular PRINT statement, but with trailing blanks removed.

   Printing from a DCL variable is considered to have an explicit length.

2011/03/06 s300_005

CMS fix for CLOSE statement

Use of the CLOSE statement, with only the filename of the file mentioned, for a file previously mentioned with both a filename and a DSN arg, resulted in the DSN being ignored and the file being processed as DSN='FILE.ddname.A' e.g.

```
read  ABC.DATA.G
wr  TMP1  dsn='COPY.DATA.T'
if  incount = 22
  then close TMP1      * Problem statement.
  then eoj
```

Note that the CLOSE statement is redundant. The EOJ statement causes the file to be closed properly anyway.

Note also that use of the STOPAFT parameter could have achieved the same thing provided a Return Code of 4 were acceptable:

```
read  ABC.DATA.G     stopaft=22
wr  TMP1  dsn='COPY.DATA.T'
```

2011/03/19 s300_006

OPTION ERRLIM=nn

By default, on finding 10 errors in the Control Statements, the job is terminated.

Primarily for the benefit of the developers during testing, the limit of 10 control statement errors may be changed using the ERRLIM parameter on an OPTION statement.

The ERRLIM argument must be numeric and greater than 0.

ERRMAX is a synonym for ERRLIM.

2011/04/13 s300_006

&DCLvar usage to refer to the address of DCLvar

The syntax &DCLvar may be used to refer to the address, rather than the value, of the arithmetic DCL variable, DCLvar. e.g.

```
DCL ABC  BIN(4)  INI '16,909,060'  * Which is X'0102,0304'.  Note that
                    * the INIT argument is enclosed in quotes to allow
                    * use of punctuation.

print ABC           * Will print the value held in ABC, formatted
                    * using the default 14 byte FMT='SS,SSS,SSS,SS9'
                    * giving the 14 byte string:
                    *       '   +16,909,060'

print &ABC          * Will print the field ABC as a character string
                    * using the length of the declared field, ABC,
                    * which in this case is 4.
                    * For this particular example, the 4-byte binary
                    * field, ABC, contains X'0102,0304' where all 4
                    * chars are unprintable, so all get translated to
                    * dot.  So the line printed is:
                    *       '....'

print &ABC TYPE=B   * Will print the field ABC, length 4, in both
                    * character and hex, taking up 4 print lines.
                    * The 1st line is a space line.
                    * The 2nd line is the character representation.
                    * The 3rd line is the zone of each character.
                    * The 4th line is the numeric of each character.
                    * i.e.
                    *       '    '
                    *       '    '
                    *       '0000'
                    *       '1234'
```

```
                    * Not the most efficient way of printing the
                    * contents of ABC.

print &ABC FMT='xx,'
                    * Will print the field ABC, length 4, using the
                    * hex format provided, giving:
                    *       '01,02,03,04'
                    * Note that the FMT string wraps until all of the
                    * source data is processed.

print 'ABC in hex = '   &ABC FMT='xx,'
                    * Will print the literal 'ABC in hex = '  followed
                    * immediately by the formatted interpretation of
                    * the &ABC argument, giving:
                    *       'ABC in hex = 01,02,03,04'
```

2011/05/09 s300_006

DCL var assignment from Compound Source

Assignment

An assignment to an arithmetic DCL variable may be made from a combination of other arithmetic DCL variables, @ pointers or arithmetic literals, separated by a + or - operator.

If the first source field is unsigned, it is treated as positive. e.g.

```
DCL FTOT FLT(8)    FMT='sss,ss9.9999'
DCL B1   BIN       INI '11,222'    * In quotes to allow punctuation.
DCL D1   DEC(18,2) INI '11,222.33'
DCL F1   FLT       INI '11,222.33'

FTOT = B1 + D1 + F1  + '33,000.0066'  * Assignment from a Compound Source.
                                      * Scaled literals are supported.
PRINT "FTOT = " FTOT     "    Result should be '+66,666.6666'"
```

Printing

Similarly, a compound source may also be used as an argument on a PRINT statement, with an optional FORMAT parameter, in the same way as for an arithmetic DCL variable. e.g.

```
PRINT  B1+D1+F1 +'33,000.0066'  FMT='sss,ss9.9999'  " Should be same."

PRINT  B1+D1+F1 +'33,000.0066'                      " Should be same."
```

2011/07/09 s300_008

MVS System Symbols supported

On the MVS platform, System Symbols may be referenced within SELCOPY's control statements using the standard MVS notation for a system symbol, viz: the required name prefixed with ampersand ('&') and terminated with a dot ('.') (period or full stop). e.g.

```
print  "The System Symbol, SYSNAME, has a value of '&SYSNAME.'."
```

Substitution of MVS System Symbols is actioned by SELCOPY whether the symbol notation is within a quoted string or not, in the same way as for REXX variable names on MVS or CMS, or %environment_variables% on other platforms.

To show what static system symbols are available on your system, issue the following command from a SELCOPY/i edit window:

```
<tso console syscmd( display symbols )   | Using the CONSOLE command.
```

Percent Notation

For convenience, on SELCOPY control statements, the percent notation, using the percent sign ('%') as the delimiter, instead of ampersand and dot, is also supported for MVS System Symbols. e.g.

```
* The following 4 statements all read the same file.

read   &SYSNAME..TEST
read  '&SYSNAME..TEST'

read   %SYSNAME%.TEST
read  "%SYSNAME%.TEST"
```

Assuming the System Symbol, SYSNAME, has a value of "XYZ", the file to be read will be "XYZ.TEST".

Limitations

1. Where an MVS System Symbol is used for a data set name in the CBLNAME load module, the standard MVS notation for system symbols must be used. The percent notation is only supported when used in control statements.

