
# SELCOPY C++ Version (SLC) 3.20 New Features

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

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

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

# Contents

# SELCOPY C++ Version (SLC) Release 3.20 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) 3.20 Build 001 since SELCOPY (SLC) 3.10 Build 001.

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 **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 3.20 Build Level 001 that have been introduced since SELCOPY C++ Version 3.10 Build Level 005 and have not yet been included in the SELCOPY manual.

| 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. |
| WNT Windows | 3.10 | 005 | 2012/12/13 15.12 | | Internal only. |
| CMS Mainframe | 3.10 | 005 | 2012/12/13 15.12 | 2012/12/19 | GA. (ftp.cbl.com/cms) |
| WNT Windows | 3.10 | 006 | 2013/01/27 21.58 | | Internal only. |
| HPX HP-UX | 3.10 | 006 | 2013/01/27 21.58 | 2013/02/08 | GA. (ftp.cbl.com/hpx) |
| WNT Windows | 3.10 | 007 | 2013/05/10 12.07 | | Internal only. |
| LNX Linux | 3.10 | 007 | 2013/05/10 12:06 | 2013/05/10 | GA. (ftp.cbl.com/lnx) |
| WNT Windows | 3.10 | 008 | 2013/05/27 17:12 | 2013/06/21 | GA. (ftp.cbl.com/wnt) |
| CMS Mainframe | 3.10 | 008 | 2013/05/27 17:12 | 2013/06/21 | GA. (ftp.cbl.com/cms) |
| MVS Mainframe | 3.10 | 008 | 2013/05/27 17:12 | 2013/06/21 | GA. (ftp.cbl.com/mvs) |
| LNX Linux | 3.10 | 008 | 2013/05/27 17:50 | 2013/06/21 | GA. (ftp.cbl.com/lnx) |

# Important Changes

There are no important changes which can give different results from the previous releases of SELCOPY C++ on UNIX, Windows and CMS systems. Upgrading from SLC s310_001 involves only new features, fixes and minor changes.

# New Features and Other Changes

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

**2012/10/16 s310_005**

## Offset on PRINT Output

When the length of data being printed exceeds the DATAWIDTH value as set on an OPTION statement, the offset to the start of the data printed on the 2nd and subsequent lines is also shown.

For example, with OPT DW=50 in effect, a record of length 215 is shown as:

```
   INPUT   SEL SEL                                                    RECORD
   RECNO   TOT ID.          1         2         3         4         5 LENGTH
   -----   --- --- ....,....0...,....0....,....0....,....0....,....0 ------
       9     9   7 10010005....Mrs...Patricia.......Sample...........    215
         +50       ...Mrs Sample..............xxxxxxxx xxxxx.........
        +100       .....xx. xxxxxxxxx xxx.............................
        +150       ..........Fxxxxxxxxx......Kxxx..xxxx xxx..xxxxxx x
        +200       xxxxxx......C..
                   ....,....1...,....2....,....3....,....4....,....5
```

When the offset exceeds 999, it is punctuated with a comma. e.g.

```
     +1,200       xxxxxx......C..
```

**2012/11/24 s310_005**

## MVS: Default RECFM

For MVS, if RECFM is not coded for an HFS output file, the RECFM used is the same as that of the input file. For example, a RECFM=VB native MVS input file would still be written to the HFS as RECFM=VB, with BDWs and RDWs as on MVS.

Previously it defaulted to RECFM=U for an HFS file, regardless of the RECFM of the input file.

**2012/11/29 s310_005**

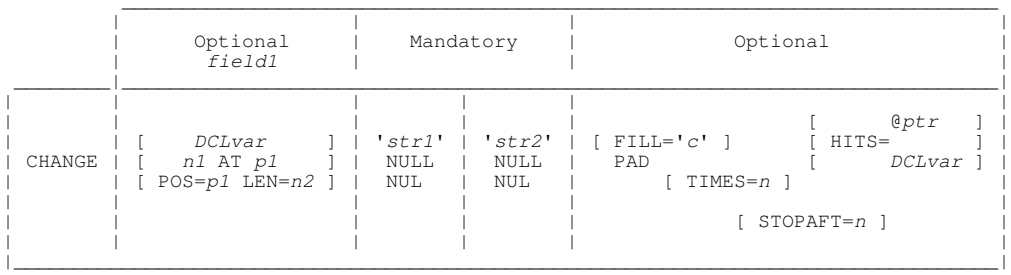
## MVS: LRECL in Summary

For MVS, the LRECL value reported in the summary for RECFM=V or RECFM=U output, and for VSAM output, has been changed to the length of the largest record written during the current run.

Previously, the system defined maximum length allowed was reported.

2013/04/10 s310_007

# CHANGE statement




*field1*
: If coded, field1 defines the character field on which the change command is to operate.

  If omitted, field1 defaults to POS 1 LEN=LRECL. i.e. the current record.

'*str1*'
: Defines the target string, enclosed in quotes, to be found in field1.

  The keyword, NULL or NUL, may be used instead, indicating a null string for the target, and a default of TIMES=1 will be used if TIMES is not coded.

  NULL will of course always exist, so the substitution string will be inserted in front of POS 1 of *field1*.

'*str2*'
: Defines the replacement string, enclosed in quotes, to be substituted for *str1*.

  The length of *str2* may differ from that of *str1*.

  The keyword, NULL or NUL, may be used instead, indicating a null string for the substitution data.

