z/OS
Version 2 Release 3

ISPF Edit and Edit Macros

SC19-3621-30

Note

Before using this information and the product it supports, read the information in "Notices" on page 439.

This edition applies to Version 2 Release 3 of z/OS (5650-ZOS) and to all subsequent releases and modifications until otherwise indicated in new editions.

Last updated: 2019-06-21

Contents

Figures xi

Tables xix

Preface xxi
About this document xxi
Who should use this document xxi
How to read the syntax diagrams xxi

z/OS information xxv

How to send your comments to IBM xxvii
If you have a technical problem xxvii

Summary of changes xxix
Summary of changes for z/OS Version 2 Release 3 (V2R3) xxix
Summary of changes for z/OS Version 2 Release 2 (V2R2) xxix
Summary of changes for z/OS Version 2 Release 1 (V2R1) xxix

What's in the z/OS V2R3 ISPF library? xxxi

Part 1. The ISPF editor 1

Chapter 1. Introducing the ISPF editor 3
What is ISPF? 3
What the ISPF editor does 3
Distributed editing 4
Double-byte character set support 4
How to use the ISPF editor 4
Beginning an edit session 4
Using the ISPF editor basic functions 11
Ending an edit session 12
Edit commands 13
Line commands 13
Primary commands 13
Edit commands and PF key processing 14
Edit macros 15
Primary command macros 15
Line command macros 16
Editing data in SCLM-controlled libraries 16
Packing data 16
Specifying z/OS UNIX pathnames with edit primary and macro commands 16

Chapter 2. Controlling the edit environment 17
What is an edit profile? 17
Using edit profile types 17
Displaying or defining an edit profile 17
Modifying an edit profile 18
Locking an edit profile 19

Edit modes....19
Edit profile modes....20
Edit mode defaults....20
Flagged lines....22
Changed lines....22
Error lines....22
Special lines....22
Edit boundaries....23
Initial macros....24
Application-wide macros....25
Statistics for PDS members....25
Effect of Stats mode when beginning an edit session....25
Effect of Stats mode when saving data....25
Version and modification level numbers....26
Sequence numbers....26
Sequence number format and modification level....26
Sequence number display....27
Initialization of number mode....27
Enhanced and language-sensitive edit coloring....27
Language support....28
The HILITE command and dialog....33
Highlighting status and the edit profile....37
Edit recovery....38

Chapter 3. Managing data....41
Creating and replacing data....41
Copying and moving data....41
Shifting data....42
Column shift....43
Data shift....44
Finding, seeking, changing, and excluding data....44
Specifying the search string....45
Effect of CHANGE command on column-dependent data....50
Using the CHANGE command with EBCDIC and DBCS data....50
Working with ASCII data....50
Working with UTF-8 data....51
Controlling the search....52
Qualifying the search string....53
Limiting the search to specified columns....54
Split screen limitations....54
Limiting the search to excluded or non-excluded lines....54
Using the X (Exclude) line command with FIND and CHANGE....54
Repeating the FIND, CHANGE, and EXCLUDE commands....55
Examples....55
Excluding lines....58
Hiding excluded lines....58
Redisplaying excluded lines....58
Redisplaying a range of lines....59
Labels and line ranges....59
Editor-assigned labels....59
Specifying a range....60
Using labels and line ranges....60
Word processing....61
Formatting paragraphs....61
Splitting lines....62
Entering text (power typing)....63
Using tabs....63
Types of tabs....63

Defining and controlling tabs....64
Defining software tab positions....64
Defining hardware tab positions....65
Using attribute bytes....65
Undoing edit interactions....66
UNDO processing....67
Understanding differences in SETUNDO processing....67

Chapter 4. Using edit models....69
What is an edit model?....69
How models are organized....69
How to use edit models....70
Adding, finding, changing, and deleting models....72
Adding models....72
Finding models....75
Changing models....75
Deleting models....76

Part 2. Edit macros....77

Chapter 5. Using edit macros....79
What are edit macros?....79
Performing repeated tasks....79
Simplifying complex tasks....81
Passing parameters, and retrieving and returning information....82
Working with an edit line command table....84

Chapter 6. Creating edit macros....87
CLIST and REXX edit macros....87
Edit macro commands and assignment statements....87
Command procedure statements....88
ISPF and PDF dialog service requests....88
TSO commands....89
Program macros....89
Differences between program macros, CLISTs, and REXX EXECs....90
Passing parameters in a program macro....90
Program macro examples....90
Writing program macros....91
Running program macros....94
Using commands in edit macros....95
Naming edit macros....95
Variables....95
Edit assignment statements....96
Performing line command functions....100
Parameters....101
Passing parameters to a macro....101
Using edit macros in batch....102
Edit macro messages....102
Macro levels....103
Labels in edit macros....103
Referring to data lines....105
Referring to column positions....106
Defining macros....106
Using the PROCESS command and operand....107
Recovery macros....109
Return codes from user-written edit macros....109
Return codes from PDF edit macro commands....110

Selecting control for errors ... 111

Chapter 7. Testing edit macros ... 113
Handling errors ... 113
Edit command errors ... 113
Dialog service errors ... 113
Using CLIST WRITE statements and REXX SAY statements ... 113
Using CLIST CONTROL and REXX TRACE statements ... 114
Experimenting with macro commands ... 115
Debugging edit macros with ISREMSPY ... 116

Chapter 8. Sample edit macros ... 119
ISRBOX macro ... 119
ISRIMBED macro ... 121
ISRMBRS macro ... 124
ISRCHGS macro ... 126
ISRMASK macro ... 130

Part 3. Command reference ... 133

Chapter 9. Edit line commands ... 135
Rules for entering line commands ... 135
Line command summary ... 136
(—Column Shift Left ... 138
)—Column Shift Right ... 140
<—Data Shift Left ... 142
>—Data Shift Right ... 144
A, AK—Specify an After destination ... 146
B, BK—Specify a Before destination ... 149
BOUNDS—Define Boundary Columns ... 151
C—Copy Lines ... 153
COLS—Identify Columns ... 155
D—Delete Lines ... 157
F—Show the First Line ... 158
HX—Show data in hexadecimal format ... 160
I—Insert Lines ... 162
L—Show the Last Line(s) ... 164
LC—Convert Characters to Lowercase ... 165
M—Move Lines ... 167
MASK—Define Masks ... 169
MD—Make Dataline ... 171
O, OK—Overlay Lines ... 173
R—Repeat Lines ... 176
S—Show Lines ... 178
TABS—Control Tabs ... 180
TE—Text Entry ... 181
TF—Text Flow ... 184
TS—Text Split ... 186
UC—Convert Characters to Uppercase ... 188
X—Exclude Lines ... 189

Chapter 10. Edit primary commands ... 193
Edit primary command summary ... 193
AUTOLIST—Create a Source Listing Automatically ... 196
AUTONUM—Number Lines Automatically ... 198
AUTOSAVE—Save Data Automatically ... 199
BOUNDS—Control the Edit Boundaries ... 201

BROWSE—Browse from within an Edit Session ... 202
BUILTIN—Process a Built-In Command ... 203
CANCEL—Cancel Edit Changes ... 203
CAPS—Control Automatic Character Conversion ... 204
CHANGE—Change a Data String ... 205
COLS—Display Fixed Columns Line ... 208
COMPARE—Edit Compare ... 209
COPY—Copy Data ... 212
CREATE—Create Data ... 217
CUT—Cut and Save Lines ... 220
DEFINE—Define a Name ... 222
DELETE—Delete Lines ... 224
EDIT—Edit from within an Edit Session ... 226
EDITSET—Display the Editor Settings Dialog ... 228
END—End the Edit Session ... 231
EXCLUDE—Exclude Lines from the Display ... 232
FIND—Find a Data String ... 234
FLIP—Reverse Exclude Status of Lines ... 236
HEX—Display Hexadecimal Characters ... 238
HIDE—Hide Excluded Lines Message ... 241
HILITE—Enhanced Edit Coloring ... 242
IMACRO—Specify an Initial Macro ... 246
LEVEL—Specify the Modification Level Number ... 247
LF—realign data on the ASCII linefeed character ... 248
LOCATE—Locate a Line ... 249
MODEL—Copy a Model into the Current Data Set ... 251
MOVE—Move Data ... 254
NONUMBER—Turn Off Number Mode ... 258
NOTES—Display Model Notes ... 258
NULLS—Control Null Spaces ... 259
NUMBER—Generate Sequence Numbers ... 260
PACK—Compress Data ... 262
PASTE—Move or Copy Lines from Clipboard ... 263
PRESERVE—Enable Saving of Trailing Blanks ... 264
PROFILE—Control and Display Your Profile ... 264
RCHANGE—Repeat a Change ... 267
RECOVERY—Control Edit Recovery ... 268
RENUM—Renumber Data Set Lines ... 269
REPLACE—Replace Data ... 271
RESET—Reset the Data Display ... 275
RFIND—Repeat Find ... 277
RMACRO—Specify a Recovery Macro ... 278
SAVE—Save the Current Data ... 278
SETUNDO—Set the UNDO Mode ... 280
SORT—Sort Data ... 282
SOURCE—describe format of data ... 284
STATS—Generate Library Statistics ... 285
SUBMIT—Submit Data for Batch Processing ... 285
TABS—Define Tabs ... 286
UNDO—Reverse Last Edit Interaction ... 288
UNNUMBER—Remove Sequence Numbers ... 290
VERSION—Control the Version Number ... 292
VIEW—View from within an Edit Session ... 294

Chapter 11. Edit macro commands and assignment statements ... 295
Edit macro command summary ... 295
AUTOLIST—Set or Query Autolist Mode ... 300
AUTONUM—Set or Query Autonum Mode ... 301

AUTOSAVE—Set or Query Autosave Mode..... 303
BLKSIZE—Query the Block Size..... 304
BOUNDS—Set or Query the Edit Boundaries..... 305
BROWSE—Browse from within an Edit Session..... 307
BUILTIN—Process a Built-In Command..... 308
CANCEL—Cancel Edit Changes..... 308
CAPS—Set or Query Caps Mode..... 309
CHANGE—Change a Search String..... 310
CHANGE_COUNTS—Query Change Counts..... 313
COMPARE—Edit Compare..... 314
COPY—Copy Data..... 317
CREATE—Create a Data Set or a Data Set Member..... 318
CURSOR—Set or Query the Cursor Position..... 320
CUT—Cut and Save Lines..... 322
DATA_CHANGED—Query the Data Changed Status..... 323
DATA_WIDTH—Query Data Width..... 324
DATAID—Query Data ID..... 325
DATASET—Query the Current and Original Data Set Names..... 326
DEFINE—Define a Name..... 327
DELETE—Delete Lines..... 329
DISPLAY_COLS—Query Display Columns..... 330
DISPLAY_LINES—Query Display Lines..... 331
DOWN—Scroll Down..... 332
EDIT—Edit from within an Edit Session..... 333
END—End the Edit Session..... 334
EXCLUDE—Exclude Lines from the Display..... 335
EXCLUDE_COUNTS—Query Exclude Counts..... 337
FIND—Find a Search String..... 338
FIND_COUNTS—Query Find Counts..... 340
FLIP—Reverse Exclude Status of Lines..... 341
FLOW_COUNTS—Query Flow Counts..... 342
HEX—Set or Query Hexadecimal Mode..... 342
HIDE—Hide Excluded Lines Message..... 344
HILITE—Enhanced Edit Coloring..... 345
IMACRO—Set or Query an Initial Macro..... 349
INSERT—Prepare Display for Data Insertion..... 350
LF—realign the data based on the ASCII linefeed character..... 351
LABEL—Set or Query a Line Label..... 352
LEFT—Scroll Left..... 353
LEVEL—Set or Query the Modification Level Number..... 354
LINE—Set or Query a Line from the Data Set..... 355
LINE_AFTER—Add a Line to the Current Data Set..... 357
LINE_BEFORE—Add a Line to the Current Data Set..... 358
LINE_STATUS—Query Source and Change Information for a Line in a Data Set..... 360
LINENUM—Query the Line Number of a Labeled Line..... 362
LOCATE—Locate a Line..... 363
LRECL—Query the Logical Record Length..... 365
MACRO—Identify an Edit Macro..... 366
MACRO_LEVEL—Query the Macro Nesting Level..... 367
MACRO_MSG—Set or Query the Macro Message switch..... 368
MASKLINE—Set or Query the Mask Line..... 368
MEMBER—Query the Current Member Name..... 370
MEND—End a Macro in the Batch Environment..... 370
MODEL—Copy a Model into the Current Data Set..... 370
MOVE— Move a Data Set or a Data Set Member..... 372
NONUMBER—Turn Off Number Mode..... 373
NOTES—Set or Query Note Mode..... 374
NULLS—Set or Query Nulls Mode..... 375

NUMBER—Set or Query Number Mode....376
PACK—Set or Query Pack Mode....379
PASTE—Move or Copy Lines from Clipboard....380
PRESERVE—Enable Saving of Trailing Blanks....381
PROCESS—Process Line Commands....383
PROFILE—Set or Query the Current Profile....384
RANGE_CMD—Query a Command That You Entered....386
RCHANGE—Repeat a Change....387
RECFM—Query the Record Format....387
RECOVERY—Set or Query Recovery Mode....389
RENUM—Renumber Data Set Lines....390
REPLACE—Replace a Data Set or Data Set Member....391
RESET—Reset the Data Display....393
RFIND—Repeat Find....395
RIGHT—Scroll Right....395
RMACRO—Set or Query the Recovery Macro....396
SAVE—Save the Current Data....397
SAVE_LENGTH—Set or Query Length for Variable-Length Data....399
SCAN—Set Command Scan Mode....400
SEEK—Seek a Data String, Positioning the Cursor....401
SEEK_COUNTS—Query Seek Counts....403
SESSION—Query Session Type....404
SETUNDO—Set UNDO Mode....404
SHIFT (—Shift Columns Left....406
SHIFT)—Shift Columns Right....407
SHIFT <—Shift Data Left....408
SHIFT >—Shift Data Right....409
SORT—Sort Data....409
SOURCE—describe format of data....412
STATS—Set or Query Stats Mode....412
SUBMIT—Submit Data for Batch Processing....414
TABS—Set or Query Tabs Mode....415
TABSLINE—Set or Query Tabs Line....416
TENTER—Set Up Panel for Text Entry....418
TFLOW—Text Flow a Paragraph....419
TSPLIT—Text Split a Line....420
UNNUMBER—Remove Sequence Numbers....421
UP—Scroll Up....421
USER_STATE—Save or Restore User State....423
VERSION—Set or Query Version Number....424
VIEW—View from within an Edit Session....425
VOLUME—Query Volume Information....426
XSTATUS—Set or Query Exclude Status of a Line....426

Appendix A. Abbreviations for Commands and Other Values....429
Edit line commands....429
Edit primary commands....429
Parameters....431
Keywords/Operands....431
Scroll amounts....432

Appendix B. Edit-related sample macros....433

Appendix C. Accessibility....435
Accessibility features....435
Consult assistive technologies....435
Keyboard navigation of the user interface....435

Dotted decimal syntax diagrams....435

Notices....439
Terms and conditions for product documentation....440
IBM Online Privacy Statement....441
Policy for unsupported hardware....441
Minimum supported hardware....442
Programming Interface Information....442
Trademarks....442