2. SELCOPY will only recognize the IBM notation on a control card for an MVS System Symbol if the dot terminator is present, whereas other IBM applications accept certain special characters as the terminator.

3. Use of substring notation on a control statement for an MVS System Symbol is not supported by SELCOPY.

2011/07/26 s300_008

OPTION PRINTABLE=hexstr UNPRINTABLE=hexstr

SELCOPY's translate tables, for the various types of printing supported with the TYPE=x parameter on a PRINT statement, may be tailored to suit the application using the PRINTABLE and/or UNPRINTABLE parameters on an OPTION statement, which may be coded within the SELCOPY Control Statements or as an installation standard in the "selcopy.nam" file.

The PRINTABLE and UNPRINTABLE parameters must each have an argument defining the characters to be changed, to become either printable or unprintable, in the translate tables affected.

The PRINTABLE and UNPRINTABLE arguments must be coded in hex notation, thereby avoiding code page problems, but at the same time making it non-portable across different platforms.

In the following example, the unprintable characters that have been changed to become PRINTABLE, for the benefit of readability here, have been replaced with the @ sign for x'AF' and the # sign for x'AC'.

Blank is displayed as x'40', not x'20', because the example was produced on an EBCDIC machine.

```
      ** c:\djh\cc\slc\ctl\SSPR07 **     L=001 --- 2011/07/26 16:38:10  (L07)
      equ wid 64
      opt dw=wid  noban    * Reduce DataWidth and suppress Startup Banner on console.

      OPT   PRINTABLE = x'AF,ac'    UNPRINTABLE=x'6160'

  1.    lrecl = wid
              * ....,....1....,....2....,....3
  2.    p  1 = 'ABC  abc  .. .. .. ..'
  3.    p 11 = x'ac,ac'
  4.    p 14 = x'AF,AF'
  5.    p 17 = x'60,60'
  6.    p 20 = x'61,61'

  7.    p 28 = 'Type=N (No translation.)'
  8.                                       pr type=n
  9.    p 28 = 'No Type coded. (Type=C) '
 10.                                       pr
 11.    p 28 = 'Type=C                  '
 12.                                       pr type=c
 13.    p 28 = 'Type=B                  '
 14.                                       pr type=b
 15.    p 28 = 'Type=M                  '
 16.                                       pr type=m
      end

INPUT   SEL SEL                                                              RECORD
RECNO   TOT ID.          1         2         3         4         5         6  LENGTH
-----   --- --- ....,....0....,....0....,....0....,....0....,....0....,....0....  ------
    0     1   8 ABC  abc  ## @@ -- //      Type=N (No translation.)                64
    0     1  10 ABC  abc  ## @@ .. ..      No Type coded. (Type=C)                 64
    0     1  12 ABC  abc  ## @@ .. ..      Type=C                                  64

    0     1  14 ABC  abc  ## @@            Type=B                                  64
               CCC4488844AA4AA466466444444EA987C4444444444444444444444444444444
               1230012300CC0FF0000110000003875E2000000000000000000000000000000

    0     1  16 ABC  abc  ## @@ 66 66      Type=M                                  64
                             00 11
               ....,....1....,....2....,....3....,....4....,....5....,....6....
```

2011/07/31 s300_008

COMPRESS and EXPAND using DCL vars and no WORKLEN

The introduction of support for DCL variables has effectively made the WORKLEN parameter for defining a workarea redundant.

However, the COMPRESS and EXPAND statements update the current LRECL value with the resultant size of the target field, and this was disallowed if the value exceeded the size of the current record, or WORKLEN size if coded.

Use of the COMPRESS and EXPAND statements, when the destination field is a variable defined with a DCL statement, will now update the current LRECL value properly, regardless of what, if any, WORKLEN value were coded.

Previously, if the resultant size for the target DCL variable exceeded the WORKLEN value, the current LRECL value remained unchanged.

Residual data

COMPRESS and EXPAND statements with a DCL var as the destination field only modify the target up to the resultant compressed or expanded length. The remainder of the target field is unchanged. e.g.

```
                          * ....,....1....,....2....,
DCL SRCE  CHAR(100)  INIT='ppppppp   qqqqqqqqq   rrr'  FILL='s'
DCL DSTN  CHAR( 20)  INI  'd'  FILL='d'
DCL COPY  CHAR(300)  INI  'c'  FILL='c'

COMPRESS        SRCE   into DSTN     * Will process all the 's' chars too.
EXPAND          DSTN   into COPY     * Will set LRECL=100, leaving 200 'c'
                                     * chars unchanged in COPY.
     * Alternatively:
COMPRESS  25 AT SRCE   into DSTN     * Will ignore the 's' chars.
EXPAND     L AT DSTN   into COPY     * Will set LRECL=25, leaving 275 'c'
                                     * chars unchanged in COPY.
```

Error Messages

The general message, which is used on the Mainframe Assembler version of SELCOPY:

```
E550 COMPRESS/EXPAND FAILED
```

has been replaced with the following:

```
E611 COMPRESS/EXPAND - SOURCE/DESTN INVALID LENGTH
E612 COMPRESS/EXPAND - SOURCE/DESTN OVERLAP
E613 COMPRESS/EXPAND - DESTN NOT IN WORK AREA AND NOT A DCL VAR
E614 EXPAND SOURCE LENGTH EXCEEDS END-OF-INPUT FLAG (X'FE') IN WORKAREA
E615 COMPRESS/EXPAND OVERFLOWS DESTN AREA
E616 EXPAND REQD MORE DATA FROM SOURCE THAN AVAILABLE
```

all of which have additional descriptive information in the section below, titled "ERROR Messages - Select Time".