  An error message is given if NULL is used for both *str1* and *str2*.

**FILL**='*c*'
: Defines the fill character to be used when *str2* is shorter than *str1*. If FILL is coded, its argument must be length 1, enclosed in quotes.

  The default FILL char is ' ' (Blank). PAD is a synonym for FILL.

  When the replacement string is short, data to the right of *str1* is shifted left in-situ by the difference between the lengths of *str1* and *str2*, and the FILL char is used to replace the residual data on the right of *field1*.

**TIMES**=*n*
: By default, when NULL is not coded for the target string, the CHANGE command will change every occurrence of *str1* found in *field1* to the *str2* string.

  However, when NULL is coded for the target string, a default of TIMES=1 is used for the CHANGE command.

  Thus, the TIMES parameter is only required when the number of changes are to be restricted, or increased when *str1* is NULL.

**HITS**=@*ptr*
: Defines an arithmetic field into which selcopy is to put the count of the number of changes made.

**Return Code:**

RC=8 will be set if a CHANGE statement causes data, other than data matching the FILL character, to be truncated at the end of *field1* due to *str2* being longer than *str1*.

**Examples:**

```
SELCOPY/WNT 3.10 at Compute Bridgend - Wales - (pw=94)  -djh-      2013/06/16 20:35   PAGE   1
-------------------------------------------------------------      ----------------   --------


        ** c:\djh\cc\slc\ctl\SSCHG04 **    L=001 --- 2013/06/16 20:35:32  (L07)
                       * Example for documentation - 2013/06/16 -djh-
        equ wid          66      * For width of data on a PRINT line.
        opt    pw=94 dw=wid  noban   prtsum=3        * w 2200    Worklen=80 provided by default.
                           * ....,....1....,....2....,....3....,....4....,....5....,....6....,.
        dcl a_z      cha ini 'ABCDEFGHIJKLMNOPQRSTUVWXYZ ### and ### abcdefghijklmnopqrstuvwxyz '
        dcl nums     cha ini 'ABCDEFG  11,222.33  4,555.66  77,888.99  111,222.33  44,555.66    '
        dcl xxx      cha ini 'xxxxxxxxxx   abc    xxxxxxxxxx    def   xxxxxxxxxx    ghi   xxxxxx'


     1.    line 1                s=1
     2.    lrecl = wid

     3.    p 1 = a_z
     4.                  t pr          * Original data.
     5.      t change a_z        '###'  '-'   fill='X' times=3  hits=@hits       * field1 is a DCL var.
     6.      t p 1 = a_z               * Copy changed data to POS 1  LEN=66.
     7.      t do showit   '/###/  to  /-/'


     8.    p 1 = nums
     9.                  t pr          * Original data.
    10.      t change           ','    NULL           times=22 hits=@hits       * field1 is default.
    11.      t do showit   '/,/    to  NULL'
    12.      t change  s=1        '  '   '   '          times=2  hits=@hits  * 2 blanks to 3,  2 times.
    13.      t do showit   '2 blanks to 3 blanks   times 2 '


    14.    p 1 = xxx
    15.                  t pr          * Original data.
    16.      t change           'x'    NULL       pad='Z'   hits=@hits     * Omit TIMES.
    17.      t pr '@hits = '  @hits fmt=99   "   changing  /x/   to   NULL   with fill='Z'"
    18.      t pr
    19.      t space 1
    20.      t change           NULL   '---#'               hits=@hits  * RC=8 expected.
    21.      t do showit   "NULL     to   /---#/  RC=8 as ZZZZ trunc'd."

    22.    eoj



         =showit:=    desc             * The : indicates params exist.
          ------
    23.    pr '@hits = '  @hits fmt=99   '   changing  '   desc
    24.    pr           * Changed data.
    25.    space 2
    26.  =ret=
```

```
SELCOPY/WNT 3.10 at Compute Bridgend - Wales - (pw=94)  -djh-      2013/06/16 20:35   PAGE   2
------------------------------------------------------------      ----------------   --------
    INPUT   SEL SEL                                                                   RECORD
    RECNO   TOT ID.          1         2         3         4         5         6      LENGTH
    -----   --- --- ....,....0...,....0...,....0...,....0...,....0...,....0....,.  ------
        0     1   4 ABCDEFGHIJKLMNOPQRSTUVWXYZ ### and ### abcdefghijklmnopqrstuvwxyz      66
        0     1  23 @hits = 02   changing  /###/  to  /-/                               66
        0     1  24 ABCDEFGHIJKLMNOPQRSTUVWXYZ - and - abcdefghijklmnopqrstuvwxyz XXXX     66


        0     1   9 ABCDEFG  11,222.33  4,555.66  77,888.99  111,222.33  44,555.66       66
        0     2  23 @hits = 05   changing  /,/    to  NULL                               66
        0     2  24 ABCDEFG  11222.33  4555.66  77888.99  111222.33  44555.66           66


        0     3  23 @hits = 02   changing  2 blanks to 3 blanks   times 2                66
        0     3  24 ABCDEFG   11222.33   4555.66  77888.99  111222.33  44555.66         66


        0     1  15 xxxxxxxxxx   abc    xxxxxxxxxx    def   xxxxxxxxxx    ghi   xxxxxx   66
        0     1  17 @hits = 36   changing  /x/   to   NULL   with fill='Z'               66
        0     1  18    abc        def       ghi   ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ   66

        0     4  23 @hits = 01   changing  NULL     to   /---#/  RC=8 as ZZZZ trunc'd.    66
        0     4  24 ---#   abc        def       ghi   ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ   66


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

SUMMARY..
 SEL-ID      SELTOT           FILE     BLKSIZE  LRECL          FSIZE   CI    DSN
 ------      ------           ----     -------  -----          -----   --    ---
    1             1   line
    2             1   LRECL=
    3             1   mod
    4             1   pr                                                 * Original data.
    5             1   change                                             * field1 is a DCL var.
    6             1   mod                                                * Copy changed data to POS 1  LEN=66.
    7             1   do   showit
    8             1   mod
    9             1   pr                                                 * Original data.
   10             1   change                                             * field1 is default.
   11             1   do   showit
   12             1   change                                             * 2 blanks to 3,  2 times.
   13             1   do   showit
   14             1   mod
   15             1   pr                                                 * Original data.
   16             1   change                                             * Omit TIMES.
   17             1   pr
   18             1   pr
   19             1   space
   20             1   change     (***01 RETCD=8***)                      * RC=8 expected.
   21             1   do   showit
   22             1   eoj

                    =showit=                       * The : indicates params exist.
   23             4   pr
   24             4   pr                                                 * Changed data.
   25             4   space
   26             4 =ret=

***WARNING*** (SEL---20)        8 = RETURN CODE FROM SELCOPY

    ** SELCOPY/WNT 3.10.008  Licensed by Compute (Bridgend) Ltd  +44 (1656) 652222 & 656466 **
                                  ** Expiry: 03 Mar 2016 **
```