Index....443

Figures

1. Sample syntax diagram.....xxii

2. Edit Entry panel (ISREDM01)..... 5

3. panelsEdit data displayCreating a new data set (ISREDDE2)..... 9

4. Example Primary Edit panel (ISREDDE2)..... 9

5. Edit Profile display (ISREDDE2)..... 18

6. HILITE Initial Screen (ISREP1)..... 34

7. HILITE Language Element Specification Screen (ISREPC1)..... 36

8. HILITE Language Keyword List (ISREPK)..... 37

9. Examples of edit profile lines showing HILITE options..... 38

10. Edit Recovery panel (ISREDM02)..... 39

11. Before FIND command (ISREDDE2)..... 56

12. After FIND command..... 56

13. Before CHANGE command..... 56

14. After CHANGE command..... 57

15. Before EXCLUDE command..... 57

16. After EXCLUDE command..... 57

17. Model Classes panel (ISREMCLS)..... 70

18. CLIST Models panel (ISREMCMD)..... 71

19. DISPLAY Service Model..... 71

20. Panel Models panel (ISREMPNL)..... 73

21. Changed Panel Models panel (ISREMPNL)..... 73

22. Changed)PROC section of Panel Models panel (ISREMPNL)..... 74

23. Source code for Block Letter Model Selection panel..... 75

24. ISRDASH macroISRDASH, sample macro....80
25. ISRDASH macro - before running....80
26. ISRDASH macro - after running....81
27. ISRTDATA macroISRTDATA, sample macro....81
28. ISRTDATA macro - before running....82
29. ISRTDATA macro - after running....82
30. ISRCOUNT macroISRCOUNT, sample macro....83
31. ISRCOUNT macro - before running....83
32. ISRCOUNT macro - after running....84
33. ISRSLREX REXX macroISRSLREX, sample macro....92
34. ISRSEPP PL/I macroISRSLPLI, sample macro....93
35. ISRSEPC COBOL macroISRSEPC, sample macro....94
36. ISRTDATA macro with CLIST WRITE statementsISRTDWRI, sample macro....114
37. Results of ISRTDATA macro with CLIST WRITE statements....114
38. ISRTRYIT macroISRTRYIT, sample macro....115
39. ISRTRYIT macro - before running....116
40. ISRTRYIT macro - after running....116
41. ISRBOX macro....119
42. ISRBOX macro - before running....121
43. ISRBOX macro - after running....121
44. ISRIMBED macro....122
45. LIST with imbed statements....123
46. ISRIMBED macro - after running....124
47. ISRMBRS macro....125
48. ISRCHGS macro....127

49. ISRCHGS macro - before running....129
50. ISRCHGS macro - after running....129
51. ISRMASK macro....130
52. ISRMASK macro - before running....131
53. ISRMASK macro - after running....132
54. Before the ((Column Shift Left) line command....139
55. After the ((Column Shift Left) line command....140
56. Before the) (Column Shift Right) line command....141
57. After the) (Column Shift Right) line command....142
58. Before the < (Data Shift Left) line command....143
59. After the < (Data Shift Left) line command....144
60. Before the > (Data Shift Right) line command....145
61. After the > (Data Shift Right) line command....146
62. Before the A (After) line command....148
63. After the A (After) line command....148
64. Before the B (Before) line command....150
65. After the B (Before) line command....151
66. Before the BOUNDS line command....152
67. After the BOUNDS line command....153
68. Before the C (Copy) line command....154
69. After the C (Copy) line command....155
70. Before the COLS line command....156
71. After the COLS line command....156
72. Before the D (Delete) line command....158
73. After the D (Delete) line command....158

74. Before the F (Show First Line) line command.....159
75. After the F (Show First Line) line command.....160
76. Before the HX (display in hexadecimal format) line command.....161
77. After the HX (display in hexadecimal format) line command.....162
78. Before the I (Insert) line command.....163
79. After the I (Insert) line command..... 163
80. Before the L (Show Last Line) line command.....164
81. After the L (Show Last Line) line command.....165
82. Before the LC (Lowercase) line command.....166
83. After the LC (Lowercase) line command..... 167
84. Before the M (Move) line command..... 168
85. After the M (MOVE) line command.....169
86. Before the MASK line command.....170
87. After the MASK line command..... 171
88. Before the MD (Make Dataline) line command..... 172
89. After the MD (Make Dataline) line command.....173
90. Before the O (Overlay) line command.....176
91. After the O (Overlay) line command.....176
92. Before the R (repeat) line command.....177
93. After the R (Repeat) line command.....178
94. Before the S (Show) line command.....179
95. After the S (Show) line command.....179
96. TAB line command example.....181
97. Before the TE (Text Entry) line command..... 183
98. After the TE (Text Entry) line command.....183

99. Sample text during text entry mode....184
100. Sample text after text entry mode....184
101. Before the TF (Text Flow) line command....185
102. After the TF (Text Flow) line command....186
103. Before TS (Text Split) line command....187
104. After TS (Text Split) line command....187
105. Before the UC (Uppercase) line command....189
106. After the UC (Uppercase) line command....189
107. Before the X (Exclude) line command....191
108. After the X (Exclude) line command....191
109. Member with COLS indicator line....209
110. Edit Compare Settings and/or Command Parameters panel....212
111. Member before data is copied....215
112. Edit/View - Copy panel (ISRECPY1)....216
113. Contents of member to be copied....216
114. Member after data has been copied....216
115. Member before new member is created....219
116. Edit/View Create panel (ISRECRA1)....219
117. Member after new member has been created....220
118. New member created....220
119. EDIT primary command example....227
120. Edit Command Entry panel (ISREDM03)....227
121. Nested member editing example....228
122. Edit and View Settings panel (ISREDSET)....229
123. EDITSET primary command example....231

124. Example of data set.... 237
125. Example of data set with excluded lines.... 238
126. Example of data set using FLIP on excluded lines.... 238
127. Member with hexadecimal mode off.... 240
128. Hexadecimal display, vertical representation.... 240
129. Hexadecimal display, data representation.... 241
130. Before the HIDE primary command.... 242
131. After the HIDE primary command.... 242
132. Member with modification level of 03.... 248
133. Member with modification level reset to 00.... 248
138. Before Model command.... 253
139. REXX Models panel (ISREMRXC).... 253
140. REXX model of VGET service.... 254
141. Member before data is moved.... 256
142. Edit Move panel (ISREMOV1).... 257
143. Data set to be moved.... 257
144. Member after data has been moved.... 258
148. Edit Profile display.... 267
149. Member before lines are renumbered.... 271
150. Member after lines are renumbered.... 271
151. Member before other member is replaced.... 274
152. Edit/View Replace panel (ISRERPL1).... 274
153. Member after the other member has been replaced.... 275
154. Other member replaced.... 275
155. SETUNDO STORAGE and RECOVERY OFF.... 282

156. Member before lines are deleted..... 289

157. Member after lines are deleted..... 290

158. Member after lines have been restored..... 290

159. Member before lines are unnumbered..... 291

160. Member after lines are unnumbered..... 292

161. Member before version number is changed..... 293

162. Member after version number is changed..... 293

Tables

1. Examples of passing a string to RFIND....15

2. An example of passing string values to RCHANGE....15

3. Default bounds settings for data sets....23

4. Example edit macro commands....88

5. Example edit macro assignment statements....88

6. Passing commands using a different environment....88

7. Service requests in an edit macro....89

8. TSO commands....89

9. Edit assignment statements....96

10. Separating two literals....97

11. Assigning a value to a variable....98

12. Substituting a value in a variable....99

13. Edit macro commands corresponding to line commands....100

14. Summary of the line commands....136

15. Line and primary commans for A and AK....147

16. Line and primary commands for B....149

17. Summary of the primary commands....193

18. Summary of the macro commands....295

19. Cursor position....321

20. Data width return value....325

21. Aliases and abbreviations for Edit line commands....429

22. Aliases and abbreviations for Edit primary commands....429

23. Allowable abbreviations for parameters....431

24. Aliases and abbreviations for keywords and operands..... 431

25. Aliases and abbreviations for scroll amounts..... 432

Preface

This document describes the ISPF editor and provides conceptual, usage, and reference information for the ISPF edit line, primary, and macro commands.

About this document

This document contains three parts:

- Part 1 introduces and describes how to use the ISPF editor.
- Part 2 describes how to use, write and test edit macros. It also discusses sample CLIST, REXX, and program edit macros.
- Part 3 is a reference for the edit line, primary, and macro commands available for ISPF.

Who should use this document

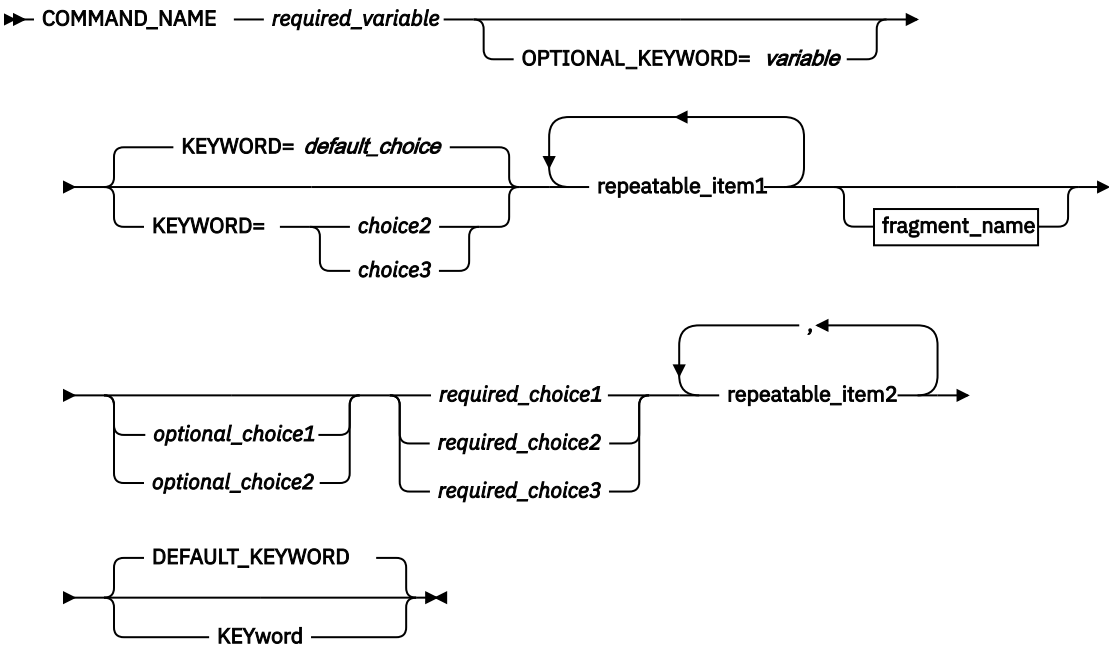
This document is for application and system programmers who develop programs, and who use the ISPF editor and edit macro instructions. Users who write edit macros should be familiar with coding CLISTs, REXX EXECs, or programs in the z/OS® environment.

How to read the syntax diagrams

The syntactical structure of commands described in this document is shown by means of syntax diagrams.

Figure 1 on page xxii shows a sample syntax diagram that includes the various notations used to indicate such things as whether:

- An item is a keyword or a variable.
- An item is required or optional.
- A choice is available.
- A default applies if you do not specify a value.
- You can repeat an item.



fragment_name

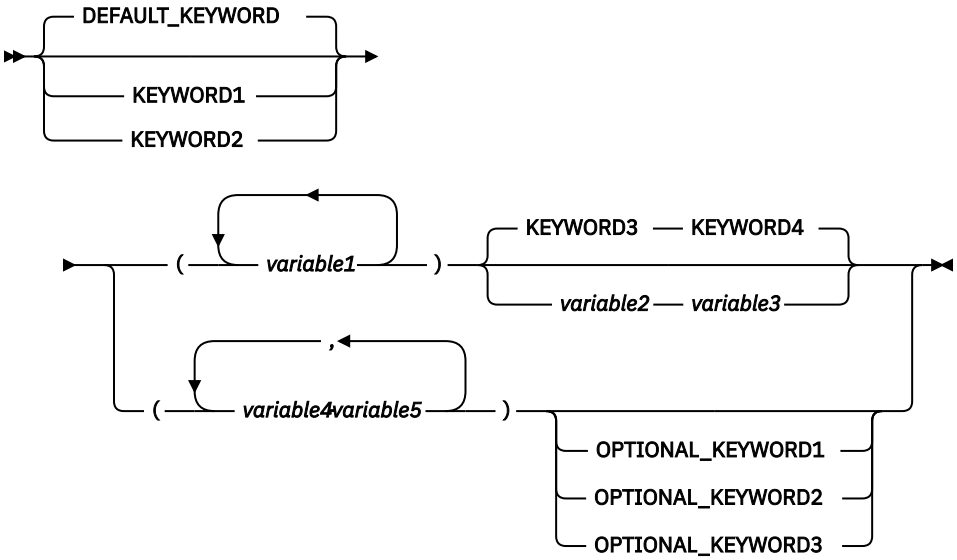


Figure 1. Sample syntax diagram

Here are some tips for reading and understanding syntax diagrams:

Order of reading

Read the syntax diagrams from left to right, from top to bottom, following the path of the line.

The ►►—— symbol indicates the beginning of a statement.

The ——► symbol indicates that a statement is continued on the next line.

The ►—— symbol indicates that a statement is continued from the previous line.

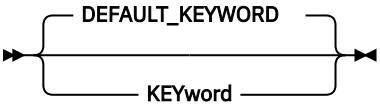
The ——►◄ symbol indicates the end of a statement.

Keywords

Keywords appear in uppercase letters.

►► COMMAND_NAME ►◄

Sometimes you only need to type the first few letters of a keyword, The required part of the keyword appears in uppercase letters.

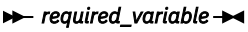


In this example, you could type "KEY", "KEYW", "KEYWO", "KEYWOR" or "KEYWORD".

The abbreviated or whole keyword you enter must be spelled exactly as shown.

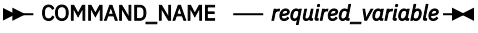
Variables

Variables appear in lowercase letters. They represent user-supplied names or values.



Required items

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



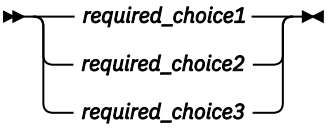
Optional items

Optional items appear below the main path.

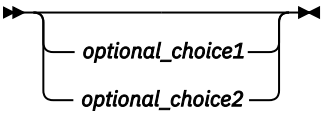
Choice of items

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

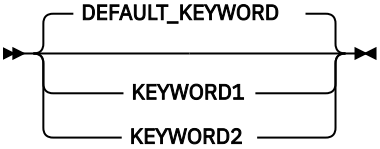
If you *must* choose one of the items, one item of the stack appears on the main path.



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

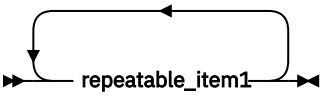


If a default value applies when you do not choose any of the items, the default value appears above the main path.

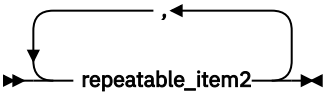


Repeatable items

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

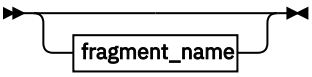


If you need to specify a separator character (such as a comma) between repeatable items, the line with the arrow returning to the left shows the separator character you must specify.



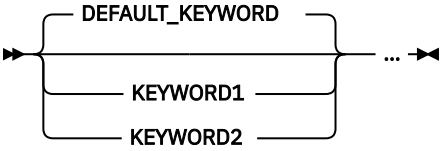
Fragments

Where it makes the syntax diagram easier to read, a section or *fragment* of the syntax is sometimes shown separately.



⋮

fragment_name



z/OS information

This information explains how z/OS references information in other documents and on the web.

When possible, this information uses cross document links that go directly to the topic in reference using shortened versions of the document title. For complete titles and order numbers of the documents for all products that are part of z/OS, see *z/OS Information Roadmap*.

To find the complete z/OS library, go to IBM Knowledge Center (www.ibm.com/support/knowledgecenter/SSLTBW/welcome).

How to send your comments to IBM

We invite you to submit comments about the z/OS product documentation. Your valuable feedback helps to ensure accurate and high-quality information.

Important: If your comment regards a technical question or problem, see instead "If you have a technical problem" on page xxvii.

Submit your feedback by using the appropriate method for your type of comment or question:

Feedback on z/OS function
If your comment or question is about z/OS itself, submit a request through the IBM RFE Community (www.ibm.com/developerworks/rfe/).

Feedback on IBM® Knowledge Center function
If your comment or question is about the IBM Knowledge Center functionality, for example search capabilities or how to arrange the browser view, send a detailed email to IBM Knowledge Center Support at ibmkc@us.ibm.com.

Feedback on the z/OS product documentation and content
If your comment is about the information that is provided in the z/OS product documentation library, send a detailed email to mhvrcfs@us.ibm.com. We welcome any feedback that you have, including comments on the clarity, accuracy, or completeness of the information.

To help us better process your submission, include the following information:

- Your name, company/university/institution name, and email address
- The following deliverable title and order number: z/OS ISPF Edit and Edit Macros, SC19-3621-30
- The section title of the specific information to which your comment relates
- The text of your comment.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute the comments in any way appropriate without incurring any obligation to you.

IBM or any other organizations use the personal information that you supply to contact you only about the issues that you submit.

If you have a technical problem

If you have a technical problem or question, do not use the feedback methods that are provided for sending documentation comments. Instead, take one or more of the following actions:

- Go to the IBM Support Portal (support.ibm.com).
- Contact your IBM service representative.
- Call IBM technical support.

Summary of changes

This information includes terminology, maintenance, and editorial changes. Technical changes or additions to the text and illustrations for the current edition are indicated by a vertical line to the left of the change.