The additional descriptive information for SELCOPY/C++'s ERROR Messages is also included in the editable file, "selcopy.msg", with its appropriate HLQ.

2011/08/15 s300_008

TRAN statement supports HITS parameter

e.g.

```
DCL SRCE  CHAR(100)  INIT='aa bbb cccc dd eeeeee'
DCL STR1  CHAR       INI  'abc'
DCL STR2  CHAR       INI  'xyz'

TRAN SRCE   'ad'  'ad'   HITS=@A     * Will set @A to 4, changing every
                                    * 'a' to 'a' * and 'd' to 'd',
                                    * effectively changing nothing,
                                    * but you get a count of the hits.

TRAN SRCE   STR1  STR2   HITS=@A     * Will set @A to 9 and change the
                                    * contents of SRCE:
                                    *    all 'a' chars to 'x'
                                    *    all 'b' chars to 'y' and
                                    *    all 'c' chars to 'z'
```

2011/09/12 s300_008

DO statement with Parameters for Sub-Rtn

Parameters may be passed on a DO statement to a subroutine, provided the subroutine declares the number of parameters expected or allowed, by naming them on the label defining the start of the subroutine.

The label name defining the subroutine must be suffixed with a colon to indicate that parameter names may follow.

The number of parameters passed on a DO statement to not need to match the number of parameters expected or allowed by the subroutine.

Reference by the subroutine to a parameter which was not passed will result in RC=8 being set for any statement in the subroutine which uses that parameter, and a value of "*?*" is used for the missing parameter.

Excess parameters passed to a subroutine are ignored.

For arithmetic fields passed to a subroutine, the DO statement may provide a FORMAT parameter following the field identifier. e.g.

```
  DCL C1      CHA  INI='Field-1'
  DCL C2      CHA  INI='Field-2'

  @A=1 !@B=2 !@C=3
            * <----------------- TEXT ------------->    <-- P2 ->   <--- P3 --->   <-- P4 ->
  DO OUTRTN  "@A FMT=99   @B+@C FMT=99   @C FMT=99 "   @A FMT=99   @B+@C FMT=99   @C FMT=99

  DO OUTRTN  "C1 'and' C2   "                          C1          ' and '        C2
  EOJ


=OUTRTN:=     TEXT  P2 P3 P4 P5
  SPACE 1
  PRINT "--- DO OUTRTN --- "      TEXT
  PRINT  P2  ' & '  P3  ' & '  P4
  PRINT  P2 P3 P4 P5       * Gives RC=8 due to 5th parameter not provided by caller.
=RET=
```

2012/02/20 s300_009

DSNPFX=NO for MVS

DSNPFX is a vailable only for SELCOPY C++ version

```
 ____________________________________________________________
|                __           __                             |
|               |              |                            |
|               |              |                            |
|   DSNPFX      | =  YES | NO  |    * May be coded on       |
|   USERID      |    ---       |    *    an I/O statement, or |
|   UID         |              |    *    an OPTION statement, |
|               |__          __|    * but not in "selcopy.nam". |
|                                                            |
|____________________________________________________________|
|                                                            |
|   NODSNPFX                     * May be coded on           |
|   NOUSERID                     *    an I/O statement only. |
|   NOUID                        *                          |
|____________________________________________________________|
```

The C++ and Assembler versions of SELCOPY for MVS essentially behave in the same way, obeying the accepted TSO convention for referencing a Data Set Name when using a literal on an I/O control statement, viz:

1. When the DSN is in quotes, the name provided is used as an absolute name without modification.
2. When the DSN is NOT in quotes, the name provided is modified by prefixing with "userid.", where userid is the user's TSO logon id. e.g.

```
READ 'ABC.TEST.DATA'    * Reads 'ABC.TEST.DATA'
READ      TEST.DATA     * Reads 'uid.TEST.DATA'
```

There are however 3 known differences between the C++ and Assembler versions, described below.

Therefore, to facilitate use of the C++ version, the DSNPFX keyword may be coded on an OPTION statement, or following the DSN parameter on an I/O statement, in order to control the automatic prefixing of the userid on a DSN which is not in quotes.

The DSNPFX keyword and argument are accepted as valid for all C++ supported platforms, but only affect MVS. On other platforms DSNPFX is silently ignored.

The DSNPFX keyword affects only a DSN which is not enclosed in quotes. A DSN enclosed in quotes will always be treated as an absolute name, regardless of any DSNPFX coded.

Use of DSNPFX=YES|NO on an OPTION statement in the system wide file, "selcopy.nam", is disallowed.

Default is DSNPFX=YES.

USERID and UID are synonyms for DSNPFX

NODSNPFX, NOUSERID and NOUID are synonyms for DSNPFX=NO.

Differences

There are 3 known differences between the C++ and Assembler versions of SELCOPY for MVS, concerning the interpretation of a DSN supplied by the user:

DSN in workarea:

When a DSN is supplied as data in the workarea, the C++ version of SELCOPY, SLC, still obeys the DSN prefix convention, observing the same rules as for a DSN supplied as a control statement literal. The caller's userid is prefixed unless the DSN is enclosed in quotes, whereas

the Assembler version always treats a DSN in the workarea as an absolute name, stripping off and ignoring any enclosing quotes.

Solution:
Enclose the DSN in the workarea in quotes, making both versions operate in the same way. Alternatively, code DSNPFX=NO on the I/O statement for the C++ version to make it operate in the same way as the Assembler version.

BATCH work:
Under batch, the C++ version of SELCOPY will use the RACF userid, or its equivalent, instead of the TSO userid, and apply the same DSN prefix convention as it does for TSO, whereas the Assembler version under batch always treats a DSN as an absolute name regardless of its source, stripping off and ignoring any enclosing quotes that may exist.