**2013/04/23 s210_007**

## Multiple Fields allowed on a MOD statement

```
        DCL  XYZ    CHA (80)
        MOD  XYZ  = ABC  DCLvar1  DCLvar2 'literal' DCLvar3 'Lit2'  FROM 20 at 1  'lit3'
```

Provided the destination field of a MOD statement is a CHAR field, multiple source fields may be provided.

All source fields supplied following the destination field are concatenated as character strings, with no intervening blanks, and written to the destination field, XYZ in the above example.

Specifying a source field as a position and length in the workarea will require use of a FROM parameter. e.g. `FROM 20 AT 1`

Any blanks required should be provided as literals.

The same rules for conversion of arithmetic DCL variables to CHAR apply as documented for the PRINT statement. See SELCOPY C++ Version (SLC) 3.10 New Features.

**2013/05/13 s210_007**

## Multiple Fields allowed on a WRITE statement

```
WRITE   [F=]ABC  [FROM] DCLvar1  DCLvar2 'literal' DCLvar3 'Lit2'  FR 20 at 1  'lit3'
REPLACE
UPDATE
```

All fields supplied as arguments to the FROM parameter on a WRITE, REPLACE or UPDATE statement are concatenated as character strings with no intervening blanks and written to the output file, ABC in the above example.

Any blanks required should be provided as literals.

The same rules for conversion of arithmetic DCL variables apply as documented for the PRINT statement. See SELCOPY C++ Version (SLC) 3.10 New Features.

To avoid conversion to character, declare a different variable as a CHA field of the reqd length, overlaying the required DCL variable. e.g.

```
DCL  FLD1   BIN (4)   INIT 123456
DCL  FLD1C  CHA (4)   POS FLD1     * Overdefinition.
WR   ABC   '4 bytes of binary follow>'  FLD1C  '<End of binary data.'
```

The FROM parameter may be omitted for most arguments, but is still needed when the field is defined using n AT p syntax, or POS p LEN n syntax.

Note that the FROM parameter may be used repeatedly on a WRITE statement in order to refer to multiple fields within the workarea or at control blocks for which special POS keywords exist, such as POS DATE or HEAD. e.g.

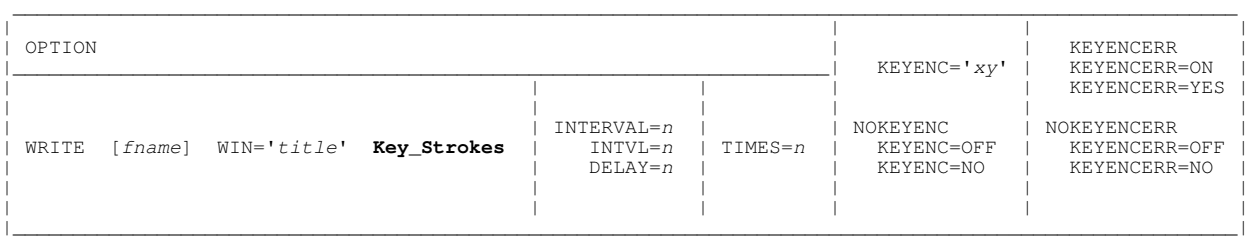
```
slc !p 1 = abcdefg !wr a.tmp  'Lit1 '  from 16 at date-2  ' '  from 3 at 3   " Lit2" !e
```

The above example results in "Lit1 2013/07/08 16:23 CDE Lit2" being written.