Summary of changes for z/OS Version 2 Release 3 (V2R3)

The following changes are made for z/OS Version 2 Release 3 (V2R3).

June 2019

Maintenance and terminology changes are made for z/OS Version 2 Release 3 in June 2019.

May 2018

Maintenance and terminology changes are made for z/OS Version 2 Release 3 in May 2018.

September 2017
Changed information

- Extended statistics usability, see the following topics:
  - “STATS—Generate Library Statistics” on page 285
  - “STATS—Set or Query Stats Mode” on page 412

Summary of changes for z/OS Version 2 Release 2 (V2R2)

The following changes are made for z/OS Version 2 Release 2 (V2R2).

New information

- Updates for PDSE member generations are added under:
  - “Edit recovery” on page 38
  - “COMPARE—Edit Compare” on page 209
  - “REPLACE—Replace Data” on page 271
  - “SAVE—Save the Current Data” on page 278
  - “COMPARE—Edit Compare” on page 314
  - “REPLACE—Replace a Data Set or Data Set Member” on page 391
  - “SAVE—Save the Current Data” on page 397

Summary of changes for z/OS Version 2 Release 1 (V2R1)

The following changes are made for z/OS Version 2 Release 1 (V2R1).

Editor line command macros

You, or your installation, can now define a table of Edit user line commands and associated Edit macros using the ISPF table editor. The table has a set structure which is enforced by the table editor when you indicate that they are working with an edit line command table. For each command you must specify if it supports a multiline format, a block format, and if it will use a destination. A multiline format is when the you can include a numeric suffix on the command to indicate the number of lines

that the command applies to. When invoking Edit or View, you can specify to use an Edit user line command table and the name of the table to be used. The associated line macro uses the PROCESS macro statement to determine the lines the command applies to and the destination to be used by the macro.

For changes to this document relating to this modification, see:

- “What are edit macros?” on page 79
- “Working with an edit line command table” on page 84
- “Passing parameters to a macro” on page 101
- “Using the PROCESS command with edit line macros” on page 107
- Figure 120 on page 227
- “MACRO—Identify an Edit Macro” on page 366
- “PROCESS—Process Line Commands” on page 383

Editor hilight support
Edit HILITING support for COBOL has been updated to support Enterprise Cobol V4.1, Enterprise PL/I V3.8, and current DTL verbs. Obsolete PL/I functions have been removed.

Browse limit
The BROWSE command has been enhanced to report when the browse limit of 99,999,999 records is reached during a FIND string ALL operation.

Edit macro parameters
In edit macro parameters, ISPF now treats comma delimiters in the same way as blank delimiters.

What's in the z/OS V2R3 ISPF library?

You can order the ISPF books using the numbers provided below.

Title
Order Number

z/OS ISPF Dialog Developer's Guide and Reference
SC19-3619–30

z/OS ISPF Dialog Tag Language Guide and Reference
SC19-3620–30

z/OS ISPF Edit and Edit Macros
SC19-3621–30

z/OS ISPF Messages and Codes
SC19-3622–30

z/OS ISPF Planning and Customizing
GC19-3623–30

z/OS ISPF Reference Summary
SC19-3624–30

z/OS ISPF Software Configuration and Library Manager Guide and Reference
SC19-3625–30

z/OS ISPF Services Guide
SC19-3626–30

z/OS ISPF User's Guide Vol I
SC19-3627–30

z/OS ISPF User's Guide Vol II
SC19-3628–30

Part 1. The ISPF editor

Chapter 1. Introducing the ISPF editor

This topic introduces the ISPF editor. It provides an overview of:

- The ISPF editor functions
- A typical edit session
- Edit line commands and primary commands
- Edit macros

What is ISPF?

The Interactive System Productivity Facility (ISPF) is a dialog manager that provides tools to improve program, dialog, and development productivity and control.

The PDF component of ISPF is an integrated work environment used to develop programs, dialogs, and documents. PDF provides an MVS™-compatible hierarchical library and many productivity-improving functions. Some examples of these functions are:

- ISPF dialog test tools
- Full-screen editor, with a dialog interface called edit macros
- Multiple update access to data sets
- Online tutorials
- Data set management
- Customized library controls

This document describes the ISPF editor and its dialog interface. A *dialog* is a program running under ISPF. The interface allows a dialog to access the usual ISPF dialog functions and the ISPF editor functions.

What the ISPF editor does

You can use the ISPF editor to create, display, and change data stored in ISPF libraries or other partitioned or sequential data sets with these characteristics:

- Record Format (RECFM):
  - Fixed or variable (non-spanned)
  - Blocked or unblocked
  - With or without printer control characters
- Logical Record Length (LRECL):
  - From 1 to 32760, inclusive, for fixed-length records
  - From 5 to 32756, inclusive, for variable-length records.

    Note: For variable-length records, the amount of editable data in each record is 4 bytes less than the logical record length.

Generally, the editor truncates variable-length lines by removing blanks at the end of each line during a save. If a variable-length line is completely blank and has no line number, a blank is added so that the line length is not zero.

However, with the PRESERVE function, you can save the trailing blanks of variable-length files. The "Preserve VB record length" field on the Edit Entry panel and the PRESERVE edit and macro commands enable you to save or truncate the blanks as you prefer.

Distributed editing

ISPF enables you to edit host data on a workstation, and workstation data on the host. ISPF calls this function *distributed editing*.

The ISPF Workstation Tool Integration dialog, or tool integrator, is a workstation customization tool that enables any workstation application to use data from an MVS host system. After setting up the tool integrator, your workstation-installed applications can interact with the ISPF View and Edit functions and services. Data flow goes both ways with the tool integrator connection. You can work with workstation files on the host or with host files on the workstation.

For more information about distributed editing, refer to the *z/OS ISPF User's Guide Vol II* and the *z/OS ISPF Services Guide*.

Double-byte character set support

The ISPF editor supports DBCS alphabets in two ways:

- Formatted data where DBCS characters are in the column positions specified in the format definition created with the Format Utility (option 3.11)
- Mixed characters delimited with the special shift-out and shift-in characters.

  If you are using mixed mode and the record length of a data set is greater than 72 bytes, there is a possibility that a DBCS character might encroach on the display boundary. Here, PDF attempts to display the other characters by replacing an unpaired DBCS character byte with an SO or SI character. If there is a possibility that the replaced SO or SI character was erased, the line number of the line is highlighted. If you change the position of the SO and SI characters on the panel, or if you delete the SO and SI characters entirely, the DBCS character on the boundary is removed to keep the rest of the data intact.

How to use the ISPF editor

This topic provides an overview of an edit session and covers:

- Beginning an Edit Session
- Using the ISPF editor Basic Functions
- Ending an Edit Session

Beginning an edit session

To begin using the ISPF editor, select option 2 on the ISPF Primary Option Menu. PDF then displays the Edit Entry panel (Figure 2 on page 5).

```
  Menu  RefList  RefMode  Utilities  Workstation  Help
_____________________________________________________________________________
                              Edit Entry Panel

ISPF Library:
   Project . . . MYPROJ
   Group . . . . DEV     . . . ________ . . . ________ . . .
   Type  . . . . SOURCE
   Member  . . . ________        (Blank or pattern for member selection list)

Other Partitioned, Sequential or VSAM Data Set, or z/OS UNIX file:
   Name . . . ____________________________________________________________
   Volume Serial . . . ______    (If not cataloged)

Workstation File:
   File Name . . . . . ____________________________________________________

                                    Options
Initial Macro  . . . . ________      /  Confirm Cancel/Move/Replace
Profile Name . . . . . ________      _  Mixed Mode
Format Name  . . . . . ________      _  Edit on Workstation
Data Set Password  . . ________      _  Preserve VB record length
Record Length  . . . . _____
Line Command Table . . ________      Data Encoding
                                    _  1. ASCII
                                       2. UTF-8

Command ===> ___________________________________________________________________
 F1=Help      F2=Split     F3=Exit      F7=Backward  F8=Forward   F9=Swap
F10=Actions  F12=Cancel
```

Figure 2. Edit Entry panel (ISREDM01)

Edit entry panel action bar

The Edit Entry panel action bar choices function as follows:

Menu
For information on the Menu pull-down, see the topic about action bars in *z/OS ISPF User's Guide Vol I*.

Reflist
The Reflist pull-down offers these choices:

1. **Reference Data Set List**
   Displays the Reference Data Set List panel, which displays a list of up to 30 data set names you have referenced in PDF panels.
2. **Reference Library List**
   Displays the Reference Library List panel.
3. **Personal Data Set List**
   Displays the Personal Data Set List panel, of which you can have any number, as long as each has a unique name.
4. **Personal Data Set List Open**
   Displays the **Open** dialog for all Personal Data Sets.
5. **Personal Library List**
   Displays the Personal Library List panel, which maintains up to 8 lists, each with a unique name. If more than one list exists, the most recently used list displays.
6. **Personal Library List Open**
   Displays the **Open** dialog for all Personal Library Lists.

Refmode
Refmode sets reference lists to either retrieve or execute mode. The Refmode pull-down offers these choices:

1. List Execute
Sets reference lists, personal data set list and personal library lists into an execute mode. When you select an entry from the list, the information is placed into the ISPF Library or the "Other" Data Set Name field and an Enter key is simulated. (If this setting is current, the choice is unavailable.)

2. List Retrieve
Sets reference lists, personal data set list and personal library lists into a retrieve mode. When you select an entry from the list, the information is placed into the ISPF Library or the "Other" Data Set Name field, but the Enter key is *not* simulated. (If this setting is current, the choice is unavailable.)

Utilities
For information on the Utilities pull-down, see the topic about action bars in *z/OS ISPF User's Guide Vol I*.

Workstation
Configure ISPF workstation tool integration. For information about the workstation and ISPF, refer to the *z/OS ISPF User's Guide Vol I*.

Help
The Help pull-down provides general information about the Edit environment as well as information about the main options and edit commands.

Edit entry panel fields

You can specify a concatenated sequence of up to four ISPF libraries, but the libraries must have been previously allocated to ISPF with the Data Set utility (3.2).

The fields on this panel are:

Project
The common identifier for all ISPF libraries belonging to the same programming project.

Group
The identifier for the particular set of ISPF libraries; that is, the level of the libraries within the library hierarchy.

You can specify a concatenated sequence of up to four existing ISPF libraries.

The editor searches the ISPF libraries in the designated order to find the member and copies it into working storage. If the editor does not find the member in the library, it creates a new member with the specified name.

When you save the edited member, the editor places or replaces it in the first ISPF library in the concatenation sequence, regardless of which library it was copied from.

Type
The identifier for the type of information in the ISPF library.

Member
The name of an ISPF library or other partitioned data set member. Leaving this field blank or entering a pattern causes PDF to display a member list. See *z/OS ISPF User's Guide Vol I* for information about entering patterns.

Data Set Name
Any fully qualified data set name, such as USERID.SYS1.MACLIB, VSAM data set name, or z/OS UNIX file path name. If you include your TSO user prefix (defaults to user ID), you must enclose the data set name in apostrophes. However, if you omit the TSO user prefix and apostrophes, your TSO user prefix is automatically added to the beginning of the data set name.

If you specify a VSAM data set, ISPF checks the configuration table to see if VSAM support is enabled. If it is, the specified tool is invoked. If VSAM is not supported by the configuration settings, an error message is displayed.

Volume Serial
A real DASD volume or a virtual volume residing on an IBM 3850 Mass Storage System. To access 3850 virtual volumes, you must also have MOUNT authority, which is acquired through the TSO ACCOUNT command.

Workstation File:
If you have made a connection to the workstation, you can also specify a workstation file name, for example C:\AUTOEXEC.BAT, on the Edit Entry Panel. Or you can specify which environment (host or workstation) should be used to edit a data set. With these options, one of four editing situations can occur:

1. Edit a Host Data Set on the Host

   The editor searches the ISPF libraries in the designated order to find the member and copy it into working storage. If you specified a nonexistent member of an ISPF library, a new member is created with the specified name.

   When you save the edited member, the editor places or replaces it in the first ISPF library in the concatenation sequence, regardless of which library it was copied from.

2. Edit a Host Data Set on the Workstation

   The editor searches the ISPF libraries in the designated order to find the member and copy it into working storage. The data set name is converted to a workstation file name, and that name is appended to the workstation's current working directory. The host data set is transferred to the workstation, and the working file is then passed to the user's chosen edit program.

   When you finish the edit session, the working file is transferred back to the host and stored in the first ISPF library in the concatenation sequence.

3. Edit a Workstation File on the Host

   The editor searches the workstation filesystem to find the file and copy it into working storage. The workstation file name is converted to a host data set name, and, if greater than 44 characters, it is truncated to be 44. The workstation file is transferred to the host, where you can edit it.

   When you finish the edit session, the working file is transferred back to the workstation and stored.

4. Edit a Workstation File on the Workstation

   This edit proceeds as it normally does on your workstation.

Initial Macro
You can specify a macro to be processed before you begin editing your sequential data set or any member of a partitioned data set. This initial macro allows you to set up a particular editing environment for the Edit session you are beginning. This initial macro overrides any IMACRO value in your profile.

If you leave the Initial Macro field blank and your edit profile includes an initial macro specification, the initial macro from your edit profile is processed.

If you want to suppress an initial macro in your edit profile, type NONE in the Initial Macro field. See “Initial macros” on page 24 and “IMACRO—Specify an Initial Macro” on page 246 for more details.

Profile Name
The name of an edit profile, which you can use to override the default edit profile. See the description in “What is an edit profile?” on page 17.

Format Name
The name of a format definition or blank if no format is to be used.

Data Set Password
The password for OS password-protected data sets. This is not your RACF® password.

Record Length
Applicable when editing a z/OS UNIX file. ISPF normally treats z/OS UNIX files as having variable length records. This field allows you to specify a record length which is used by the editor to load the records from the file into the edit session as fixed-length records. When the file is saved, it is saved with fixed-length records. The Record Length field allows you to convert a variable-length file to fixed length. The value specified in this field must be able to accommodate the largest record in the file. If the editor finds a record that is larger than the length specified, an error message is displayed and the edit session does not proceed.

Line Command Table
Use this field to define a set of user line commands that you can use during the edit session. The table you specify can be generated using the ISPF table editor and contains the line commands that you wish to have available and associates each line command with an edit macro that will be run if the line command is entered during the edit session.

Confirm Cancel/Move/Replace
When you select this field with a "/", a confirmation panel displays when you request one of these actions, and the execution of that action would result in data changes being lost or existing data being overwritten.

- For MOVE, the confirm panel is displayed if the data to be moved exists. Otherwise, an error message is displayed.
- For REPLACE, the confirm panel is displayed if the data to be replaced exists. Otherwise, the REPLACE command functions like the edit CREATE command, and no confirmation panel is displayed.
- For CANCEL, the confirmation panel is displayed if any data changes have been made, whether through primary commands, line commands, or typing.

  Note: Any commands or data changes pending at the time the CANCEL command is issued are ignored. Data changes are "pending" if changes have been made to the displayed edit data, but no interaction with the host (ENTER, PF key, or command other than CANCEL) has occurred. If no other changes have been made during the edit session up to that point, the confirmation panel is not displayed.

Mixed Mode
When you select this field with a "/", it specifies that the editor look for shift-out and shift-in delimiters surrounding DBCS data. If you do not select it, the editor does not look for mixed data.

Edit on Workstation
You can select this option to use your workstation as the editing environment for whichever host data set or workstation file you want to edit.

Preserve VB record length
You can select this option to cause the editor to store the original length of each record in variable-length data sets and when a record is saved, the original record length is used as the minimum length for the record.

Data Encoding
You can use this option to select whether to edit data as ASCII (CCSID 819) or UTF-8 (CCSID 1208). When you select a value for this option, the editor uses the selected CCSID in converting the data to the CCSID for the terminal.

You can also specify this option when creating a new file to contain ASCII or UTF-8 data.

For z/OS UNIX files, the editor breaks up data into records using the ASCII (and UTF-8) linefeed character (X'0A') and the ASCII (and UTF-8) carriage return character (X'0D') as the record delimiter. The linefeed and carriage return characters are removed from the data loaded into the editor, but written back to the file when the data is saved. When the file is saved, ISPF ensures the file is tagged with a CCSID of 819 (or 1208).

Creating a new data set

Before you can edit a new sequential data set, you must allocate space for it. When you specify an empty sequential data set or nonexistent member of a partitioned data set, the first edit display contains several empty lines between the `Top of Data` and `Bottom of Data` message lines (Figure 3 on page 9). The editor replaces the quote marks on the left of the panel with sequence numbers when you type information on the lines.