Solution:
Same as for #1 above, but applies also to a DSN provided as a control statement literal as well as a DSN provided in the workarea.

Unquoted INPUT DSN:
For INPUT files where the DSN is not in quotes, the C++ version of SELCOPY, after a failed OPEN with the userid prefix, will retry the open as an absolute name, without the userid prefix, whereas the Assembler version will not retry the failed open.

Solution:
Enclose the DSN in quotes, avoiding ambiguity, to make both versions operate in the same way. Alternatively, code DSNPFX=YES, or DSNPFX with no arg, on the READ statement for the C++ version to inhibit the 2nd open attempt.

Note that coding DSNPFX=YES on an OPTION statement will have no effect.

Example:
The dataset TEST.ABC exists, but user U01 wishes to read his own version, U01.TEST.ABC which has accidentally been deleted. READ TEST.ABC would normally read U01's version, but because it's not found, SLC C++ reads the unprefixed TEST.ABC dataset, with no error message.

READ TEST.ABC DSNPFX=YES however would give the correct Not Found msg.

2012/03/19 s310_001

MVS: Oversized SELCOPY.MSG file now tolerated

On MVS, the SELCOPY.MSG file should be allocated as RECFM=VB with an LRECL value of at least 256.

An abend ocurred on previous SLC builds when oversized geometry was used for the file. e.g. RECFM=FB,LRECL=256

This abend will no longer occur, but it is still recommended that RECFM=VB is used.

2012/04/12 s310_002

MVS: Trailing blanks on SYSPRINT eliminated

SLC's listing file on SYSPRINT, which it dynamically allocates as RECFM=VB, on earlier builds had trailing blanks on certain lines of output. All trailing blanks are now removed.

2012/05/12 s310_002

Pagination error fixed

Bottom of the page was not recognised when reached during a PRINT TYPE=D statement.

2012/05/17 s310_002

CMS: Reading off an attached VSE disk

Reading a file off an attached z/VSE disk via a CMS FILEDEF is now fully supported by SLC, as in the SELCOPY Assembler version.

However, the real VSE DSN is reported in the summary by SLC, whereas SELCOPY reports the DDNAME of the FILEDEF.

2012/05/20 s310_002

CMS: RECFM=U on CMS minidisks

Files on CMS minidisks are either RECFM=F or RECFM=V. RECFM=U for a CMS file does not exist.

For RECFM=V files on CMS minidisks, the RDW (4 byte Record Descriptor Word) is hidden from the application by the system, and consequently is never seen by SLC or SELCOPY.

SELCOPY, the Assembler version:
Treats RECFM=V and RECFM=U as synonymous and reports RECFM=U in the summary for RECFM=V files on a CMS minidisk.

It has been this way since SELCOPY was first adapted to run on CMS using CMS I/O for minidisk files in the 1970's. Although wrong, the Assembler version must remain unchanged due to extensive use in legacy applications that are still operational.

SLC, the C++ version:
Also treats RECFM=V and RECFM=U as synonymous, but correctly reports RECFM=V in the summary for RECFM=V files on a CMS minidisk.

2012/06/26 s310_003

MVS: Abended with 2 LIST input files.

Use of a 2nd input file using the LIST keyword with an argument in a DCL var or in the workarea resulted in an 0C1 abend. The following sample job will now run cleanly.

```
** c:\djh\cc\slc\ctl\SSLL10 ***    L=007 --- 2012/06/22 11:47:45  (L07)
     * Martin said:  "I want to read the VTOCs of every DASD VOLume and tried this:"
 dcl vol_rec    char(6)
 dcl vtoc_rec   char(500)     ini '<--- ################# ** vtoc_rec area, length 500.'
 dcl lv_cmd     char          ini 'LV volser'

equ safestop  99      * Safety stopaft.
equ wid       72      * Truncate long DSNs for print output.
equ vtoc_where    where "recfm='VB' & alu='T'"                    * Reduce output.
equ vtoc_sel      sel   "Org RecFm Lrecl Blksz Alu Vol Dsn"       * Reduce output.

option worklen=10000
opt    noban  pw=94  dw=wid  prtsum=3    * notrap

=vol_loop=                                                          **
  line 1            s=1
  read   INVOL    list='LVOL *'  select='VOL'   into vol_rec

  lv_cmd+03 = vol_rec
  space 2               s=safestop
  print   'lv_cmd = "'        lv_cmd        '"                *' '*' '*'

  do  vtoc_rtn
  goto vol_loop     s=2       * Just the 1st 3 disks.
  eoj

=eo_vol=
  close INVOL
  eoj


=vtoc_rtn=   * Sub-Rtn *                                            ***
=vtoc_loop=                                                         **
* read   INVTOC  list lv_cmd       head  into vtoc_rec  vtoc_where vtoc_sel  * Gives Err542.
*           Err542 due to HEAD treated as upper limit of the range, lv_cmd to head.
*           POS HEAD is selcopy's storage for heading line, which is lower than lv_cmd storage,
*           resulting in a negative length.  Admittedly, Err542 (@ptr not set) is misleading.
* read   INVTOC  list 9 at lv_cmd  head  into vtoc_rec  vtoc_where vtoc_sel  * Works ok with explicit length.
* read   INVTOC  list lv_cmd       into vtoc_rec  head  vtoc_where vtoc_sel  * Works ok with HEAD coded elsewhere.

  read   INVTOC  list lv_cmd       hdr   into vtoc_rec  vtoc_where vtoc_sel  * Works ok with HDR instead of HEAD.
  if eof INVTOC
    then  goto eo_vtoc

  print   vtoc_rec l=wid s=safestop
* wr   SSLL10.LISTOUT                     fr vtoc_rec  recfm=vb b=4096  * Native MVS file.
  wr  '/mnt/l07/c/tmp/MVS/ssLL10_out.tmp' fr vtoc_rec                   * PC file on HFS.
*|<e _         c:/tmp/MVS/ssLL10_out.tmp
  goto vtoc_loop

=eo_vtoc=
  close INVTOC
=ret=
```

2012/05/19 s310_002

CMS: Recursion Loop fixed.