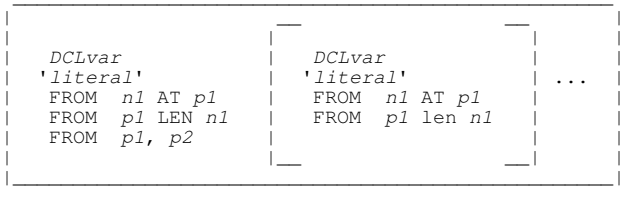
**2013/05/20 s310_007**

## WNT: Writing Keystrokes to a Different Window

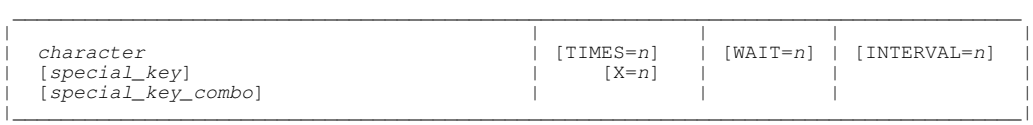
Available ONLY for the Microsoft Windows platform. Note that SELCOPY mainframe and unix server customers are also licensed to use the Microsoft Windows version of SELCOPY.




**Single or Multiple Key_Stroke Specification:**




**Single Key_Stroke:**



## Acknowledgement

Courtesy of Tom Brennan, whose web site http://tombrennansoftware.com sparked off the idea with his zType program, the following is now supported as a built-in feature in selcopy for the MS Windows platform.

Tom Brennan is the architect of the "Vista tn3270 Terminal Emulator" product.

## Description

SELCOPY is able to write keystrokes to a different MS window on the same machine, effectively copy and paste, where the data copied is controlled by the SELCOPY control statements.

The target window may be any window you have open, such as a web browser, an FTP site or Tom Brennan's "Vista tn3270 Terminal Emulator" accessing a z/OS, z/VSE or z/VM m/c.

If the required window is not open, selcopy's SYSTEM command can be used to start the window first, before sending keystrokes to it.

Many tedious tasks can then be automated into a single command. e.g.

```
slc   !sys 'start IEXPLORE.EXE http://www.google.co.uk'   !sleep 4   !wr win=goo 'tn3270[cr]'  !e
```

The `sleep 4` is included because sometimes the google window does not open immediately and results in an error message from SELCOPY saying that the requested window was not found. A delay of 4 seconds is normally more than enough.

Alternatively, you may prefer to set up a command file, where parameter 1 indicates the topic required, such as:

```
:  c:\djh\ca\goo.cmd **            L=001 --- 2013/06/16 21:53:06  (L07)
@echo off
:           Invoke a google search.
: L=001 2013/06/16 -djh- Use SLC to enter topic into google's topic box.
  if .%1==.   (
    echo goo.cmd: Param 1 reqd to define topic.
    goto end  )

  start    IEXPLORE.EXE  http://www.google.co.uk
  slc      !opt noban   !sleep 4   !wr win=goo "%1[CR]"   !e
:end
```

Then, to get the same results, you just need to enter on the command line:

```
goo tn3270
```

## Parameters

**WIN**='*title*'

Identifies the title of the target window, as displayed in the window's title bar, to which keystrokes are to be sent.

If *title* is found as a substring anywhere, within any of the available titled windows, with the exception of the initiating window which has the focus when SLC is executed, it is considered eligible as a target window.

To nominate as the target window the window from which SELCOPY is executed, *title* must be a substring of the window's title starting at offset 0 (zero). This restriction overcomes a problem which occurs when all of the SELCOPY control statements have been supplied as command line parameters. i.e. The CMD.exe window in which SELCOPY executes, adopts a title which contains the entire SELCOPY command string if entered from a command prompt. Therefore, if WIN=*title* is specified in the SELCOPY control statements, the executing window's title will always contain a match for *title*.

All possible windows are searched and the focus is given to the window where *title* is a substring at the lowest offset into a window's title.

If the offset of *title* within a window's title is the same for more than window, then the first window identified will become the target. Therefore, a window with *title* at an offset of 0 (zero) will cause immmediate success.

ERROR 617 indicating that the Window Title was not found, will only be given if *title* is not found anywhere within the title name of any of the available windows.

**INTERVAL**=*n*

INTERVAL, INTVL or DELAY on the WRITE statement, specifes the default interval between keystrokes that are sent to a different window from SELCOPY.

The numeric argument, *n*, on an INTERVAL parameter represents the number of milliseconds to wait between each keystroke for the data string being written and may differ for each WRITE statement. Default is 10 milliseconds.

The INTERVAL coded will take effect when the WRITE statement starts and will remain in effect for all WRITE statements to that window, until changed by a different WRITE statement or by an [INTERVAL=n] encountered within the output data string.

**TIMES**=*n*

TIMES is a parameter, supported on all SELCOPY action statements, with a numeric argument defining how many times the operation is to be actioned. e.g.

```
  WRITE WIN=title  'abc'  TIMES=22   * Will write the string 'abc' 22 times.
```

**KEYENC**='*xy*' | **KEYENC=OFF|NO** | **NOKEYENC**

KEYENC and NOKEYENC are parameters on the following operations:

| | |
|---|---|
| OPTION | Changes the prevailing default. |
| WRITE | Temporarily override the prevailing default. |

KEYENC='*xy*' defines the pair of supported enclosing characters for a Windows Special Key name (See *special_key* and *special_key_combo* below). The 1st character ('*x*') is one of the supported symbols depicting the start of a Windows Special Key name, while the 2nd character ('*y*') is the matching closing symbol depicting the end of the Special Key name. Either apostrophes (') or quotes (") may be used when specifying the character pair.