See “Creating and replacing data” on page 41 and “Word processing” on page 61 for more information on using the editor to create data.

```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
------------------------------------------------------------------------------
EDIT       SBURNF.PRIVATE.DATA(EDITNEW) - 01.00            Columns 00001 00072
****** ***************************** Top of Data ******************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
''''''
Command ===> ____________________________________________ Scroll ===> CSR
 F1=Help      F2=Split     F3=Exit      F5=Rfind     F6=Rchange   F7=Up
 F8=Down      F9=Swap     F10=Left     F11=Right    F12=Cancel
```

Figure 3. Creating a new data set (ISREDDE2)

Editing an existing data set

When you edit an existing data set, ISPF displays the Primary Edit Panel as shown in Figure 4 on page 9.

```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
------------------------------------------------------------------------------
EDIT       SBURNF.PRIVATE.DATA(EDITOLD) - 01.00            Columns 00001 00072
****** ***************************** Top of Data ******************************
000100 PROC 0
000200 EX 'PDFTOOL.COMMON.EXEC.(ALLOCPDF)' 'REL(DEV) FVT NOTOOLS'
000300 PDF
****** **************************** Bottom of Data ****************************









Command ===> ____________________________________________ Scroll ===> CSR
 F1=Help      F2=Split     F3=Exit      F5=Rfind     F6=Rchange   F7=Up
 F8=Down      F9=Swap     F10=Left     F11=Right    F12=Cancel
```

Figure 4. Example Primary Edit panel (ISREDDE2)

Primary Edit panel action bar choices

The Primary Edit panel action bar choices function as follows:

File.
The File pull-down offers you these choices:

1. **Save**
   Executes the SAVE command.
2. **Cancel**
   Executes the CANCEL command (which ignores all changes made to the member) and redisplays the Edit Entry panel.
3. **Exit**
   Executes the END command (which saves the data set or member) and redisplays the Edit Entry panel.

Edit
The Edit pull-down offers you these choices:

1. **Reset**
   Performs the RESET command.
2. **Undo**
   Performs the UNDO command.
3. **Hilite**
   Displays the Edit Color Settings pop-up.
4. **Cut**
   Cuts the selected data from the file, placing it on the clipboard.
5. **Paste**
   Puts the selected data from the clipboard into the chosen area of the current file.

Edit_Settings
When selected, causes an additional panel to display to enable you to set the characteristics of your edit sessions.

1. **Edit settings**
   Causes the additional panel to display.

Menu
For information on the Menu pull-down, see the topic about action bars in *z/OS ISPF User's Guide Vol I*.

Utilities
For information on the Utilities pull-down, see the topic about action bars in *z/OS ISPF User's Guide Vol I*.

Compilers
The Compilers pull-down provides shortcuts to the compilers on the Foreground Selection Panel and Batch Selection Panel, the ISPPREP panel preprocessing utility, and the DTL compiler.

Test
The Test pull-down offers you these choices:

1. **Functions**
   Displays the Dialog Test Function/Selection panel.
2. **Panels**
   Displays the Dialog Test Display panel.
3. **Variables**
   Displays the Dialog Test Variables panel.
4. **Tables**
   Displays Dialog Test Tables panel.

5. **Log**
   Displays the ISPF Transaction Log panel.
6. **Services**
   Displays the Invoke Dialog Service panel.
7. **Traces**
   Displays the Dialog Test Traces panel.
8. **Break Points**
   Displays the Dialog Test Breakpoints panel.
9. **Dialog Test**
   Displays the Dialog Test Primary Option panel.
10. **Dialog Test appl ID**
    Displays the Dialog Test Application ID panel.

Help
The Help pull-down provides general information about the main options available during an Edit session as well as information about edit line commands and primary commands.

Editing the data set

When the editor displays existing data, each line consists of a 6-column line command field followed by a 72-column data field. The line command fields contain the first 6 digits of the sequence numbers in the data. If the data has no sequence numbers, the line command fields contain relative numbers that start at 1 and are incremented by 1.

Based on your action, the ISPF editor places the cursor in the most useful position. To help you find the cursor, the editor intensifies the line command field that contains the cursor.

If the data contains characters that cannot be displayed, blanks replace those characters on the panel but not in the data. You cannot type over the blanks. You can display and edit undisplayable characters by entering hexadecimal mode or by using the FIND and CHANGE commands with hexadecimal strings. See "HEX—Display Hexadecimal Characters" on page 238 for information on entering hexadecimal mode.

Printer control characters, if present, are displayed and are treated as part of the data. ASA control characters are alphanumeric and you can edit them. Machine control characters, however, cannot be displayed and are replaced on the panel with blanks.

When you are editing existing data, the selected member or sequential data set is read into virtual storage, where it is updated during edit operations. Use of virtual storage for editing work space results in high performance, but might require a large user region. If you use all available storage, an ABEND occurs, and you lose the work space unless recovery mode is on.

Using the ISPF editor basic functions

The basic functions of the ISPF editor are simple and can be used immediately:

- To alter data, type over the existing material or use the Ins (Insert) and Del (Delete) keys to add or remove characters.
- To view data that is not displayed, use the scroll commands. These are PDF default values:

Fn key
Action

F7/F19
Scrolls up

F10/F22
Scrolls left

F8/F20
Scrolls down

F11/F23
Scrolls right

- To insert a line between existing lines, type I over a number in the line command field and press Enter. The line command field is the 6-column row displayed on the left side of the panel when you create or edit a data set. The new line is inserted after the one on which you typed the I.

  Note: The editor does not distinguish between input mode and edit mode. Use the I or TE line commands to insert new lines, either between existing lines or at the end of the data.
- To delete a line, type D over the number to the left and press Enter.
- To save your work and leave the editor, type END on the command line and press Enter.

Ending an edit session

Usually, you complete your editing session with the END command and, based on the values in your edit profile, PDF performs these tasks:

- If autosave mode is on and you have made changes to the data:
  - If both number mode and autonum mode are on, the data is renumbered. If not, the numbers remain unchanged.
  - The data is automatically saved. Special temporary lines, such as =PROF>, =MASK>, ==ERR>, ==CHG>, =BNDS>, =TABS>, ==MSG>, =NOTE=, =COLS>, and ====== are not part of the data and are not saved. However, you can convert =COLS>, ==MSG>, =NOTE=, and ====== lines to data lines and save them as part of the data set by using the MD (make dataline) line command before entering END.
  - If STATS mode is on and the data is a member of an ISPF library or other partitioned data set, the statistics are either generated or updated, depending on whether statistics were previously maintained for the member. If the member is an alias, the alias indicator is turned off.
  - If autolist mode is on, a source listing of the data is recorded in the ISPF list data set for eventual printing.
- If autosave mode is off with the PROMPT operand, a prompting message is displayed. You can issue SAVE to save the data or CANCEL to end the edit session without saving the data.
- If autosave mode is off with the NOPROMPT operand, the data is not saved. The result is the same as that which occurs if you enter a CANCEL command. (You can opt to confirm cancellations by selecting that option from the Primary Edit panel action bar Confirm choice.)
- PDF returns to the previous panel, which is either a member list or the Edit Entry panel.
- When END is issued from a macro, the edit session does not complete until the macro terminates all processing. For example, when a Rexx macro executes the EXIT statement or a COBOL language program executes STOP RUN.

You can end editing without saving by using CANCEL.

By default, the editor truncates variable-length lines by removing blanks at the end of each line during a save. If a variable-length line is completely blank and has no line number, a blank is added so that the line length is not zero.

If you select "Preserve VB record length" on the edit entry panel, or specify PRESERVE on the edit service, the editor stores the original length of each record in variable-length data sets and when a record is saved, the original record length is used as the minimum length for the record. The minimum line length can be changed by using the SAVE_LENGTH edit macro command. The editor always includes a blank at the end of a line if the length of the record is zero.

Because VIEW is a special type of edit session, it is important to note that the use of the REPLACE or CREATE commands from within VIEW always honors the setting of the "Preserve VB record length" option on the edit entry panel. This setting can be overridden by using the PRESERVE primary command.

Attention: CANCEL cancels all changes made since the beginning of the edit session or the last SAVE command, whichever is most recent.

The RETURN command is logically equivalent to the repeated use of the END command. PDF performs the same actions at the end of the edit session.

When a space ABEND such as D37 occurs, ISPF deallocates the data set so that you can swap to another screen or user ID and reallocate the data set. This does not occur for data sets that were edited using the DDNAME parameter of the EDIT service.

Edit commands

You can use two kinds of commands to control editing operations: line commands and primary commands.

Line commands

Line commands affect only a single line or block of lines. You enter line commands by typing them in the line command field on one or more lines and pressing Enter. The line command field is usually represented by a column of 6-digit numbers on the far left side of your display. When you are editing an empty data set or member, however, the line command field contains quotes. This field can also be used to define labels and to display flags that indicate special lines, such as the =NOTE= flag, which indicates a note line.

You can use line commands to:

- Insert or delete lines
- Repeat lines
- Rearrange lines or overlay portions of lines
- Simplify text entry and formatting
- Define an input mask
- Shift data
- Include or exclude lines from the display
- Control tabs and boundaries for editing
- Convert some types of special temporary lines to data lines

You can enter edit line commands as primary commands on the command line by prefixing them with a colon (:) and placing the cursor on the target line. For example, if you enter **:D3** on the command line and move your cursor to line 12 of the file, the three lines 12, 13, and 14 are deleted from the file. This technique is normally used for PF key assignments.

See Chapter 3, “Managing data,” on page 41 for ways you can use line commands to manipulate data and Chapter 9, “Edit line commands,” on page 135 for the line command syntax.

Primary commands

Primary commands affect the entire data set being edited. You enter primary commands by typing them on the command line (Command ===>), usually located on line 2, and pressing Enter. Any command entered on the edit command line is first intercepted by ISPF. If the command entered is an Edit Primary Command or an Edit Macro, PDF processes the command.

You can use primary commands to:

- Control your editing environment
- Find a specific line
- Find and change a character string
- Combine several members into one
- Split a member into two or more members
- Submit data to the job stream
- Save the edited data or cancel without saving
- Sort data

- Delete lines
- Access dialog element models
- Run an edit macro

If you have a primary command that is too long for the input field in the command line the ISPF command ZEXPAND can be used to display a popup window with the input field expanded to a length of 255 characters. The long primary command can then be entered in this expanded input field. After you exit the popup window and return to the data display press Enter to have the editor process the command. This popup window is only for the input of edit primary commands. To input other commands (for example TSO commands) that are too long for the command field, use the CMDE command.

Note:

- A long editor command entered in the popup window is truncated at the length of the edit panel command field when saved in the command retrieve stack.
- The support for an expandable command field is enabled for the IBM-supplied edit panels ISREDDE2, ISREDDE3, ISREDDE4, ISREDDE5, and FLMEDDE. The LEFT and RIGHT commands cannot be used to scroll data in the command field.

You can prefix any primary command with an ampersand to keep the command displayed on the command line after the command has processed. This technique allows you to repeat similar commands without retyping the command. For example, if you type:

```
&CHANGE ALL ABCD 1234
```

the command is displayed after the change has been made, which allows you then to change the operands and issue another CHANGE command. You can recall previous commands with the ISPF RETRIEVE command.

See Chapter 3, “Managing data,” on page 41 for some of the ways you can use primary commands to manipulate data and Chapter 10, “Edit primary commands,” on page 193 for the primary command syntax.

Edit commands and PF key processing

In the Edit function there are some differences between the way ISPF processes commands when they are entered from the command line as compared to when they are entered by a combination of the command line and a function (PF) key. In most applications, when you press a PF key, ISPF concatenates the contents of the command line to the definition of the function key. The result is handled as a single command by ISPF or by the application.

When you use a PF key defined as a scroll command (UP, DOWN, LEFT, or RIGHT) the system processes the command as follows:

- If the concatenation of the scroll command PF key definition and the contents of the command line does not create a valid scroll command:
  - If the word after the scroll command PF key definition begins with a numeric character (0-9), you get a message telling you the scroll amount was not valid.
  - Otherwise, edit processes the contents of the command line as an edit command, then processes the scroll command using the default scroll amount.

    In this case, the processing of the command line contents as an edit command bypasses the command table, because the command table is used to resolve the scroll key.
- If the concatenation of the scroll command PF key definition and the contents of the command line does create a valid scroll command edit scrolls the screen the specified amount.

If you manually type a scroll command on the command line (you do not use any PF keys) and it has an operand, the operand is checked for validity. However, in the case of a scroll operand that is not valid, the operand is not processed as a separate edit command as it is when used with a PF key.

When you use a PF key defined as RFIND or RCHANGE, first the command line is processed and then the PF key is actioned. For example, if you type a Find command then press PF5, the new find string is passed to RFIND:

Table 1. Examples of passing a string to RFIND

Command	Action	Result
F STR1	press Enter	Edit finds the next occurrence of STR1
F STR2	press PF5	RFIND finds the next occurrence of STR2

If you type C STR1 STR2 and press Enter to change STR1 to STR2, then on the command line type F STR3 and press the RCHANGE key, this results in the command C STR3 STR2 being run:

Table 2. An example of passing string values to RCHANGE

Command	Action	Result
C STR1 STR2	press Enter	Edit changes the next occurrence of STR1 to STR2
F STR3	press PF6	RCHANGE changes the next occurrence of STR3 to STR2

You can change this behavior of RCHANGE by using the EDITSET command to set an option, Force ISRE776 if RCHANGE passed arguments. If this option is set, RCHANGE will treat anything that you type on the command line as an invalid parameter and will return an error message ISRE776.

Edit macros

Edit macros are primary commands or line commands that you write. You can save time and keystrokes by using macros to perform often-repeated tasks.

Primary command macros

To run a primary command macro, type its name and any operands on the command line, and press Enter. Your installation may have written and documented common macros for your use. Of course, you can also write your own edit macros.

The rules for running a specific macro, and the expected results, depend on the particular macro. Your installation is responsible for documenting these rules and results. If you want to write your own macros, read Part 2, “Edit macros,” on page 77 and Chapter 11, “Edit macro commands and assignment statements,” on page 295.

ISPF enables the installer of the program to specify an edit macro that runs for all users. If a macro name is specified in the ISPF configuration table, then that macro runs before any macros specified in the users' profiles, in programs that invoke edit, or on the edit entry panels.

The site-wide macro can be used to alter existing profiles, enforce site-wide standards, track edit usage, deny edit and view of a data set member, or for any other purposes for which edit macros are designed. Site-wide macros normally end with a return code of 1 (one) in order to place the cursor on the command line. Site-wide macros must be available to each user in the appropriate data set concatenation (SYSPROC, STEPLIB, and so forth) or in Linklist or LPA (program macros only).

Users can also set an application-wide macro if they choose. See “Application-wide macros” on page 25 for more information.

The effect of running a macro depends on the implementation of the macro. Results such as cursor positioning, output messages, and so on, may or may not conform to the results that you expect from built-in edit commands.

Line command macros

You can define a table of user line commands and associated user macros using the ISPF table editor.

To run a user line command, type its name over the 6-digit number in the line command field and press Enter (in the same way as for any other line command). ISPF then invokes the associated user macro.

If you want to write your own line command to invoke a specific macro, see "Working with an edit line command table" on page 84.

Editing data in SCLM-controlled libraries

For information about editing libraries that are controlled under SCLM, refer to *z/OS ISPF Software Configuration and Library Manager Guide and Reference*.

Packing data

Data can be saved in either packed or standard format. You can control the format by using the PACK primary command to change the edit profile. The editor reads the data in and you can edit it the way you normally would. When you end the editing session, the data is packed and stored. See "PACK—Compress Data" on page 262 and "PACK—Set or Query Pack Mode" on page 379 for more information.

The packed data format has the advantage of saving space. It allows for a more efficient use of DASD by replacing repeating characters with a sequence that shows the repetition.

There are two disadvantages:

- The space saving is at the expense of additional processing when the data is read or written.
- The data cannot be directly accessed by programs. You must access the data through PDF dialogs and library access services. You would not, for example, pack an executable such as a CLIST or REXX exec. A packed CLIST or REXX exec would not run, because pack mode analysis is not done before the member is passed to the system for execution.