When a command line option is coded to define a Control Statement input file which does not exist, earlier builds of SLC looped. e.g.

```
slc  -ctl nonexeist.ctl.a
```

Message "ERROR 156 OPEN FAILED FOR INC FILE" is now issued instead.

2011/10/25 s300_008

Other Fixes

All Platforms:

1. RC=8 is now set for a DIVIDE by zero operation.

CMS:

1. The CP command is now fully supported, as in the Assembler version.
2. The UPDATE statement now updates the correct record.

MVS:

1. DIR and DIRDATA processing on a PDS has been brought more into line with the Assembler version of SELCOPY.
2. Use of RECFM=U coded on the control statement as an override is now accepted and obeyed. Previously gave ERROR 571 geometry conflict.

Messages

The following section describes updates made to existing messages and messages that have been introduced since SELCOPY C++ Version 2.08 Build 387.

ERROR Messages - Control Statement Analysis

E003 SELCOPY.NAM FILE NOT FOUND

SELCOPY has failed to read the file "selcopy.nam" from the following sources:

1. For CMS only, any accessed CMS disk.
2. The current directory.
3. The same directory as the SELCOPY program file.
4. Any directory in the PATH.

E009 POS/LEN/REC/STEP=0 OR NEG HAS NO MEANING

The "OR NEG" has been added.

E027 RESRVD Was: UNSUPPORTED STRING NOTATION

See control statement analysis messages E270 to E277 which replace E027 giving more information.

E061 DCL STMT - UNKNOWN DATA TYPE

E106 DBASE: FMT REQD OR INVALID

Originally for the ADABAS database, but also possible using ODBC.

E107 DBASE: SEQ REQD FOR KEYED READ

Originally for the ADABAS database, but also possible using ODBC.

E122 ARITH FMAT HAS TOO MANY DECIMAL POINTS

Only 1 radix point can be supplied in the FORMAT argument used for the destination field of a floating point or decimal conversion.

E127 OFFSET ILLEGAL ON ARITH OR PTR ASSIGNMENT TARGET

The target for an assignment, to an @ pointer or an arithmetic DCL variable, may not have an offset.

E135 DATA TYPE UNSUITABLE FOR ARITH COMPARE

TYPE=B|P|F|Z only are supported for arithmetic compare.

E146 DBASE: DIFF SSN/PLAN

E148 DBASE: TAB/FMT/SORT/SRCH/UPD ON 1 STMT - OMIT HERE

E162 LIBSELC SHARED LIB OR NAMED RTN NOT FOUND

Either the shared library, "libselc", was not found, or it did not contain the function required. See message on terminal for name of library or function not found.

Note:

1. Function names on all systems are case sensitive, and for this reason the case used for the function name on the CALL statement is preserved.
2. It is the user's responsibility to create the shared library containing the CALL routines by compiling and linking the C source code in the slccall.c and slccall.h files, which are supplied with the SELCOPY product. Please refer to slccall.c for more info.
3. The shared library is only required when the CALL statement is used.

E163 MICRO FOCUS SHARED LIB OR EXTFH RTN NOT FOUND

Either the shared library, "cblrtss", was not found, or it did not contain the function, "extfh". See message on terminal for name of library or function not found.

Note:

1. For SELCOPY to process Micro Focus Cobol files, it is necessary that the user has a valid current licence for "Net Express", or equivalent product from Micro Focus, on the machine being used. Please refer to: http://www.microfocus.com/

E182 RESERVED WORD MAY NOT BE USED AS A LABEL

Normally, a control card with only 1 word is treated as a user label which may be referenced by DO and GOTO statements. Certain words however are exceptions and are treated as commands. Reserved words are:

CANCEL	GG	PLOG	R
DUMMY	L	PR	REPORT
E	LOG	PRINT	RET
EL	OPT	PRT	RETURN
ELSE	OPTION	PRTCTL	SPACE
END	OPTIONS	PUNCH	STOP
EOJ	PDUMP	QUIT	WTO

E183 OPT MFC REQD FOR VSAM SUPPORT

VSAM-type file organizations are supported natively on IBM mainframe platforms. On other platforms, VSAM-type file organizations are currently only supported when the host machine has Micro Focus Cobol (MFC) proprietary software installed.

Micro Focus Cobol software is available from Micro Focus International Limited, 9420 Key West Avenue, Rockville, Maryland 20850.

E184 FLEN ARGS MUST BE NUMERIC GT 0 AND NO MORE THAN 98 OF THEM

Field LENgths for a CSV (Comma Separated Variable) record used on a COMPRESS or EXPAND statement must be positive integers.

The maximum no of FLEN arguments is 98, the minimum is 1.

E185 ARG FOR DLM, ENC OR ESC MUST BE LEN 1

COMPRESS and EXPAND statements, for CSV (Comma Separated Variable) processing, may use the keywords, DLM, ENC and ESC, with or without an argument. If an argument is coded, it must be a single character, otherwise it is assumed that the default value is required and the intended argument is left unprocessed, to be validated as a separate keyword, which will result in ERROR 045 if not recognized as such.