Supported pairs of KEYENC characters are: '[]' '{}' '<>' '()'

Default is KEYENC = "[]"

Any other character combination will return "ERROR 167 DUMPENC OR KEYENC ARG INVALID".

In the following example, "Editor" identifies the name in the Title Bar of the window to be addressed. The identification of the title is case insensitive and will still succeed if the name supplied is shorter than the actual title. The first window encountered with a title matching the title coded will be accepted and addressed.

```
  OPT  KEYENC = "<>"
  WRITE  WIN="Editor"   FROM "<CR>edit x.x<CR><CR>top<CR>add 22<CR><DOWN><x=5>"
```

The following example demonstrates syntax executed as paramters on the SELCOPY command using the default DUMPENC characters.

```
  selcopy  !write win=ked  from "[cr]e x.x[cr]"   !end     * Default [] used.
```

NOKEYENC, KEYENC=OFF or KEYENC=NO disables the interpretation of all Windows Special Keys.

**KEYENCERR** | **KEYENCERR=ON|YES** | **KEYENCERR=OFF|NO** | **NOKEYENCERR**

KEYENCERR and NOKEYENCERR are parameters on the following operations:

| | |
|---|---|
| OPTION | Changes the prevailing default. |
| WRITE | Temporarily override the prevailing default. |

KEYENCERR, KEYENCERR=ON or KEYENCERR=YES indicates that unrecognised key names, which are enclosed in the current KEYENC delimiters, are treated as data. However, the probable error is highlighted by setting RC=8.

When writing keystrokes to a different window, if an unrecognised Special Key, enclosed in KEYENC delimiters, is encountered in the string to be written, the whole of the Special Key notation, including the KEYENC delimiters, is treated as data and written to the specified target window, but RC=8 (Return Code 8) is set for that selection to indicate the failure.

If more than 1 unrecognised Special Key is found, RC=8 is set for each occurrence and the total displayed in the selection summary for that selection. e.g.

```
  wr WINP   WIN="prim"  '[CR]'   * Write to the "CMD.EXE - Primary" window.
  wr WINP  'rem - Bad special keys: [123][XYZ] get used as data and RC=8 set.[CR]'
```

In the above example, the keystroke data written is as follows (hash "#" represents the ENTER key):

```
  #rem - Bad special keys: [123][XYZ] get used as data and RC=8 set.#
```

NOKEYENCERR, KEYENCERR=OFF or KEYENCERR=NO will suppress the RC=8 for unrecognised key names.

Default at startup is KEYENCERR.

*character*

A keystroke that, when typed, represents a character visible on the display device.

**[***special_key***]**
The string defining the keystrokes to be written may include special keys, represented by enclosing a keyword within delimiters, such as `[ENTER]` where the delimiters used are `[]`. The `[ENTER]` then gets interpreted as hitting the ENTER key. See the `KEYENC` option below for changing the enclosing delimiters.

A *special_key* name is treated by SELCOPY as being case-insensitive. The full list of *special_key* names may be found at:

`http://msdn.microsoft.com/en-us/library/windows/desktop/dd375731(v=vs.85).aspx`

Alternatively, you can google: "Virtual-Key Codes (Windows)".

The `VK_` prefix used by Microsoft's key names should be removed for SELCOPY syntax, but please note that most, but not all, of the defined codes are supported by SELCOPY.

For C++ users, the `VK_` codes are also defined in the **Winuser.h** file, but without description.

Cursor movement is achieved with `[LEFT]`, `[RIGHT]`, `[UP]` or `[DOWN]` codes.

PF keys may be supplied as `[F1]`, `[F2]`, ... `[F24]`

For convenience, selcopy supports the following abbreviations:

| | |
|---|---|
| `[CR]` | `[RETURN]` |
| `[ENTER]` | `[RETURN]` |
| `[LE]` | `[LEFT]` |
| `[RI]` | `[RIGHT]` |
| `[DO]` | `[DOWN]` |
| `[DN]` | `[DOWN]` |
| `[ALT]` | `[MENU]` |
| `[CTL]` | `[CONTROL]` |
| `[CNTL]` | `[CONTROL]` |
| `[CTRL]` | `[CONTROL]` |

**[***special_key_combo***]**
A *special_key_combo* represents a single key stroke of a key with one of the special key ALT, CTL or SHIFT (key modifiers).

To combine a *character* or *special_key* keystroke with one of these key modifiers, the character or special key specification must be imbedded within the delimiters of the ALT, CTL or SHIFT specification. A *special_key_combo* may also be entered with a key modifier to simulate a keystroke involving more than one key modifier. (e.g. Ctrl+Alt+Z)

| | |
|---|---|
| `[ALT x]` | To hit the 'x' key while holding down the Alt key. |
| `[ALT [F1]]` | To hit the F1 key while holding down the Alt key. |
| `[CTL [ALT [F9]]]` | F9, while holding down both Ctl and Alt keys. |

**[TIMES=***n***]** | **[X=***n***]**
TIMES or X, with its numeric argument, may be embedded within the data string being written by the WRITE statement, but must be enclosed within the appropriate `KEYENC` characters.

`[TIMES=n]` will repeat the previous keystroke *n* times, which can be useful for cursor movement. e.g. `[right][times 22]` or `[right][x 22]`

```
OPT  KEYENC = "()"
WRITE  WIN='Title'   "(cr)abc (times 11)def (x 9)(cr)"
   * To get "abc" followed by 12 blanks, then "def" and 10 blanks.
```

**[INTERVAL=***n***]** | **[INTVL=***n***]** | **[DELAY=***n***]**
INTERVAL, INTVL or DELAY, with its numeric argument, may also be embedded anywhere within the data string being written by the WRITE statement, provided they are enclosed with the appropriate `KEYENC` characters.