Specifying z/OS UNIX pathnames with edit primary and macro commands

These edit primary and macro commands support the specification of a z/OS UNIX pathname as an operand:

- COMPARE
- COPY
- CREATE
- MOVE
- REPLACE

You can specify a pathname in the format accepted as input in the "Other Partitioned, Sequential or VSAM Data Set, or z/OS UNIX file" data set name field. If you are editing a z/OS UNIX file when these commands are used, you can specify a + (plus) as the first character of the pathname to represent the pathname of the directory containing the file being edited. For example, if you are currently editing the file /u/usr1/prog1, the command `copy +/src1` copies in the data in file /u/usr1/src1.

Chapter 2. Controlling the edit environment

This chapter describes the editing environment and how you can customize that environment to best suit your needs.

ISPF defaults control much of the editing environment. However, you can use line and primary commands to change number and statistical fields on a data display panel and to determine how the data appears.

What is an edit profile?

An edit profile controls your edit session through modes and temporary lines. These modes and lines convert data to uppercase (caps mode), automatically renumber lines of data (autonum mode), or specify the left and right boundaries used by other commands (=BNDS> line).

The library type (the last of the data set name qualifiers), record format (fixed or variable), or the record length can implicitly specify an edit profile. You can choose an edit profile in three ways:

- Issue the PROFILE command with a profile name as parameter
- Fill in the Profile field on the Edit Entry panel
- Supply a PROFILE keyword and name when calling the EDIT service, such as:

```
ISPEXEC EDIT PROFILE(name) ...
```

Using edit profile types

Different kinds of data can have different edit profiles. For example, you could set up one edit profile for COBOL programs, another edit profile for memos, and a third edit profile for test data. Your installation determines how many different edit profiles are available to you. Typically, 25 edit profiles are available.

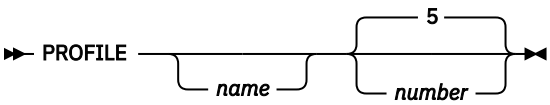
If you attempt to create more edit profiles than defined by your installation, the least-used edit profile is deleted first. Locked edit profiles are not deleted unless all your edit profiles are locked. In that case, the least-used locked edit profile is deleted first. Again, if you continue to add edit profiles, all of the unlocked edit profiles are deleted before locked edit profiles.

You can control the use of profiles from the Edit Entry panel. If you leave the Profile Name field blank, the profile name defaults to the data set type, which is the last qualifier in the data set name. If you type a profile name, it overrides the data set type qualifier. In either case, if a profile of that name currently exists, it is used. If it does not exist, a new profile is defined. The initial contents of the new profile include the default mode settings, all-blank mask and tabs, and default bounds. To eliminate the profile lines from your panel, use the RESET command.

When editing a z/OS UNIX file, if the file name has a suffix then the first 8 characters of the suffix are used to identify the edit profile (any lowercase characters in the suffix are converted to uppercase). If the file name does not have a suffix the profile name defaults to HFSPROF.

Displaying or defining an edit profile

You can display none, all, or part of an edit profile by entering the PROFILE command using this syntax:



where *name* is the name of the edit profile that you want to display and *number* is a number from 0 to 9.

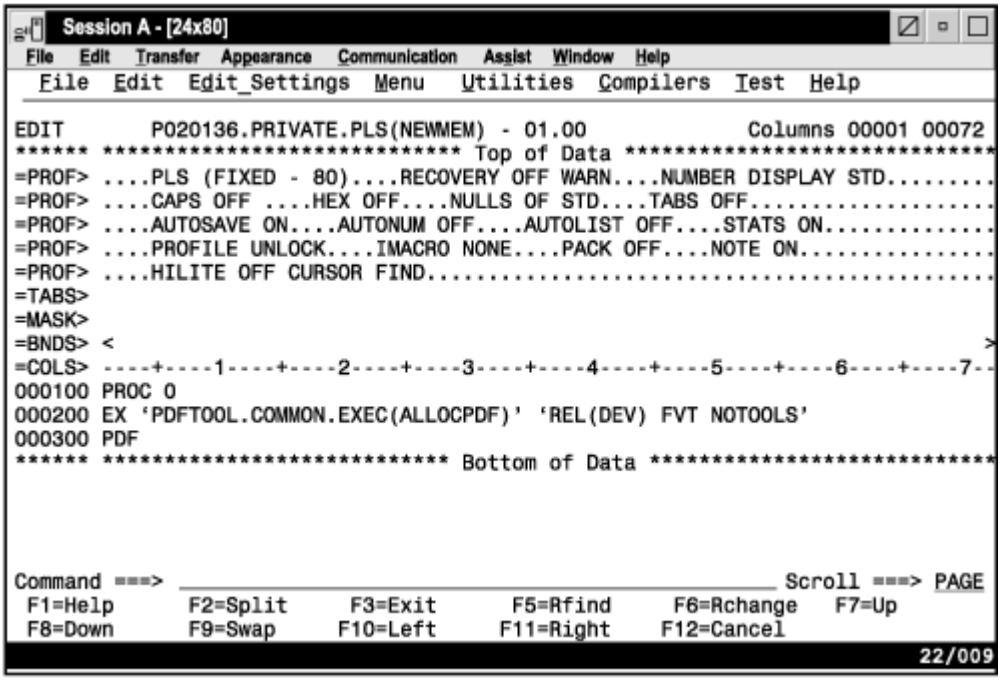


Figure 5. Edit Profile display (ISREDDE2)

Note: See "Primary Edit panel action bar choices" on page 10 for information on the action bar choices on this panel.

The first five lines of the edit profile (Figure 5 on page 18) are the current mode settings. The remaining lines are the current contents of the =TABS>, =MASK>, and =BNDS> lines, with the =COLS> positioning line. When no operands are entered, the first five lines, which contain the =PROF> flags, are always displayed. However, the =MASK> and =TABS> lines do not appear if they contain all blanks. If the =MASK> and =TABS> lines do contain data, they are displayed, followed by the =COLS> line.

The =BNDS> line does not appear if it contains the default boundary positions. It does appear when the bounds are set to something other than the default, and no 'number' parameter is entered into the PROFILE command.

Note: If enhanced edit coloring is not enabled for the edit session, the profile line displaying HILITE status is not shown. If highlighting is available, and if you explicitly set the language, then the language appears in RED on color terminals.

If you include the name of an existing profile, the editor immediately switches to the specified profile and displays it.

If you include a new profile name, the editor defines a profile using the current modes, options and temporary lines.

The number operand controls the number of lines shown in the profile display. If you type the number 0, the profile is not displayed. If you type a number from 1 through 8, that number of lines of the profile is displayed. If you type the number 9, the complete profile is displayed, even if the =MASK> and =TABS> lines are blank and the =BNDS> line contains the defaults. Because masks are ignored when using a format name, the "=MASK>" line is not displayed by the profile command in formatted edit sessions.

Modifying an edit profile

You modify an edit profile by entering commands to set various modes, options, and temporary lines. Whenever you change an edit profile value, ISPF saves the value (unless the edit profile is locked). The next time you edit data using the edit profile, the data is retrieved and the environment is set up again. This is easier than it sounds. First, there are defaults for all the modes, and, in most cases, you do not need to change them. Second, if you decide that you want to change a mode, you just enter the

appropriate command. The edit profile is automatically changed and saved. See “Edit modes” on page 19 for more information about edit modes.

Locking an edit profile

Once you have an edit profile exactly the way you want it, you can lock it. To do this, type `PROFILE LOCK` and press Enter. The edit profile is saved with all the current modes, options, and temporary lines, and it is marked so that the saved copy of the edit profile is not changed. Usually, each time you begin an editing session the edit profile you start with is exactly the way you locked it. The exceptions are caps, number, stats, and pack, which are made to match the data and are noted with messages. You can change a mode during an editing session, but if the edit profile is locked, the change affects only the current session; it does not affect any later sessions.

If you have locked your current edit profile, you cannot change the initial macro name with IMACRO. For information on IMACRO, see “IMACRO—Specify an Initial Macro” on page 246. For information on the LOCK operand, see “PROFILE—Control and Display Your Profile” on page 264.

Edit modes

The edit modes control how your edit session operates. To set these modes, use the associated primary commands. For example, if you are editing a COBOL program that is in uppercase and you want all your input to be converted to uppercase, set caps mode on by entering CAPS ON.

The list shown here summarizes the primary commands you use to display and change your edit profile. See Chapter 10, “Edit primary commands,” on page 193 for a complete description and for the operands you can type with the commands.

PROFILE
Displays the current setting of each mode in this list and controls whether changes to these settings are saved.

AUTOLIST
Controls whether a copy of the saved data is automatically stored in the ISPF list data set.

AUTONUM
Controls whether lines of data are automatically renumbered when the data is saved.

AUTOSAVE
Controls whether data is saved when you enter END.

CAPS
Controls whether alphabetic characters are stored in uppercase when the data is saved.

HEX
Controls whether data is displayed in hexadecimal format.

HILITE
Controls the use of enhanced edit color.

IMACRO
Names an edit macro used at the start of the edit session.

NOTES
Controls whether tutorial notes are included in an Edit model.

NULLS
Controls whether blank spaces at the end of a line are written to the panel as blanks or nulls. The difference is that nulls allow you to insert data; blanks do not.

NUMBER
Controls the generation of sequence numbers in a data set.

PACK
Controls whether ISPF packs (compresses) the data when it is saved.

RECOVERY
Controls the recovery of an edit session following a system failure.

SETUNDO
Controls the method of saving changes for the UNDO command.

STATS
Controls whether statistics for a data set are generated.

TABS
Controls tab settings for aligning data.

Edit profile modes

The data you edit controls four special edit profile modes. These modes are set when data is first edited or new data is copied in.

Caps mode
The editor sets caps mode on if it detects that a member to be edited contains no lowercase characters and sets caps mode off if the member does contain lowercase characters.

Number mode
The editor sets number mode on and changes number options if it detects that the data contains valid sequence numbers. It sets number mode off if the data does not contain valid sequence numbers.

Pack mode
The editor sets pack mode on if the data being edited was previously saved in packed format and sets pack mode off if the data was not previously saved in packed format.

Stats mode
The editor sets STATS mode on if the member being edited currently has ISPF statistics and sets STATS mode off if the member did not previously have ISPF statistics.

The ISPF editor changes the special data modes even if the original edit profile of the member edit profile is locked. However, for locked profiles, it does not save the changes to the profile.

For your convenience, the editor changes the special data modes automatically to correspond to the data. This allows you to use the default edit profile with a single data set, even though some members may contain programs (CAPS ON) while other members contain text (CAPS OFF). Some of the members may have statistics to be maintained, while other members are stored without statistics. Some members may be in packed data format, while others are in standard data format. Finally, some members may have sequence numbers while others do not.

When the editor changes your edit profile to correspond to the data, special message lines appear. If you want to override the change, enter the appropriate command. For example, if the editor changes caps mode from on to off because it finds lowercase characters in the data, type `CAPS  ON` and press Enter to reset it.

If you have special requirements, you might not want the editor to change the special modes. You may want to have caps mode on, even if the data contains lowercase data, or you may want to generate statistics on output, regardless of whether the member originally had statistics. If so, you can write an initial macro to specify how the editor is to run these special modes. You would then use IMACRO to associate the initial macro with the edit profile. See “Initial macros” on page 24 for more information on initial macros.

Edit mode defaults

ISPF saves several different edit modes in an edit profile. The user can specify the desired edit profile on the Edit Entry Panel. If the Profile field is left blank, the data set type is used as the profile name.

To preinitialize a set of edit profiles for first-time users, do these steps:

1. Start ISPF.
2. Select the Edit option.
3. Set the edit profile with the defaults you chose.

For example, to set "COBOL FIXED 80" in your profile, edit a member of a partitioned data set that has a RECFM of F or FB, a LRECL of 80, and a type qualifier of COBOL (or enter COBOL as the profile name on the Edit Entry Panel).

ISPF provides two methods for setting defaults for new edit profiles. You can set up a profile called ZDEFAULT in the ISPTLIB concatenation, or you can modify the edit profile defaults in the ISPF configuration table. The ISPF configuration table method is recommended because it is easier to maintain than the ZDEFAULT method. The ZDEFAULT method can still be used by individual users.

Site-wide Edit Profile Initialization

When no ZDEFAULT profile exists in the ISPTLIB concatenation and the user has no edit profile member in the ISPPROF concatenation, new edit profiles are created based on the settings in the ISPF configuration table.

Attention: Be very careful if you override the IMACRO setting. When a setting is forced the editor WILL CHANGE the users' profiles. For this reason it is usually better to use the site-wide initial macro than to force the initial macro in each user's profile.

Using the configuration table, you can change any of the defaults for new edit profiles and you can override (force) settings for PACK, RECOVERY, SETUNDO, STATS, and IMACRO in existing profiles.

It is helpful to understand when the ZDEFAULT profile is used and where it exists in a user's concatenations. The ZEDFAULT profile exists as a row of the edit profile table named *xxx*EDIT, where *xxx* is the application profile.

If ZDEFAULT exists in the edit profile table in the ISPTLIB concatenation, and the user has NO edit profile table in the ISPPROF allocation, the ZDEFAULT profile is copied from ISPTLIB into the user's edit profile when the user's edit profile is created. Therefore, many of your existing users might already have a ZDEFAULT profile in their edit profile. Individual users can delete their ZDEFAULT profiles using the PROFILE RESET command from within an edit session. Doing so allows them to use the site-wide configuration for new profiles. You can also use a site-wide edit initial macro to issue a PROFILE RESET for all users. ISPF does not ship any edit profiles.

Note: If you use the force settings such as PACK OFF, edit macro commands that attempt to change forced settings will not receive a failing return code, but the settings will not change.

Creating a ZDEFAULT edit profile

Set up a special edit profile named ZDEFAULT (enter ZDEFAULT as the profile name on the Edit Entry Panel). The ZDEFAULT profile is the one used for the initial settings whenever a new edit profile is generated, regardless of the RECFM and LRECL values. For example, if you do not have an ASM profile and you edit an ASM data set, an ASM profile is generated using ZDEFAULT for the initial settings. If no ZDEFAULT profile exists, one is automatically generated with settings obtained from the ISPF Configuration Table. This list shows an example:

Modes set on:
CAPS STATS NULLS NUMBER AUTOSAVE NOTE

Modes set off:
RECOVERY HEX TABS AUTONUM AUTOLIST PACK

Profile set to:
UNLOCK

IMACRO set to:
NONE

SETUNDO set to:
STG

HILITE set to:
DEFAULT

The number of profiles you can establish also is described in the configuration table. See “Displaying or defining an edit profile” on page 17 for more details. When you finish, exit ISPF. Your entire set of edit profiles is saved in your profile library (referenced by ddname ISPPROF) as the ISREDIT member.

Flagged lines

Flagged lines are lines that contain highlighted flags in the line command field. These lines can be divided into these categories:

- Changed lines
- Error lines
- Special lines

The flags in the line command field are not saved when you end an edit session.

Changed lines

==CHG>
Shows lines that were changed by a CHANGE or RCHANGE command.

Error lines

==ERR>
Shows lines in which ISPF finds an error when you enter a line command, primary command, or macro command. For example, when you enter a CHANGE command, and there is not enough room on the line to make the change.

Special lines

Special lines can be divided into two categories:

- Edit profile lines. The values associated with these lines are stored in your edit profile.

  =PROF>
  Contains the settings of the individual edit modes. This line is not saved as part of your data set or member. See “Edit modes” on page 19 for more information.

  =TABS>
  Defines tab positions. This line is not saved as part of your data set or member.

  =MASK>
  Can contain data to be inserted into your data set or member when you use the I (insert) line command. This line is not saved as part of your data set or member.

  =BNDS>
  Specifies left and right boundaries that are used by other commands. This line is not saved as part of your data set or member.

  =COLS>
  Identifies the columns in a line.

  The column identification line can be saved as part of the data set or member if you use the MD (make dataline) line command to convert it to a data line.

- Message lines, note lines, and information lines. These lines are not saved as part of the data set or member unless you use the MD (make dataline) line command to convert them to data lines.

  ==MSG>
  Message lines inform you of changes to the edit profile. These changes are caused by inconsistencies between the data to be edited and the edit profile settings. Message lines also warn you that the UNDO command is not available when edit recovery is off.

  You can insert message lines manually by using an edit macro that contains the LINE_AFTER and LINE_BEFORE assignment statements.

=NOTE=
Note lines display information when you insert edit models. However, these lines do not appear if the edit profile is set to NOTE OFF.

You can insert note lines manually by using an edit macro that contains the LINE_AFTER and LINE_BEFORE assignment statements.

======
Temporary information lines are lines you can add to provide temporary information that is not saved with the data. They can be inserted into an edit session by using an edit macro containing the LINE_AFTER and LINE_BEFORE assignment statements.