When the DLM keyword is coded with no argument, or DLM is omitted altogether, the default is `DLM=','` (Delimiter is Comma.)

Defaults when the ENC and ESC keywords are coded with no argument are `ENC='"'` (Enclosing char is Double Quote), `ESC='\'` (Escape char is BackSlash).

Default when both ENC and ESC keywords are omitted altogether is that all chars of the source, other than the DLM chars, are treated as data. No check is made for enclosing or escape chars.

E186 ENC/ESC ARE MUTUALLY EXCLUSIVE

COMPRESS and EXPAND statements, for CSV (Comma Separated Variable) processing, may use the keywords, ENC or ESC, but not both.

E187 STR/IFNEC/ALL ARE MUTUALLY EXCLUSIVE

The COMPRESS statement, for CSV (Comma Separated Variable) processing, may use the keywords, STR, IFNEC or ALL, but only one of them. If coded on an EXPAND statement, they are validated but ignored.

E188 LIBNAME ARG MISSING OR NOT IN QUOTES

If OPTION LIBNAME="libnam" is used, it must be provided with an argument, which is the name of the Shared Library to be used instead of the default, "libselc.so" for UNIX, or "libselc.dll" for Windows. The argument, "libnam", must be provided in single or double quotes.

Note:

1. The run-time shared library name may only be changed once.

E189 LIBNAME ALREADY CHANGED

Only one OPTION LIBNAME=xxx.xxx statement may be provided for changing the name of the Shared Library to be used instead of the default, "libselc.so" for UNIX, or "libselc.dll" for Windows.

E190 INVALID CALLTYPE ARG

If OPTION CALLTYPE is used, it must be provided with an argument, which may only be DIRECT or VIA_SLCCALL.

DIRECT (The default, which is recommended.)
Means that user-written routines in the shared library, "libselc.so", will be called directly with just 1 argument, a pointer to an array of pointers to a char string.

VIA_SLCCALL (Obsolete.)
Means that user-written routines in the shared library, "libselc.so", will NOT be called directly. Instead, a user-written routine called "slccall" will be called with 2 arguments:

1. A pointer to a char string holding the name of the required routine.
2. A pointer to an array of pointers to a char string.

The "slccall" function must then call the required routine with whatever arguments are required.

E191 DEFAULTFP ARG INVALID

Default Floating Point notation may only be set to HEX (synonym HFP), BIN (synonym BFP), or NATIVE (synonym NAT).

HEX means Base 16 (Hexadecimal) notation, as on IBM Mainframes.
BIN means Base 2 (Binary) notation (IEEE 754), as on PCs and UNIX.
NAT is the default, native notation appropriate to local machine.

Note:

1. Additional conversion is reqd if DEFAULTFP is set to the non-native notation. It has to be converted to/from native format before and after performing each floating point operation.

2. All floating point data is read from and written to storage in Big Endian format. i.e. The senior byte first (on the left).

E192 REM PARAM INAPPROP FOR FLOAT

Floating Point divide does not produce a remainder. The result is accurate to the maximum available precision, which means that any remainder is therefore too small to represent.

Note that precision differs for single precision (32-bit) and double precision (64-bit), and also differs for Base 16 (Hexadecimal) notation and Base 2 (Binary) notation.

E193 FORMAT NOT APPROP FOR DESTN DATA TYPE

For CVxx (conversion) statements, the FORMAT parameter is only appropriate where the destination field is in Character notation, or on the CVCH conversion statement where the destination is in Hex notation.

If a FORMAT string is used for an arithmetic destination field, the TYPE parameter should be omitted or coded as TYPE=C. For a FORMAT string, TYPE=C is default.

E194 SELCODB SHARED LIBRARY NOT FOUND OR INVAL

The shared library, "selcodb.dll" or "selcodb.so", was not found, or it did not contain the "slcodbc" function which is required for processing a DataBase using Open Data Base Connectivity. See message on terminal for name of library or function not found.

Note:

1. The shared library, SELCODB, is supplied as part of the distibution material for the SELCOPY product, but is only required when ODBC statements are used for issuing SQL commands on DataBases such as DB2 and Oracle.

E195 ... E199 RESRVD

E200 DIRTYPE/TYPE/TYPEDIR ARGUMENT FOR DIR/DIRDATA INPUT UNKNOWN

DIR and DIRDATA input may use the DIRTYPE parameter to control the selection of directory entries based on the file Attributes reqd.

The argument for DIRTYPE may be any subset or combination of the string ".ADHNRSY".

Code	File Attrib reqd
.	The "." and ".." entries if present.
R	Read-only.
H	Hidden.
S	System.
V	Volume-ID.
D	Subdirectory.
A	Archive. (Amended file.)
Y	Symbolic Link. (UNIX only.)
N	Normal. (No attribute flags set.)

Blanks and commas are treated as punctuation and ignored, but if used, the TYPE argument must be enclosed in quotes.

E201 DCL INIT VALUE EXCEEDS FIELD WIDTH

E202 DCL INIT VALUE FOR BIN MUST BE INTEGER ONLY

E203 DCL INIT VALUE FOR DEC EXCEEDS SCALE

E204 DCL INIT VALUE FOR DEC EXCEEDS PRECISION

E205 DCL FIELD SIZE CONFLICTS WITH FORMAT ARG

E206 DCL OFFSET NOT WITHIN FIELD

E207 DCL INVALID SCALE

The scale for a Packed Decimal DCL variable may not be negative and must be less than its precision. Default scale is 0.

E208 ... E209 RESRVD

E210 SLCLIST MODULE NOT FOUND

SELCOPY failed to load the mainframe load module or TEXT file, SLCLIST, which is used for reading LIST tables with SELCOPY/i routines. Ensure that SELCOPY/i is installed on your system.