The INTERVAL coded will take effect immediately and will remain in effect for all WRITE statements to that window, until changed by a different WRITE statement or by another `[INTERVAL=n]` encountered within the output data string.

**[WAIT=***n***]**
WAIT is supported when embedded within the data string being written by the WRITE statement, and must be enclosed within the appropriate `KEYENC` characters. The value *n* is the number of milliseconds to wait. If *n* exceeds 60000 (60 secs) the requested WAIT is ignored.

An immediate unconditional one-off wait will then be actioned prior to continuing with the next keystrokes to be sent to the different window.

Should a larger wait period be required, selcopy's SLEEP statement should be used. e.g.

```
SLEEP 5 MINS      * Duration may be specified in SECS, MINS or HRS.
SLEEP 2           * Default is SECONDS.
```

## Keyboard Languages

When SELCOPY is instructed to write a character as a keystroke to a different window, it translates that character to a Virtual Key Code before passing it to a Microsoft Windows function to actually type it at the cursor position of the target window.

By default, SELCOPY assumes that the destination window has a Standard US Keyboard Layout, which would result in the following special characters being wrongly interpreted if the a non-US keyboard layout is in use (e.g. if a Standard UK Keyboard Layout is in effect):

| Character | ASCII Hex | Description |
|---|---|---|
| " | x'22' | Double quote |
| # | x'23' | Hash |
| @ | x'40' | Commercial at sign |
| ` | x'60' | Backwards quote |
| ~ | x'7E' | Tilde |
| £ | x'A3' | Currency symbol |
| ¬ | x'AC' | NOT sign |

Conveniently, Microsoft Windows provides a method of cycling through the installed keyboard languages which usually comprises English-US and the local region keyboard language. Pressing Alt+Shift keys simultaneously will switch to the next, installed keyboard language.

Therefore, if the current keyboard language is not English-US **and** one of these special characters is to be included in SELCOPY key stroke syntax, the *special_key_combo*, `[ALT [SHIFT]]` may be used to temporarily switch the keyboard language of the target window to English-US. e.g.

```
* "Non_US_test.ctl" is a file on the current directory of a m/c with a non-US keyboard.
*  ---------------
  system 'start notepad'    * No filename, so TitleBar will be "Untitled - Notepad".
  sleep 1                   * Wait 1 second, although unnecessary.
*          The notepad window will have a local keyboard layout.

  wr win='untitled'   delay 20  \
    '[cr]2013/06/20 20.14   "QUOT2"  @AT@  #HASH#  in non-US mode.[cr]'  * Problems.


  wr win='untitled'    '[ALT [SHIFT]]'   * To toggle the US/local keyboard layout.
                                         * Will result in US layout.
  wr win='untitled'   delay 5   \
    '[cr]2013/06/20 20.20   "QUOT2"  @AT@  #HASH#  after switch to US mode.[cr]'

  wr win='untitled'    '[ALT [SHIFT]]'   * To toggle the US/local keyboard layout.
                                         * Will result in restoring local layout.

  wr win='untitled'    '[alt [tab]]'     * To return focus to the window which
                                         * issued the command to run SELCOPY.

  eoj      * The EOJ statement is unnecessary because there is no input file, so
           * the usual loop back to the 1st statement is suppressed.

  end      * The END statement is unnecessary when control statements are from a file.
```

At a command prompt, the following command will run SELCOPY using the control statements in the file "UK_test.ctl" shown above and write the SELCOPY output listing, including control statements, selection totals and diagnostics, to the file "C:\tmp\Non_US_test.lst":

```
  selcopy   -ctl=Non_US_test.ctl  -lst=C:\tmp\Non_US_test.lst
```

The Notepad application window will be updated with:

```
2013/06/20 20.14   ~QUOT2~  "AT"  £HASH£  in non-US mode.

2013/06/20 20.20   "QUOT2"  @AT@  #HASH#  after switch to US mode.
```

Note that, since SELCOPY is simulating typing of keys as if typed by a user at the target window, the Alt+Shift must be typed at the target window and not at the window in which SELCOPY executes. The keyboard language in the SELCOPY execution window is irrelevant since the source of the key stroke string being processed is not via a keyboard but exists in SELCOPY's storage as data.

**Exception:**
If key stroke processing is performed and the current keyboard language is **English-UK**, SELCOPY automatically performs an implied `[ALT [SHIFT]]` before processing key stoke strings and again once key strokes have been processed. Therefore, no specification of `[ALT [SHIFT]]` is required within SELCOPY syntax for users of a US or UK keyboard.

**2013/09/22 s310_009**

# WNT: READ or WRITE CLIPBOARD

Support has been introduced into SELCOPY for reading text from and writing text to the system clipboard as a pseudo file for the Microsoft Windows platform.