Edit boundaries

Boundary settings control which data in a member or data set is affected by other line, primary, and macro commands. You can change the boundary settings by using the BOUNDS line command, primary command, or macro command. Here are the commands that work within the column range specified by the current boundary setting:

Line commands

```
<  >  (  )  O  TE  TF  TS
```

Primary commands

```
CHANGE  EXCLUDE  FIND  LEFT  RCHANGE  RFIND  RIGHT  SORT
```

Macro commands

```
CHANGE   EXCLUDE  FIND     LEFT  RCHANGE  RFIND  RIGHT   SEEK  SHIFT <
SHIFT >  SHIFT (  SHIFT )  SORT  TENTER   TFLOW  TSPLIT  USER_STATE
```

This column range is in effect unless you specify overriding boundaries when entering a command. See the individual command descriptions for the effect the current bounds settings have.

If you do not explicitly set bounds, the editor uses the default bounds. These bounds change as the number mode changes. If you have changed the bounds settings for a data set and would like to revert to the default settings, you can use any BOUNDS command to do so. Table 3 on page 23 shows the default bounds settings for various types of data sets:

Table 3. Default bounds settings for data sets

RECFM	Data Set Type	Number Mode	BNDS When LRECL=80	BNDS Using Other LRECL
FIXED	ASM	ON STD	1, 71	1, LRECL-8
		OFF	1, 71	1, LRECL
	COBOL	OFF	1, 80	1, LRECL
		ON STD	1, 72	1, LRECL-8
		ON COBOL STD	7, 72	7, LRECL-8
		ON COBOL	7, 80	7, LRECL
	OTHER	ON STD	1, 72	1, LRECL-8
		OFF	1, 80	1, LRECL

Table 3. Default bounds settings for data sets (continued)

RECFM	Data Set Type	Number Mode	BNDS When LRECL=80	BNDS Using Other LRECL
VARIABLE	ALL	ON STD	9, record length	N/A
		OFF	1, record length	N/A

If the default boundaries are in effect, they are automatically adjusted whenever number mode is turned on or off. If you have changed the bounds from the default settings, they are not affected by the setting of number mode.

If a left or right scroll request would cause the display to be scrolled 'past' a left or right bound, the scrolling stops at the bound. A subsequent request then causes scrolling beyond the bound.

This scrolling feature is especially useful when you are working with data that has sequence numbers in the columns to the left. It allows left and right scrolling up to (but not past) the bounds so that the sequence numbers are normally excluded from the display.

If you specify an invalid value for either the left or right boundary when changing the current boundary settings, the editor resets the value for that boundary to the default. These constitute invalid boundary values:

- A right boundary value that is greater than the logical record length of a fixed-block file if the file is unnumbered.
- A right boundary value that is greater than the logical record length-8 of a fixed-block file if the file with standard numbers.
- A right boundary value that is greater than the logical record length-4 of a variable-block file.
- A left boundary value that is less than or equal to 8 for a variable-block file with standard numbers
- A left boundary value that is less than or equal to 6 for a file that is numbered with COBOL numbers

Initial macros

The editor runs an initial macro after the data is first read but before the data is displayed. An initial macro can be used to do tasks such as initializing empty data sets, defining program macros, and initializing function keys.

For example, if you want caps mode on even if the data contains lowercase data, create an initial macro with a CAPS ON command. The editor first reads the edit profile and the data, then it sets caps mode to correspond to the data. Next, it runs your initial macro, which overrides the edit profile setting of caps mode.

To store an initial macro name in the edit profile, use the IMACRO command:

```
IMACRO initmac
```

See “IMACRO—Specify an Initial Macro” on page 246 for more information on the IMACRO command.

To execute an initial macro for the current session, use one of these methods:

- Type the macro name in the INITIAL MACRO field on the Edit Entry panel:

  ```
  INITIAL MACRO ===> initmac
  ```

- Specify the initial macro name on the EDIT service call:

  ```
  ISPEXEC EDIT DATASET(dsname) MACRO(initmac) ...
  ```

Once the initial macro is stored in a profile, it runs at the start of each edit session that uses the profile. It can be overridden by an initial macro typed in the INITIAL MACRO field on the Edit Entry panel or

specified on the EDIT service call. You can type NONE in the INITIAL MACRO field to suppress the initial macro defined in the profile.

Note:

1. If the current profile is locked, the IMACRO command cannot be run.
2. Remember that commands referencing display values (DISPLAY_COLS, DISPLAY_LINES, DOWN, LEFT, RIGHT, UP, LOCATE) are invalid in an initial macro because no data has been displayed.
3. If the initial macro issues either an END or CANCEL command, the member is not displayed.

Application-wide macros

You can specify a macro to run at the beginning of your edit sessions by placing a variable called ZUSERMAC in either the shared or profile pool. ZUSERMAC must contain the name of the macro and cannot include any operands. ZUSERMAC must not be longer than 8 characters.

If ZUSERMAC exists in the profile or shared pool, the macro it specifies is run after the site-wide initial macro, and before the initial macro specified on the edit panel, on EDIT service command, or in the edit profile.

If you want to remove the user application-wide macro, you can issue the VERASE service to remove ZUSERMAC from the shared or profile pool.

Statistics for PDS members

If STATS mode is on, ISPF creates and maintains statistics for partitioned data set members. The following sections explain the effect STATS mode has on your statistics, first when you are beginning an edit session and then when you are saving data.

- “Effect of Stats mode when beginning an edit session” on page 25
- “Effect of Stats mode when saving data” on page 25

Note: Stats mode is ignored for sequential data sets.

Included in the statistics are version and modification levels. These numbers can be useful in controlling library members. See “Sequence number format and modification level” on page 26 for a discussion of how the generation of statistics affects the format of sequence numbers.

Effect of Stats mode when beginning an edit session

Whenever a member is retrieved for editing, the ISPF editor checks the setting of STATS mode. ISPF does not display any warning messages if the STATS mode and the member are consistent. For example:

- If the STATS mode is on and the member has statistics
- If the STATS mode is off and the member does not have statistics

If the STATS mode and the member are not consistent, however, ISPF displays a warning message. For example:

- If STATS mode is on and the member has no statistics, ISPF displays a warning message, but does not change the STATS mode.
- If STATS mode is off and the member has statistics, ISPF automatically turns on STATS mode and displays a message indicating the mode change.

Effect of Stats mode when saving data

If STATS mode is on when you save the member, ISPF updates the statistics, or creates statistics if the member did not previously have them.

If STATS mode is off when you save the member, ISPF does not store any statistics; any previous statistics are destroyed.

Stats mode is saved in the edit profile.

Version and modification level numbers

Two of the statistics that the editor creates and maintains for members of ISPF libraries and partitioned data sets (when STATS mode is on) are the version and modification level numbers. These numbers are displayed in the form `VV.MM` at the top of the edit panel following the data set name.

When the editor creates statistics for a new member, the default version and modification level numbers are 01 and 00, respectively. Otherwise, the values are taken from the previous statistics stored with the member.

You can change the version number with the VERSION command.

The modification level number appears in the last 2 digits of the line numbers for new or changed lines to provide a record of activity. The number is automatically incremented by one when the first change is made to the data. It can also be changed explicitly with the LEVEL command. The numbers for both can range from 00 to 99, inclusive. After the modification level number reaches 99, it does *not* increment by one to return to level 00.

The editor normally increments the modification level the first time that data is changed. This incrementing is suppressed if:

- You have set the modification level with a LEVEL command before making the first change.
- Statistics did not previously exist, and the editor has set the modification level to 0 for a new member.

If both STATS mode and standard sequence number mode are on, the current modification level replaces the last two positions of the sequence number for any lines that are changed. At the time the data is saved, it is also stored for any lines that already are marked with a modification level higher than the current modification level. If you type `LEVEL 0`, press Enter, and then save the data, all lines are reset to level 0. See "LEVEL—Specify the Modification Level Number" on page 247 for more information.

Sequence numbers

Each line on the panel represents one data record. You can generate and control the numbering of lines in your data with these commands:

AUTONUM
Automatically renumbers data whenever it is saved, preserving the modification level record.

NUMBER
Turns number mode on or off, and selects the format.

RENUM
Renumbers all lines, preserving the modification level number.

UNNUMBER
Turns off numbering and blanks the sequence number fields on all lines. This deletes all modification level records.

Sequence number format and modification level

Sequence numbers can be generated in the standard sequence field, the COBOL sequence field, or both:

- The *standard sequence field* is the last 8 characters for fixed-length records, or the first 8 characters for variable-length records, regardless of the programming language. Use NUMBER ON STD to generate sequence numbers in the standard sequence field.

  For members of partitioned data sets, the format of standard sequence numbers depends on whether statistics are being generated. If statistics are being generated, standard sequence numbers are 6 digits

followed by a 2-digit modification level number. The level number flag reflects the modification level of the member when the line was created or last changed. If, for example, a sequence number field contains `00040002`, the line was added or last changed at modification level 02. The sequence number is `000400`.

If STATS mode is off, or if you are editing a sequential data set, standard sequence numbers are 8 digits, right-justified within the field.

- The *COBOL sequence field* is always the first 6 characters of the data and is valid only for fixed-length records. Use the NUMBER ON COBOL or NUMBER ON STD COBOL to generate COBOL sequence numbers.

Attention:

If number mode is off, make sure the first 6 columns of your data set are blank before using either the NUMBER ON COBOL or NUMBER ON STD COBOL command. Otherwise, the data in these columns is replaced by the COBOL sequence numbers. If that happens and if edit recovery or SETUNDO is on, you can use the UNDO command to recover the data. Or, you can use CANCEL at any time to end the edit session without saving the data. COBOL sequence numbers are always 6 digits and are unaffected by the setting of STATS mode.

Sequence numbers usually start at 100 and are incremented by 100. When lines are inserted, the tens or units positions are used. If necessary, one or more succeeding lines are automatically renumbered to keep the sequence numbers in order.

Sequence number display

For numbered data, the line command field displayed to the left of each line duplicates the sequence number in the data. Normally, the editor automatically scrolls left or right to avoid showing the data columns that contain the sequence numbers. However, you can explicitly scroll left or right to display the sequence numbers. The DISPLAY operand of the NUMBER and RENUMBER commands also causes the editor to display the sequence numbers.

For example, assume that the data has COBOL numbers in columns 1 through 6 and the number mode is `NUMBER ON COBOL`. When the data is displayed, column 7 is the first column displayed. If you change number mode to `NUMBER OFF`, the data is scrolled so that column 1 is the first column displayed. If you then change number mode to `NUMBER ON`, the data is scrolled back to column 7. But if you change number mode to `NUMBER ON DISPLAY`, the sequence numbers in columns 1 through 6 remain displayed. The sequence numbers in columns 1 through 6 become part of the data window, but cannot be modified.

Initialization of number mode

When you retrieve data for editing, the editor determines whether it contains sequence numbers. The editor always examines the standard sequence field. It examines the COBOL sequence field if the data set type (the lowest level qualifier in the data set name) is COBOL.

If all lines contain numeric characters in either the standard or COBOL sequence field positions, or both, and if the numbers are in ascending order, the editor assumes the data is numbered and turns on number mode. Otherwise, the editor turns off number mode.

If the first setting of the number mode differs from the setting in the edit profile, a message indicating that the editor has changed the mode is displayed. For new members or empty sequential data sets, the first setting of number mode is determined by the current edit profile. For a new edit profile, the default is NUMBER ON for standard sequence fields, and NUMBER ON COBOL if the data set type is COBOL.

Enhanced and language-sensitive edit coloring

The editor provides language-sensitive coloring as a productivity aid for users who are editing program source. It is used in a variety of programming languages. Some coloring enhancements are also useful for editing data other than program source.

Note: Language-sensitive and enhanced coloring of the edit session is only available when enabled by the installer or the person who maintains the ISPF product. For information on enabling the enhanced color functions, see *z/OS ISPF Planning and Customizing*.

These enhancements allow programmers to immediately see simple programming errors, such as mismatched quotes or parentheses, unclosed comments, and mismatched logical constructs. The language-sensitive component allows you to take advantage of the editor's coloring capabilities for a number of programming languages simultaneously. Enhanced coloring is also a general productivity aid, because it improves your ability to locate text quickly.

The editor provides enhanced highlighting in these areas:

1. Programming language constructs, including:
   - Keywords for each individual language
   - Comments
   - Quoted strings (using both single and double quotes)
   - Compiler directives (C, COBOL, PL/I, and PASCAL only)
   - Special characters that the user chooses
2. Language-sensitive program logic features, such as logical blocks and IF/ELSE logic.
3. Any strings that match the previous FIND operation or that would be found by an RFIND or RCHANGE request.
4. Default color for the data area in non-program files.
5. The phrase containing the cursor in the data area.
6. Characters that have been input since the previous Enter or function key entry was pressed.

Note: Highlighting is *not* available for edit sessions that involve:

- Only CURSOR and FIND highlighting is valid for data sets with record lengths greater than 255
- Mixed mode edit sessions (normally used when editing DBCS data)
- Formatted data

Language support

These languages are supported for language-sensitive coloring:

- Assembler
- BookMaster®
- C
- COBOL
- HTML
- ISPF Dialog Tag Language (DTL)
- ISPF Panels (non-DTL)
- ISPF Skeletons
- JCL (Job Control Language)
- Pascal
- PL/I
- REXX
- SuperC Listing
- XML
- OTHER, which includes languages that use constructs similar to PL/I, such as DO, BEGIN, END, SELECT, and so forth. Limited support for CLIST is provided with the OTHER language. OTHER does not support any compiler directives.

Automatic language selection

If you choose not to set the language explicitly, the editor can *automatically* determine the language of the file being edited. The language is determined by looking at the first nonblank string in the file. In cases where ambiguity exists between languages, as in the case of C and JCL (where both may start with //) and in the case of PL/I and REXX (where both may start with a /* comment), the last qualifier of the data set name may be used to determine the language. The rules for automatic language recognition are as follows:

Assembler
Asterisk in column 1 or a recognized opcode of CSECT, DSECT, MACRO, TITLE, START or COPY.

Note: *PROCESS starting in column 1 is recognized as PL/I.

BookMaster
First character is . or : in column 1.

C
Any of these:

- First string is #
- First string is // and last qualifier of the data set name is not .CNTL, .JCL, or .ISPCTLx
- First string is /* and last qualifier of the data set name is .C

COBOL
First nonblank is a * or / in column 7.

HTML
First nonblank character is <, and the first tag in the file that is not a comment is either a <!DOCTYPE HTML> tag or a <?HTML> tag.

ISPF DTL
First nonblank character is <, and the file is not identified as an HTML or XML file.

ISPF Panel
First string is) in column 1, followed by a panel section name, or the first string is % in column 1.

ISPF Skeleton
) in column 1 in a file that does not seem to be a panel.

JCL
Any of these:

- //*anything* followed by the word COMMAND, DD, ELSE, ELSEIF, EXEC, IF, INCLUDE, JCLLIB, JOB, OUTPUT, PROC, SET, XMIT, or any word beginning with the characters 'MSG'
- //* in column 1
- // in column 1, and the last qualifier of the data set name is .CNTL, .JCL, or ISPCTLx
- Any of these starting in column 1:

  *$
  /*JOBPARM
  /*MESSAGE
  /*NETACCT
  /*NOTIFY
  /*OUTPUT
  /*PRIORITY
  /*ROUTE
  /*SETUP
  /*SIGNOFF
  /*SIGNON
  /*XEQ
  /*XMIT

Pascal
First string is (*, or the first string is /* and the last qualifier of the data set name is .PASCAL.

PL/I
First string is % or /* or the first string is *PROCESS starting in column 1. The use of carriage control characters in column one may cause PL/I detection to fail. When the last qualifier of the data set name starts with "PL", automatic language detection is retried ignoring column one if the first nonblank characters occur in column one, and no language can be detected. See REXX, C, and Panel for more information.

REXX
First string is a /* comment containing REXX, or the first string is a /* comment, and the last qualifier of the data set name is .EXEC or .REXX.

SuperC
Either of these starting in column 3 or 4:

- ISRSUPC -
- ASMFSUPC -

XML
First nonblank character is <, and the first tag in the file that is not a comment is either a <!DOCTYPE XML> tag or a <?XML> tag.

Other
First word is PROC, CONTROL, ISPEXEC, or ISREDIT.

HILITE AUTO selects a language based on the first nonblank line, and in some cases, the last qualifier of the data set name.