E211 fSLCLIST OPEN FAILED

E212 LOAD IKJCT441 FAILED FOR MVS XV STMT

The IKJCT441 assembler load module was not found, but is needed for the XV statement on MVS..

E213 ONLY GET,SET,NEXT SUPPORTED FOR MVS XV STMT

The DROP,ARG,SOURCE and VERSION arguments are only supported for CMS.

E214 CONFLICTING OPERATOR FOR DATATYPE TEST

Testing for NUMERIC, ALPHA, ALPHANUMERIC, HEX, OCTAL can only have the NOT or EQ operators. Some examples are: e.g.

```
IF  P 1001 L=20    NUMERIC   * Valid.
IF  20 AT 1001  =  NUM       * Valid.
if not num   fld1            * Valid, where fld1 is a char DCL var.
OR  DCLVAR123  NOT NUMERIC   * Valid.
AND DCLVARXYZ   GT NUMERIC   * Error 214. Bad operator, GT.
```

The keywords ALPHA, ALPHANUMERIC, HEX and OCTAL are not yet supported.

E215 AMBIGUOUS DCL FIELD ATTRIBS

Ambiguous or conflicting attributes for a field. For example, when the field is an arithmetic DCL variable, which has an implied length, and a different field length is also supplied as a separate keyword. e.g.

```
DCL   FLT4 FLOAT(4) INIT=1.234
DCL   FLT8 FLOAT(8) INIT=5.678
DCL   BOTH CHAR (12)   POS=FLT4    * Overlay on FLT4 and FLT8.
PRINT FROM FLT4  TYPE=B  LEN=12    * Invalid - Error 215.
PRINT FROM BOTH  TYPE=B  LEN=12    * Valid.
PRINT FROM BOTH  TYPE=B            * Valid.  Length got from DCL.
```

E216 INVALID DCL COMBINATION

On the CHOP statement, the combination of DCL variables is not permitted. Also, use of a DCL variable as a length argument is not permitted on a DCL field. e.g.

```
DCL   FLT4 FLOAT(4) INIT=1.234
DCL   FLT8 FLOAT(8) INIT=5.678
CHOP  '1.3 4.6'  INTO FLT4+FLT8    * Invalid - Error 216.
CHOP  '1.3 4.6'  INTO FLT4 FLT8    * Valid.
PRINT FR FLT4 LEN=FLT8             * Invalid - Error 216.
PRINT FR FLT4 LEN=20               * Valid.
```

E217 HEX ARG REQD FOR OPTION PRINTABLE

On an OPTION statement, the parameters, PRINTABLE and UNPRINTABLE, both require an argument which must be coded in hexadecimal. e.g.

```
OPT   PRINTABLE=x'B1,BB'   UNPRINTABLE = x'ADBD'
```

E218 TRAN STMT - INVALID USE OF HITS PARAM

The HITS parameter is only suitable for the TRAN statement when the required translation is supplied as as 2 strings, the first defining the characters to be translated and the 2nd defining their respective substitution characters. Note also that the argument of the HITS parameter must be a variable which will be set by SELCOPY with the number of characters that were translated.

E219 DATA TYPE CONFLICT ON ASSIGNMENT

Assignment to a CHA field may not be made from an arithmetic field. Either use TYPE=Z for the destination field, treating it as Zoned Decimal, or use a FORMAT parameter for the destination, or use a CHA field as the source. e.g.

```
DCL   DEC7  DEC(7)  INIT=1234
MOD  8 AT 1                =  DEC7   * Error 219.
MOD  8 AT 1 TY=Z           =  DEC7   * Valid, giving "00001234".
MOD  8 AT 1 FMT='zzzz,zz9' =  DEC7   * Valid, giving "   1,234".

MOD  8 AT 1                = &DEC7   * Valid, but DEC7 raw data
                                     * (4 bytes) will be used as
                                     * a CHA string with the destination
                                     * blank padded.
```

E220 ... E269 RESRVD

E270 UNSUPPORTED STRING NOTATION

E271 UNMATCHED QUOTE, SINGLE OR DOUBLE.

E272 UNMATCHED OPEN OR CLOSE BRACKET.

E273 EMBEDDED X'00' FOUND.

E274 RECORD LEN EXCEEDS CONTINUATION CARD BUFFER.

E275 ILLEGAL HEX DIGIT.

E276 HEX STRING EXCEEDS SUPPORTED LENGTH

E277 DATA FOLLOWS IMMED AFTER TERMINATING QUOTE.

ERROR Messages - Selection Time

E523 fFILENAME NOT FOUND OR CONFLICT

The filename used on, an IF-type statement testing for EOF, DIR, DATA, or checking the INCOUNT value, or on a THEN-type statement flagging EOMEMB, EODIR or EODISK, either references a file that has not been used elsewhere, or does not refer to a DIR or DIRDATA input file for 'IF DIR|DATA' or for 'FLAG EOMEMB|EODIR|EODISK'.

For an IF-type statement, the selection id reported is that of the following THEN-type statement.

E555 fDBASE: PREPARE FAILED

E556 fDBASE: CONNECT FAILED

E557 fDBASE: OPEN FAILED

E558 fDBASE: CLOSE FAILED

E559 fDBASE: DISCONNECT NOT CLEAN

E566 STOP COMMAND OR ^C ISSUED BY OPERATOR

The run has been interrupted by the operator and terminated prematurely.