Useful in the situation where keystrokes have been written to a different window resulting in the copying of selected data from a browser page to the clipboard. SELCOPY is now able to read this data off the clipboard.

```
 _____________________________________________________________________
|         |                                                           |
|         |                                                           |
|  READ   |    CLIP    [ INTO = field ]                               |
|         |                                                           |
|_____________________________________________________________________|
|         |                                                           |
|         |                                                           |
|  WRITE  |    CLIP    [ FROM = field ]                               |
|         |                ....                                       |
|         |                further                                    |
|         |                  data                                     |
|         |                    args                                   |
|         |                     ....                                  |
|_____________________________________________________________________|
```

The ability to READ from the CLIPBOARD is particularly important when keystrokes have been sent to a different window which is running a web browser.

After keystrokes have been sent to a different window which cause opening of a different page of a web site, it may be required to access some information off the new page, possibly to verify that the correct page is displayed.

This can be done by sending the following keystrokes to the window:

| | |
|---|---|
| `[CTL a]` | which will mark all text data on the page. |
| `[CTL [INS]]` | which will copy all marked text data to the clipboard. |

On return to SELCOPY, the text written to the clipboard can be read with:

```
READ CLIP     * Reads all text data into POS 1 of the workarea and sets LRECL.
```

Then the data can be checked by further control statements.

If the clipboard data is too large to fit in the workarea, the data is returned in a dynamic buffer and the workarea is ignored until the next READ statement, provided the INTO=n parameter were not used and where *n* > 1, otherwise an error message is given.

**2012/11/25 s310_005**

# Other Fixes

## All Platforms:

**SQ12040**:
"Logic Error 3303" was issued when a minus sign (-) was used in an EQU name and the same equate name occurred as part of a subsequent label name.

**SQ12023**:
The bottom-of-page scale line was not printed for PRINT TYPE=B of a large record that involved multiple pages to print all the DATAWIDTH lines needed to display the record.

**SQ12018**:
Assignment of an exponential or rational string literal to a decimal DCL variable assigns incorrect value and sets RC=8. e.g.

```
 DCL  VAR1     DEC(5,2)
 VAR1 = 1.2e+2              * RC=8 and wrong value.
 VAR1 = 1.2                 * RC=8 and wrong value.
```

**SQ12005**:
Assigning a binary field in the workarea, (e.g. 4 at 20 TYPE=B), from another field in the workarea of any data type, set the required binary field, but used the wrong numeric value. Assigning a binary field from a numeric string literal was correct.

**SQ12010**:

The MOVE statement did not respect arithmetic conversion rules when the fields concerened were DCL vars or had TYPE=x coded. The data type was ignored and the fields were treated as CHAR strings.

The MOVE statement now operates as a MOD statement (an assignment) and arithmetic conversion rules are obeyed, using the data types obtained from the DCL vars or from the TYPE=x coded for the fields.

**SQ12079**:
RC=8 should not be returned for an Empty File.

**Original (in error)**
The following is a quote from the snfc310c.txt file:

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.

**The Error**
The last line of the original change above was **wrong**, because previously, RC=8 was **not** set for a genuine existing file that was opened successfully, but happened to be empty.

**Correction**
This is corrected in s310_009 so that RC=8 is ONLY set for a non-existent file. An existing file that has zero records is not an error and no longer gets RC=8 set.

At the same time an improvement has been made to the messages reported in the summary as demonstrated in under the next header.

**Multiple files on same filename**
Many files may be read on the same filename, even allowing the same file to be read more than once. This is done by using the DSN parameter to reference a fileid held in storage. It is therefore possible to get more than 1 occurrence of the FILE*NOT*FOUND or EMPTY*FILE condition on a single READ statement. e.g.

```
   OPT  WORKLEN=2000
   Read CARD into 1801  fill   * A list of fileids to be processed
   if eof CARD
     t eoj
   space 1           * Print a blank line.
   print   from 1801 * Print the fileid.

 =LOOP1=
   read F=ABC  DSN=80 at 1801       * INTO=1 is the default.
   if eof ABC
     t close ABC
     t goto get      * Go read the next CARD record, another fileid.

   print             * Print the record read off ABC.
   if incount > 3
     t close ABC
     t goto get      * Go read the next CARD record, another fileid.
   goto LOOP1

 end                 * End of control stmts.                      \
                       Data records follow, but note that, if the \
                       Data Set Names have leading blanks, then   \
                       the blanks are considered to be part of    \
                       the name.
 a.fil
 nonexist.file1
 empty.fil
 b.fil
 nonexist.file2
 aa.fil
 nonexist.file3
 empty.fil
 bb.fil
 /*                           * Optional End of Data record.  Only reqd if
                                comment records are to follow.
```

In the above example, the reporting in the summary will no longer be just FILE*NOT*FOUND or EMPTY*FILE for the filename ABC, but will also include the number of occurrences of each of the conditions.

So we will get the following 3 messages in the summary for the READ statement for the filename ABC.

```
          3 = FILE*NOT*FOUND
          2 = EMPTY*FILE
   (***03 RETCD=8***)
```

RC=8 is set 3 times only (for the 3 files not found) and the "2 = EMPTY*FILE" message is for information only.

For OPEN and CLOSE statements, the messages in the summary are suppressed.

**SQ12071**:
Fixed in s310_009, multiple fields on a WRITE statement did not give correct output when one of the fields supplied used the FROM parameter.

If the first field was defined using the FROM parameter, subsequent fields were ignored.