ISPF only scans up to the first 72 bytes in each line to determine the language. If the data that would identify the language is past the 72nd column, the language may be determined incorrectly.

Language processing limitations and idiosyncracies

Because ISPF does not provide true parsing, the built-in language scanner does not operate as a syntax checker. Keywords or built-in function names that are used as variables, and therefore not used in a language context, *will* be highlighted as keywords. For example, in context sensitive languages such as PL/I, the word 'ELSE' may be used as a variable name. ISPF treats 'ELSE' as a keyword in all cases, both for highlighting and logic determination.

In addition, the varying implementations and release schedules of the supported languages may result in keyword highlighting that does not reflect the latest version of the language.

Note: Nested comments are only supported when the language is REXX. When sequence numbers are in use, the editor only highlights the editable data. The sequence numbers are shown in the overtype color.

Also, because the language scanners of edit highlighting do not provide true parsing, when an unmatched end tag is encountered and the LOGIC option is enabled, subsequent end tags might be highlighted as unmatched, even if they appear to be properly matched.

Recognized special symbols

Special characters can be highlighted for each specific language. The characters are only highlighted if they are not part of another class of constructs such as a comment, a string, or a compiler directive. The default set of characters for each language follows:

Assembler
```
-+*/=<>&¬|:,
```

BookMaster
```
&.,!?$
```

C
```
-+*/=<>&¬|:!;|%?#[] \
```

COBOL
.

DTL
<>()=

HTML
<>()=

Panel
&

Skel
&?!<|>

JCL
(),|<>¬&=

Pascal
-+*/=<>&¬|:[]

PL/I
-+*/=<>&¬|:

REXX
-+*/=<>&¬|:%\

SuperC
None

XML
<>()[]=

Other
-+*/=<>&¬|:

These character sets may be changed by each user using the HILITE dialog.

Assembler

Highlighting is performed only in columns 1 through 72.

Specific keywords are not highlighted. Any word where an opcode would be expected is highlighted as a keyword.

BookMaster

Only BookMaster tags that begin with a colon (:) are highlighted. All tags should be terminated by a period, because ISPF highlights up to the next period. Dot control words (.*xx*) are never highlighted.

The keyword list supplied by the ISPF comprises the tags used to do logic matching (:*xxx*/:e*xxx*). Tags that have an optional end tag must have a matching end tag in the edited data for logical highlighting to work. The LOGIC option highlights unmatched end tags (:e*xxx* tags which do not have a corresponding :*xxx* tag) in reverse video pink.

BookMaster tags are not checked for validity. If you specify a colon (:) as a special character to highlight, the editor does not recognize BookMaster tags.

C

C++ comments (//) are recognized.

Logical highlighting highlights curly braces ({ and }).

Keywords are case-sensitive in C. Only the lowercase versions of keywords are highlighted.

COBOL

Highlighting is performed only in columns 7 through 72.

Both single quotes (') and double quotes (") are treated as unique open and close quote characters, although some COBOL languages only specifies double quotes as string delimiters. Compiler directives (also called compiler-directing statements) are supported for IBM SAA AD/Cycle COBOL/370 Version 1.1.

DTL, HTML, and XML

Only items in tags are highlighted. Any less than sign (<) is assumed to start a tag. This may cause highlighting errors if the '<' symbol appears outside of a tag.

Panels and skeletons

Quoted strings are terminated at the end of a line. For the most part, ISPF does not parse panels or skeletons. Usually any data on a line that starts with a ')' in column 1 is highlighted as a keyword.

JCL

Because automatic language determination recognizes C++ comments (//), JCL is recognized only if any of these conditions is met:

- The last qualifier of the data set name is JCL, CNTL, or PROCLIB or ISPCTLx (where x is any character)
- The second nonblank 'word' of the first nonblank line is DD, JOB, EXEC, or PROC
- The second nonblank 'word' of the first nonblank line starts with 'MSG'. This is for JCL with no JOB card, but with MSGLEVEL or MSGCLASS.
- The first three characters in the first nonblank line are //*.

Conditional JCL logic (IF/ELSE) is highlighted, but is not supported by the LOGIC option.

When the word DATA appears as the first word in a line or statement, HILITE assumes that this is a DD DATA statement and colors subsequent lines as in-stream data. To avoid this, ensure that DATA is not the first word on a line by placing other keywords before it. For example, instead of coding

```
//DCOBA2 PROC PROG=,
//   OPTCOB='DYN',
//   DATA='DATA(24)',
//   OUT='*',
//   USER='D0000',
```

move the operand starting with "DATA" to the same line as the previous operand:

```
//DCOBA2 PROC PROG=,
//   OPTCOB='DYN', DATA='DATA(24)',
//   OUT='*',
//   USER='D0000',
```

PL/I

For fixed-length record format data sets, column 1 is not scanned after the first nonblank line, except to search for *PROCESS statements.

REXX

Logic highlighting does not support a terminating semicolon in the IF expression, or a semicolon before the THEN or ELSE instructions.

In addition, IF statements which have the THEN keyword on the following line but do not have a continuation character at the end of the IF expression will cause highlighting errors.

For example, although these statements are valid in REXX, the ELSEs will be highlighted as a mismatched ELSEs:

```
     IF a=b; THEN say 'ok'; ELSE; say 'Not OK';
     IF a=b
          THEN say 'ok';
          ELSE say 'Not OK';
```

SuperC

Supports both ISPF SuperC (ISRSUPC) and High Level Assembler Tooklit SuperC (ASMFSUPC). Page, column, and section headings are used to determine the different sections within a SuperC listing.

Most forms of the SuperC listing are supported, including SuperC search-for and SuperC file, line, word, and byte compares. Both Wide and Narrow listings, with or without the printer control column, are supported.

SuperC SRCHFOR and SRCHFORC strings are highlighted (as FIND strings) within the source section of the listing. Other SRCHFOR and SRCHFORC statements parameters are processed and the ANYC process option is used for case insensitivity.

No specific action is taken with any other SuperC process option or process statement.

Other

When OTHER is in effect, ISPF tries to determine if the program is a CLIST by checking for a first word of PROC, CONTROL, ISPEXEC or ISREDIT. If ISPF determines that the data being edited is a CLIST, then CLIST comment closure and continuation rules apply.

The HILITE command and dialog

ISPF Edit supports enhanced and language-sensitive coloring through the HILITE command. The HILITE edit primary command is described in “HILITE—Enhanced Edit Coloring” on page 242. The HILITE edit macro command is described in “HILITE—Enhanced Edit Coloring” on page 345.

Note: The basic functions of HILITE cannot be accessed through a dialog that uses the GUI interface.

The HILITE dialog

The HILITE dialog is shown in Figure 6 on page 34. You can display this panel by entering the HILITE command with no operands from an edit panel, or by selecting Hilite from the Edit pull-down.

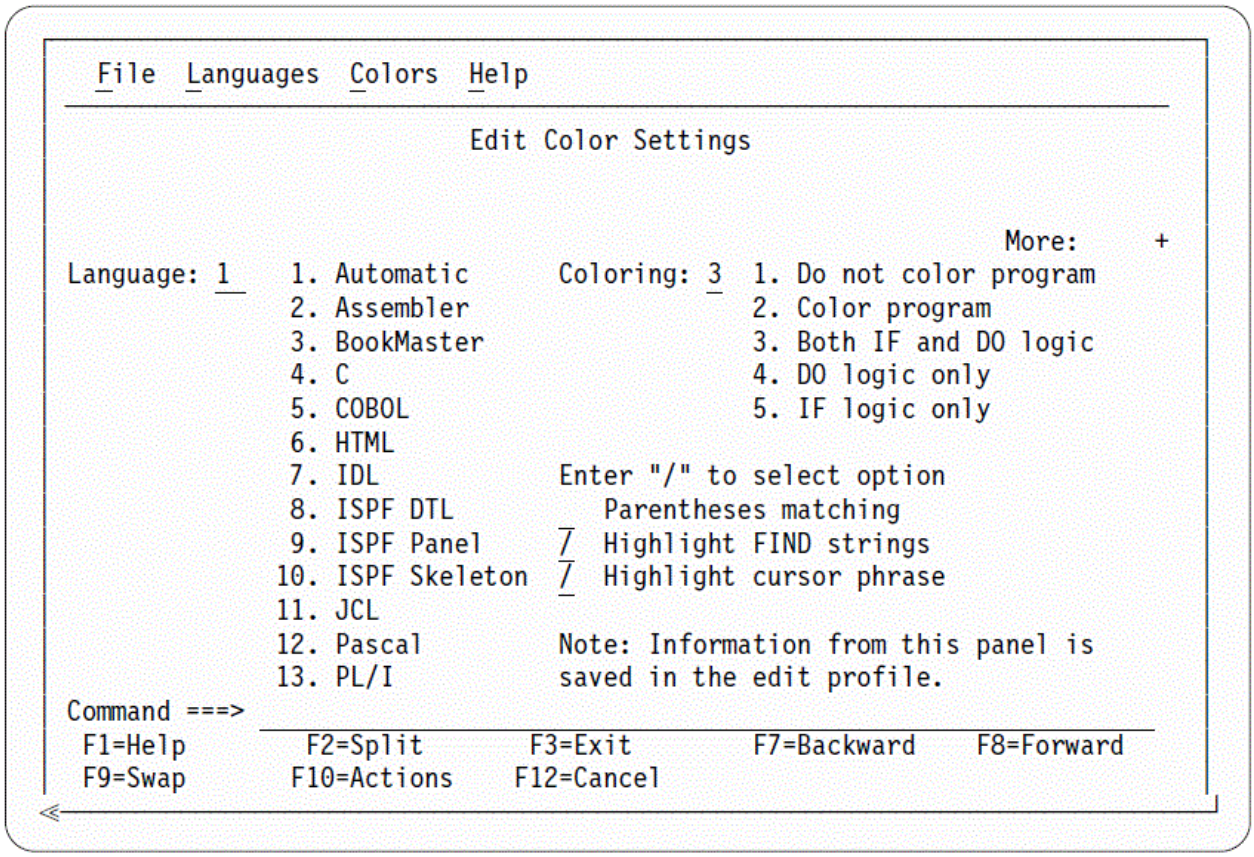


Figure 6. HILITE Initial Screen (ISREP1)

This dialog enables you to:

- Specify a language to be used for coloring, or enable automatic language detection.
- Assign colors for different language elements on a language-by-language basis or for all languages at once.
- Enable or disable logic or parenthesis matching.
- Turn FIND coloring on or off and assign the color for FIND highlighting.
- Turn cursor coloring on or off and assign the color for cursor phrase highlighting.
- Specify special symbols to be highlighted on a language-by-language basis.
- View keyword lists for each language.

  Note: Keyword lists and default highlighted symbols for each language are supplied with ISPF. IBM does not supply facilities for adding additional languages.

  However, it is possible to add or remove keywords. This facility involves assembling and link-editing an installation-modified keyword or symbol list. The keyword and symbol lists, and directions for changing them, are in member ISRPXASM in the IBM-supplied ISPF sample library.

HILITE initial panel action bar

Some of the functions of the HILITE dialog are provided through the action bar. The action bar choices on the HILITE Initial panel are:

File

Restart application
Resets all settings on all panels back to the point that HILITE was invoked.

Default All Settings
Resets all settings on this panel back to the point that HILITE was invoked.

Save and Exit
Saves changes and exits application.

Cancel
Ends application and discards changes.

Languages
This pull-down menu allows you to change the way that specific supported languages are highlighted, including the symbols that are highlighted and the colors that are used for the various language elements.

- Select a language to change the highlighting options for that language.
- Select All to change the highlighting options for all supported languages.
- Select Other to change the highlighting options for languages similar to PL/I.
- Select Default to specify the language to be used when AUTO is specified, but the language cannot be determined.

Colors

Overtype Color
Changes the color used for typed data.

Find String Color
Changes the color used to find strings.

Cursor Phrase Color
Changes the color of the phrase which contains the cursor.

Note: On a PC, the terminal emulator can affect the color. Some terminals do not support features such as "blink"; if this is selected with a color, another color might display.

Help
Immediately enters help panels, which include these choices:

- Overview
- HILITE command
- Supported Languages
- Automatic Language Determination
- Additional Functions
- Supported Comment Types
- FIND and CURSOR highlighting
- Logic Highlighting
- Notes relating to specific languages

Set Uvertype, Find String, Cursor Phrase Color action bars

These action bar choices function as follows:

File
The File pull-down offers these choices:

Reset
Resets the settings on this panel to the values they had when the panel first appeared.

Default
Sets the values to the IBM-supplied defaults.

Save and Exit
Exits this panel. Changes will be saved when the HILITE dialog completes, unless Cancel is specified.

Cancel
Exits this panel and discards changes.

Help
Immediately enters help panels for the HILITE command and dialog.

After selecting a specific language from the Languages pull-down on the HILITE Initial panel (Figure 6 on page 34), Figure 7 on page 36 is displayed:

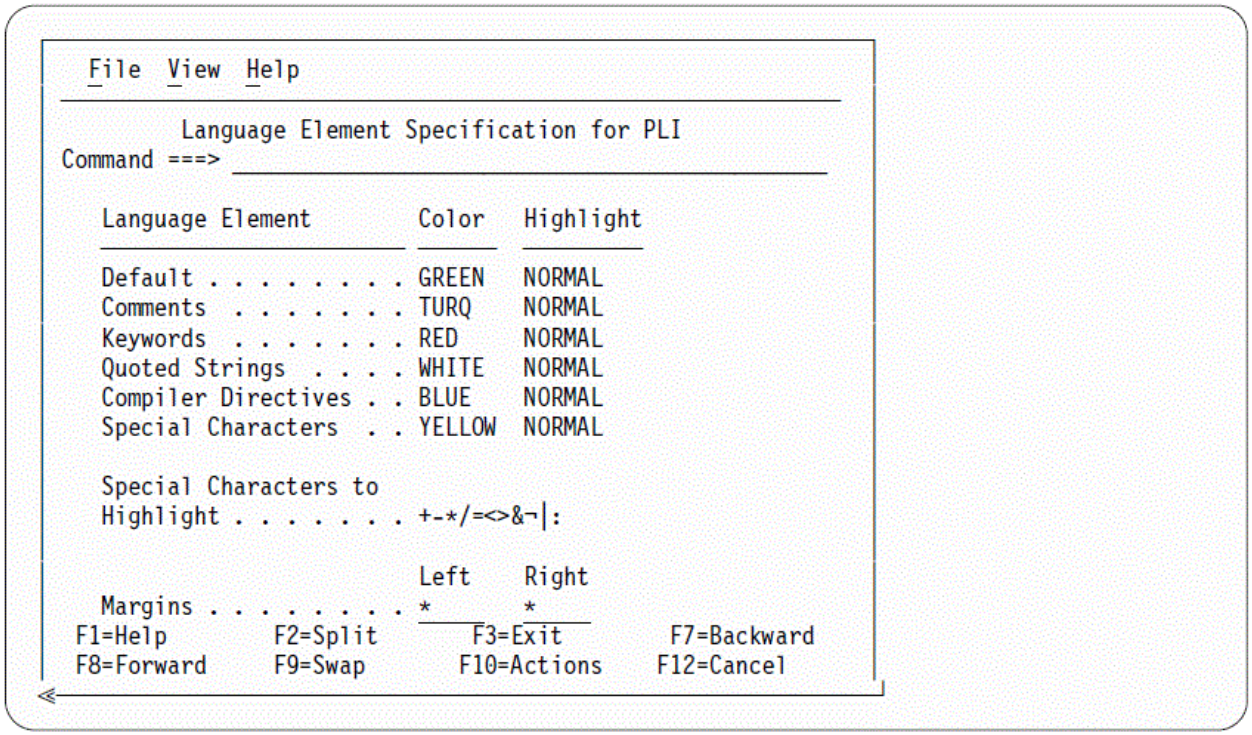


Figure 7. HILITE Language Element Specification Screen (ISREPC1)

Note:

1. If the selected language supports alternate margins (such as PL/I in Figure 7 on page 36), you can enter left and right boundaries in the Margins input field.
2. If the JCL language is selected, the Compiler Directives field in the pop-up window is replaced by a field named "DD * and Data Lines".
3. If a field is not applicable to a language, it is supplied with *n/a*.
4. When a new color is typed in, the input field is shown in that color when you press Enter.

Edit Color Settings action bar

The Edit Color Settings action bar choices function as follows:

File
The File pull-down offers these choices:

Restart '*language*'
Resets colors and symbols to the settings they had upon entry to this panel.

Defaults
Resets colors and symbols to default values.