E573 fDYN DSN OR DBASE TABNAME UNRESOLVED

E586 fWORKLEN REQD FOR READING DATABASE

E587 fDBASE: UNKNOWN SSN/USER/PASS FOR DATABASE

One or more of the following is unknown and required:

1. The Sub System Name (SSN=xxxx) for the database.
2. The userid (USER=uuuuuuuu)
3. The password (PASS=password)

Code the appropriate parameters on the I/O statement or on an OPTION statement within the control statements. Alternatively, use an OPTION statement within your selcopy.nam file to affect all SELCOPY jobs.

E588 fDBASE: DELETE FAILED

E589 fDBASE: EXEC FAILED

E590 fDBASE: ALLOC HANDLE FAILED

E591 fDBASE: CONFLICTING VERSIONS FOR SELCOPY AND SHARED LIB

The release number and build level of the SELCOPY program being used differs from that of the dynamic shared library routine, selcodb.so or SELCODB.DLL for Windows, which has been loaded by SELCOPY for use in processing a DataBase such as DB2 or Oracle.

E592 fDBASE: OPEN FAILED FOR LOGSQL FILE

OPTION LOGSQL='fileid' has been coded, either in the SELCOPY control statements or in the selcopy.nam file, and the OPEN for this output file has failed. For example:

```
  OPTION LOGSQL='abc\SelcSQL.log'    * Using a relative path and
                    * the current directory does not have an "abc" directory.
```

E593 fDBASE: UPDATE FAILED

E594 fDBASE: SET CURSOR NAME FAILED

E595 fDBASE: INSERT FAILED

E596 fDBASE: BIND FOR INSERT FAILED

E600 INIT FAILED FOR ARIT DCL
The INIT argument is either an invalid decimal number or exceeds the limit for the arith variable defined.

Note:

1. For a Floating Point variable, the INIT argument must include a dot specifying the position of the decimal point, otherwise the argument is treated as a BIN(4) literal and ERROR 600 will be issued if the value exceeds the BIN(4) limit. e.g.

```
DCL ABCVAR    FLOAT(8)  INIT='2,147,483,647'   * Works ok.
DCL ABCVAR2   FLOAT(8)  INIT='2,147,483,648'   * Gives ERROR 600.
DCL ABCVAR3   FLOAT(8)  INIT='2,147,483,648.0. * Works ok.
```

E601 fSLCLIST READ FAILED
More detailed information will be shown on SELCOPY's listing file.

E602 ALLOC 24-BIT STORAGE FAILED
MVS only: Failed to allocate dynamic storage below the line for a legacy subroutine being invoked with the CALL statement.

E603 CALLING 24-BIT RTNS NOT SUPPORTED WITH DCL VARS
MVS only: A legacy below-the-line subroutine (AMODE=24) is being invoked with the CALL statement, but DCL variables have been used which are held in above-the-line storage (AMODE=31). Either remove the use of DCL vars or recompile the CALL routine with AMODE=31.

E604 ... E610 RESRVD

E611 COMPRESS/EXPAND - SOURCE/DESTN INVALID LENGTH
Either the source length is negative, or the destination length is 0 or negative.

E612 COMPRESS/EXPAND - SOURCE/DESTN OVERLAP
Overlapping source and destination fields are not supported.

E613 COMPRESS/EXPAND - DESTN NOT IN WORK AREA AND NOT A DCL VAR
The destination field must be in writeable storage, either within the work area as defined by the WORKLEN parameter, or in a declared variable as defined by a DCL statement.

E614 EXPAND SOURCE LENGTH EXCEEDS END-OF-INPUT FLAG (X'FE') IN WORKAREA
All compressed records are terminated with X'FE' as a validity check. When the source field for an EXPAND statement is in the work area, as defined by the WORKLEN parameter, the specified length of the source, or its implied length, the current LRECL value, is honoured. Thus, when the end-of-input-data flag, X'FE', is encountered in the workarea, it is checked to ensure that it is the last char of the EXPAND source data, thereby verifying its validity.

However, when the source field for an EXPAND statement is a DCL variable, the X'FE' is tolerated earlier than the last byte of the DCL var, allowing the user to reference a DCL var without an explicit length, in which case the default full length is used. However, if an explicit length is used, it must be correct.

Note that X'FE' chars may also exist within a compressed record as part of the data. The terminating X'FE' char appended to a compressed record is NOT part of the data.

E615 COMPRESS/EXPAND OVERFLOWS DESTN AREA
The size of the expanded source data for an EXPAND statement, or the size of the compressed source data for a COMPRESS statement, exceeds the size of the destination field.

E616 EXPAND REQD MORE DATA FROM SOURCE THAN AVAILABLE
Expansion of the source field for an EXPAND statement has exhausted the source data available without reaching its valid end, a X'FE' character. If the source length used is correct, it is likely that the compressed record provided for the EXPAND source has been corrupted.

E617 RESRVD

WARNING and Information Messages in Summary

Changes to the text of the information messages that may ocurr on the summary are:

1. The text:

   ```
   ### **OPEN*FAILED** ###
   ```

   has been replaced with:

   ```
   ### OPEN*FAILED ###
   ```

2. In order to differentiate between a non-existent file and an empty file which exists on the filesystem but has a length of 0 bytes, the text:

   ```
   ### **NOT*FOUND*OR*EMPTY** ###
   ```

   has been replaced with:

   ```
   ## FILE*NOT*FOUND ##
   ```

   or:

   ```
   ## EMPTY*FILE ##
   ```

   e.g.

```
SUMMARY..
 SEL-ID      SELTOT      FILE     BLKSIZE  LRECL           FSIZE   CI    DSN
 ------      ------      ----     -------  -----           -----   --    ---
    1             0 READ nonexist   2048  2046 U              0       C:\djh\cc\slc\nonexist
                                ## FILE*NOT*FOUND ##
                          (***01 RETCD=8***)
    2             0
```