If the the FROM parameter used was not for the first field, all fields were concatenated and written, but in the wrong order. The FROM field was written first. e.g.

```
print   from "Lit1"   'Lit2'                    * Gives: Lit1Lit2
wr OFIL from "Lit1"   'Lit2'                    *  Gave: Lit1

pr           'Lit1 '  fr 16 at date-2  " Lit2"  * Gives: Lit1 2013/07/01 21:44 Lit2
wr OFIL      'Lit1 '  fr 16 at date-2  " Lit2"  *  Gave: 2013/07/01 21:44Lit1  Lit2
```

## Microsoft Windows Only:

**SQ12072**:
Fixed in s310_009, when writing keystrokes to a different window, if an unrecognised Special Key, enclosed in `KEYENC` delimiters, (default is `KEYENC='[]'`), is encountered in the string to be written, the writing process for that selection was discontinued and all subsequent data in the string was ignored, with no indication of the failure. e.g.

```
wr WINP   WIN="prim"  '[CR]'   * Write to the "CMD.EXE - Primary" window.
wr WINP  'rem - Bad special keys: [123][XYZ] were ignored on s310_008.[CR]'
```

In the above example the keystroke data written was as follows (hash '#' represents the ENTER key):

```
#rem - Bad special keys:
```

**SQ12072**:
Fixed in s310_009, when writing keystrokes to a window, the following Special Keys, enclosed in `KEYENC` delimiters, (default is `KEYENC='[]'`), did not send the correct key code:

| | |
|---|---|
| `[RSHIFT]` | Right-Hand SHIFT key. |
| `[RMENU]` | Right-Hand ALT key. (AltGr) |
| `[RCONTROL]` | Right-Hand CTL key. |

Also, there was no provision for reference to the Right-Hand ENTER key, so the following has been introduced:

| | |
|---|---|
| `[RCR]` | Right-Hand ENTER key. (Right-CR) |

For mainframe users, sending keystrokes to a 3270 emulator window, the above keys are significant as it is common practice to get the emulator to re-map certain keys to more familiar positions.

In particular, the ENTER key is often re-mapped to NewLine and the Right-CTL remapped to ENTER.

## MVS Only:

**SQ12049**:
Command line options and control statements were ignored.

**SQ11994**:
DIRDATA input of a pre-allocated library (PDS) performs repeated dynamic allocation of a library of the same name but with additional HLQ of the the current user's TSO prefix or, for batch, the RACF userid. e.g. 'ABC.ABC.USERLIB' for 'ABC.USERLIB'

The unnecessary dynamic allocation of the second library occurs once for each member in the pre-allocated library as shown in the JES2 output for the job.

**SQ11981**:
Mainframe VSAM processing has been brought more into line with the Assembler version with the following enhancements:

1. Reading a KSDS by key no longer requires a KEYPOS parameter unnecessarily.
2. READ BWD or FWD parameters are supported as on Assembler version.

# Messages

The following section describes updates made to existing messages and messages that have been introduced since SELCOPY C++ Version 3.10 Build 001.

## ERROR Messages - Control Statement Analysis

**E220 DCL INIT DISALLOWED ON SLC STORAGE**

Initializing DCL storage is not allowed on storage owned by SELCOPY, for example when the POS parameter has been used to define a DCL var which is overlaid on one of SELCOPY's own control blocks. e.g.

```
DCL ABC   CHA(16)  POS DATE-2  INIT='2012/10/04 16:55'  * Error.
```

**E221 UNQUOTED POINTER KEYWORD USED AS CHAR LITERAL**

An assignment statement, where the destination field is character, has a pointer value or keyword as the source field and is therefore ambiguous.

If the arithmetic source field is intended as a character literal, it should be enclosed in quotes.

If the source field is intended as an arithmetic value, the destination field should be defined as TYPE=Z (Zoned decimal) instead or TYPE=C (Char) or the TYPE=C destination should also have a FORMAT parameter. e.g.

```
P 1                 =  RC        * Error.
MOD  4 AT 1         =  RC        * Error.
MOD  4 AT 1 TYPE=C  =  RC        * Error.

P 1                 = 'RC'       * Ok - a CHAR literal.
MOD  4 AT 1         = 'RC'       * Ok - a CHAR literal.
MOD  4 AT 1 TYPE=Z  =  RC        * Ok - arithmetic.
POS 1 FMT='9999'    =  RC        * Ok - arithmetic.
```

**E222 CHANGE STRINGS OR LITERALS BOTH NULL OR INVALID**

**E223 CLIPBOARD I/O MAY ONLY BE READ OR WRITE**

## ERROR Messages - Selection Time

**E617 WINDOW TITLE NOT FOUND**

Check that the Window Title requested on the WRITE WIN='windowtitle' statement matches the wording in the Title Bar of the required window, for the length of your coded 'windowtitle'. The first window encountered that matches your coded title will be selected.

**E618 MULTI FIELD WRITE LENGTH EXCEEDS CONTMAX BUFFER**

A WRITE statement with multiple fields as the source has resulted in a string length which exceeds the storage allocated as a temporary work area for concatenating the multiple fields in preparation for the WRITE. The CONTMAX buffer is used for the concatenation which has a default size of 4096. To increase this size, use: OPTION CONTMAX=n where n is your required limit.

**E619 CLIPBOARD OPERATION FAILED**

## WARNING and Information Messages in Summary

Changes to the text of the information messages that may occurr on the summary are:

1. For warning messages FILE*NOT*FOUND or EMPTY*FILE, the number of times the condition has occurred is also reported. i.e.

```
n = FILE*NOT*FOUND
n = EMPTY*FILE
```