Save and Exit
Exits this panel. Changes will be saved when the HILITE dialog completes, unless Cancel is specified.

Cancel
Exits this panel and discards changes.

View
The View pull-down choice is:

View Keywords
Displays a list of keywords for a particular language. See Figure 8 on page 37 for an example of a Language Keyword list.

Help
Immediately enters help panels.

If no keywords exist for a given language choice, a message is displayed instead of a Language Keyword list.

```
 File  Help
──────────────────────────────────────────────────────────────────────
                      Language Keyword List
Command ===> ____________________________________________________________

     Language: PLI         Number of keywords: 549
     (Includes preprocessor keywords)              More:     +
ABNORMAL            FOFL               PLISRTB
ABS                 FORMAT             PLISRTC
ACOS                FORTRAN            PLISRTD
ACOSF               FREE               PLITDLI
ADD                 FROM               POINTER
ADDBUFF             FROMALIEN          POINTERADD
ADDR                FS                 POINTERDIFF
ADDRDATA            GAMMA              POINTERSUBTRACT
ALIAS               GENERIC            POINTERVALUE
ALIGNED             GENKEY             POLY
ALL                 GET                POS
ALLOC               GETENV             POSITION
 F1=Help       F2=Split      F3=Exit       F7=Backward   F8=Forward
 F9=Swap      F10=Actions   F12=Cancel
```

Figure 8. HILITE Language Keyword List (ISREPK)

Language Keyword List action bar

The Language Keyword List action bar choices function as follows:

File
The File pull-down choice is:

Cancel
Exit this panel. (No changes are possible on this panel.)

Help
Immediately enters help panels.

Highlighting status and the edit profile

Colors are assigned to each character in the data area when the data appears. As you type in characters, they appear in the 'overtype' color. When Enter or a function key is pressed, the file is scanned again and the new characters are displayed in the appropriate colors for the type of data being edited. The actual color definitions and symbol sets for each language affect the entire ISPF session. However, only the language, coloring type (ON/OFF status), and logic type are saved in the edit profile.

The HILITE edit profile line shows the status of edit highlighting. Figure 9 on page 38 shows some examples. If edit highlighting is not available, the profile line is not shown. If highlighting is available, and

you explicitly set the language, the language appears in red. If you have customized the left and right margins, the values appear in red. If you have not customized the margins, the default values for the language are displayed.

```
....HILITE PLI LOGIC CURSOR FIND MARGINS(5,70).......................
or
....HILITE PLI LOGIC PAREN CURSOR FIND MARGINS(2,80).................
or
....HILITE COBOL CURSOR FIND.........................................
or
....HILITE OFF.......................................................
```

Figure 9. Examples of edit profile lines showing HILITE options

The information shown on the PROFILE command is saved in the edit profile.

Edit recovery

Edit recovery helps you to recover data that might otherwise be lost. For example, you would use edit recovery to re-establish the edit session at the point of failure after a power outage or system failure. Turning recovery mode on causes the data to be written to a temporary backup file. This is independent of whether changes have been made to the data.

You can turn on edit recovery mode by performing either of these actions:

- Entering the RECOVERY primary command:

```
RECOVERY ON
```

- Running an edit macro that contains the RECOVERY macro command:

```
ISREDIT RECOVERY ON
```

If recovery mode is on when a system crash occurs, automatic recovery takes place the next time you attempt to use edit. Recovery mode is remembered in your edit profile.

When you begin an edit session, if there is data to recover, the Edit Recovery panel appears, shown in Figure 10 on page 39.

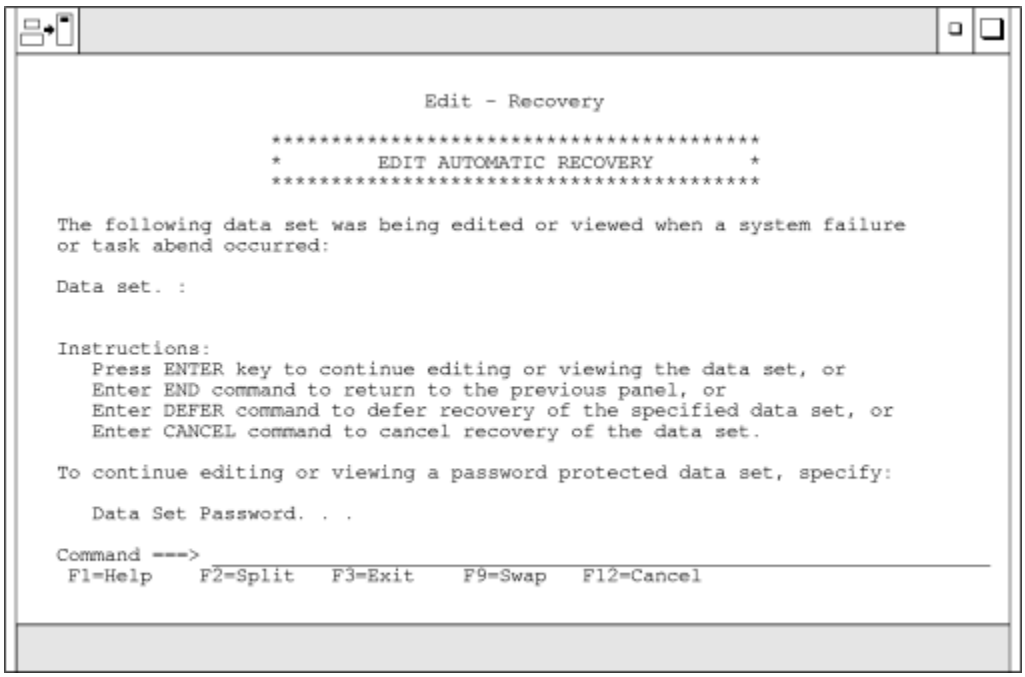


Figure 10. Edit Recovery panel (ISREDM02)

Note: For information about the Data Set Password field, refer to the topic about Libraries and Data Sets in *z/OS ISPF User's Guide Vol I*.

If you continue with, defer, or cancel recovery and you have other data to be recovered, the Edit Recovery panel is displayed again for the next data set. You can control the number of data sets to be recovered with the edit recovery table, a system data set that contains entries for each level of nested editing sessions that can be recovered. For information on changing edit recovery operands, refer to *z/OS ISPF Planning and Customizing*.

You may experience B37 (space) abends on the recovery data set if the guidelines in the *z/OS ISPF Planning and Customizing* have not been followed.

Note:

- You cannot recursively edit data while you are in an edit session that is the result of an edit recovery.
- Edit recovery is not supported when editing a generation other than the current generation (also known as generation zero) of a member of a PDSE version 2 data set that is configured for member generations.

Attention: If the data set to be recovered was edited by another user before you continue with edit recovery, the changes made by the other user are lost if you save the data.

If you press Enter to continue editing the data set, the editor runs a recovery macro if you had previously specified one by using the RMACRO primary or macro command. See "Recovery macros" on page 109 and the descriptions of the RMACRO primary and macro commands for more information.

Despite edit recovery's benefits in recovering data, there are times when you might not want to use it. You might want to turn edit recovery off in these situations:

- Operating with recovery mode off eliminates the I/O operations that maintain the recovery data and can therefore result in improved response time.
- Besides recording actual data changes, recovery mode records temporary changes, such as excluding lines and defining labels. These temporary changes are recorded to allow UNDO to undo other edit interactions besides those that change data. Therefore, when edit recovery is on, the recording of both data and temporary changes affects the amount of DASD space that is used.

You can turn off edit recovery mode by performing either of these actions:

- Entering the RECOVERY primary command:

```
RECOVERY OFF
```

- Running an edit macro that contains the RECOVERY macro command:

```
ISREDIT RECOVERY OFF
```

See Chapter 10, “Edit primary commands,” on page 193 for details on using RECOVERY.

Chapter 3. Managing data

This topic gets you started using some of the basic line and primary commands to manipulate data.

The basic functions of the ISPF editor are similar to those of a word processor. You can create, copy, move, search, and replace data, as well as perform several other word processing functions by using the line and primary commands described in this chapter .

Creating and replacing data

Use the CREATE command to add a new member to a partitioned data set, create a new sequential data set, or create a new z/OS UNIX file. Use the REPLACE command to rewrite a member, sequential data set, or z/OS UNIX file. The process of creating and replacing data is very similar. However, remember that when you replace data, the original data is deleted and replaced with the new data.

There are two ways you can use CREATE or REPLACE:

1. You can type either CREATE or REPLACE on the command line, followed by the name of a member, the name of a data set and member, the name of a sequential data set, or the name of a z/OS UNIX file to be created or replaced. You can add line labels that show the lines to be copied. If you omit the labels, you can use the C (copy) or M (move) line commands to specify which lines are to be copied or moved. Then press Enter. See “CREATE—Create Data” on page 217 and “REPLACE—Replace Data” on page 271 for the complete syntax of the commands.
2. If you omit the member name, data set name and member, sequential data set name, or z/OS UNIX file name, and just type CREATE or REPLACE and specify the lines to be used to create or replace the member, the editor displays a panel requesting the name of the member or data set you want created or replaced.

If you try to create or replace data that has inconsistent attributes (for example, replacing a sequential data set with a member of a partitioned data set), the editor displays a warning and gives you an opportunity to cancel the command:

```
                          EDIT - Confirm Replace

Data set attributes are inconsistent. Truncation may result in
the right-most portions of some records if replace is performed.

 "Target" data set attributes:
     Data set name. : USERID.PRIVATE.STUFF
     Record format. : VARIABLE
     Record length. : 133

 "Current" data set attributes:
     Data set name. : USERID.PRIVATE.EXEC(PGM1)
     Record format. : VARIABLE
     Record length. : 251

Press ENTER key to allow replace with truncation.
Enter END command to cancel replace.
```

Copying and moving data

While you are editing, you can copy or move another data set, member, or z/OS UNIX file into the current data by using the COPY or MOVE primary commands. The process of moving and copying data is very similar. However, remember that when you move data, the original information no longer exists in the member, data set, or file that it is being moved from.

When moving or copying large data sets, you can reduce the processing time significantly by specifying NUMBER OFF before the operation and NUMBER ON afterwards.

This topic explains how to use the COPY and MOVE primary commands. See “C—Copy Lines” on page 153 and “M—Move Lines” on page 167 for information about the line commands.

The two ways to perform a move or copy operation are:

- You can type either COPY or MOVE, followed by *name* and either AFTER *label* or BEFORE *label*, where *name* is the name of the member, data set, or z/OS file to be copied or moved and *label* is a label that is defined in the line command field. The label can be defined by PDF, such as .ZFIRST for the first line of data, or it can be one that you have defined. If you omit the label, you can use the A (after) or B (before) line command to specify where the information is to go. When you press Enter, the member is copied or moved. See “COPY—Copy Data” on page 212 and “MOVE—Move Data” on page 254 for the complete syntax of the commands.
- If you omit the member name, data set name, or z/OS file, and just type the command and the destination of the operation (using either the AFTER label or BEFORE label operand or the A or B line command), the editor displays a panel on which you can specify the name of the member, data set, or z/OS UNIX file to be copied or moved. The only difference between the Edit Move and Edit Copy panels is that with Copy, you can specify the number of lines you want copied.

Note: When using the ASCII (or UTF-8) edit facility with a z/OS UNIX file and the COPY or MOVE command is issued specifying another z/OS UNIX file as the source, ISPF checks if the CCSID of the source file is set to 819 (1208 for UTF-8). If so, ISPF assumes it contains ASCII (or UTF-8) data. This means when ISPF reads the source file, the data is split into records by using the ASCII (and UTF-8) linefeed character (X'0A') and the ASCII (and UTF-8) carriage return character (X'0D') as the record delimiter. For information on the ASCII edit facility, see “Working with ASCII data” on page 50. For information on the UTF-8 edit facility, see “Working with UTF-8 data” on page 51.

Shifting data

When you edit data, the editor automatically shifts characters on a line to the left or right to accommodate insertions or deletions. This shifting can be either *implicit* or *explicit* . Implicit shifts occur when the CHANGE command *string2* length is different from the *string1* length. Explicit shifts occur when you use these commands:

- Line commands

  (
  Column Shift Left

)
  Column Shift Right

  <
  Data Shift Left

  >
  Data Shift Right

- Macro commands

  Shift (
  Column Shift Left

  Shift)
  Column Shift Right

  Shift <
  Data Shift Left

  Shift >
  Data Shift Right

See the descriptions of these commands for the syntax and examples of usage.

Two columns is the default for shift operations. When shifting a block of lines more or less than the default, enter the amount on the first or last line of the block. If you enter it in both places, the line shifts only if:

- Both amounts are the same, or
- The amounts differ, but one is the default (2). Here, the lines shift according to the non-default amount.

If the shift amounts are different and neither amount is the default, an error message appears and the shift is not performed.

Shifting occurs within column boundaries. The default boundaries are typically the first and last columns in which you can type source code for the particular programming language. See "Edit boundaries" on page 23 for a discussion of default boundaries and the procedures for changing them.

Column shift

The simplest kind of shift is a column shift. Column shifting moves all characters within the bounds without altering their relative spacing. Characters shifted past the bounds are deleted. That is, blanks are inserted at the bound from which the characters are being shifted, and the characters are deleted at the opposite bound. So, this shift is called a *destructive* shift because information shifts within column boundaries without regard to its contents, and can result in the loss of data with no error being noted.

If the UNDO mode was on before you entered the shift command, you can recover by using the UNDO command. Otherwise, you can use CANCEL.

Column shifting in lines that contain DBCS strings

These rules apply:

- If half of a DBCS character is in the shift, it is excluded from the operation; the shift count is changed automatically.
- If a column shift causes a DBCS string and an EBCDIC string to be connected, a shift-out or shift-in character, as appropriate, is inserted between the strings. The shift count is changed automatically.
- If left, right, or both boundaries are set, a DBCS character can cross the boundary. The DBCS character that crosses the boundary is excluded from the operation, and the shift count is changed automatically.
- If a request to shift an odd number of columns causes an odd-length DBCS string, the requested shift number is discarded. The shift is processed up to the next field boundary within the boundary, if any. If no field boundary is found, the line number is replaced with this intensified warning message: ==ERR>. Also, the short message for an incomplete data shifting error is displayed.

If you are using the column shifting or data shifting line command while editing a formatted data set, note these points:

- The current boundaries are automatically changed during command processing, and are reset to the original values after processing is complete. Changes are as follows:
  - If the left boundary falls on the second byte of a DBCS character in a DBCS field, the boundary is shifted to the left by 1 byte.
  - If the right boundary does not fall on the same field as the left boundary, it is set to point to the last byte of the field that contains the left boundary. If it falls on the same DBCS field as the left boundary, and it also falls on the first byte of a DBCS character, the right boundary is shifted to the right by 1 byte.
- If you use the data shift or column shift line command to shift a DBCS field and you specify an odd-length shift amount, the shift amount is decreased by one to preserve DBCS data integrity.
- If a shift cannot be completed, it is partially done and the line number is replaced by this intensified warning message: ==ERR>. Remove the message by issuing the RESET primary command, or type over the message or data on that line.
- If a request to shift an odd number of bytes causes an odd-length DBCS string, the shift volume is decreased by one and the operation is performed. The line number is replaced with this intensified warning message: ==ERR>.

Data shift

Data shifting attempts to shift the body of a program statement without shifting the label or comments, and prevents loss of data. This shift is *non-destructive* because it stops before it shifts a nonblank character past the bound. This shift is explicitly done with the < and > line commands, and the SHIFT < and SHIFT > macro commands. The CHANGE command can cause an implicit shift of the same nature.

For data shift left attempts that exceed the current BOUNDS setting, text stops at the left bound and PDF marks the shifted lines with ==ERR> flags. If an error occurs in an excluded line, you can find the error with LOCATE, and remove the error flag by using RESET.

Data shifts are designed to work with typical program sources. In doing so, it makes certain general assumptions about the format of the source code. For instance, the editor assumes:

- Anything beginning at the left bound is a label and should not be shifted.
- If there are two or more consecutive blanks, one can be added or deleted.
- Blanks within quotes (' or ") are to be treated as nonblanks.
- Source statements appear on the left followed by comments on the right.
- Single blanks are used between source code and comment words. Therefore, the only strings of multiple blanks appear between the source code and the comment, and between the comment and its ending delimiter (if there is one). In this example, LABEL and */ are at the left and right bounds, respectively:

```
LABEL: DO I=1 TO 5;          /* The comment...   */
     A=A+B(I);               /* The comment...   */
     END;
```

Keeping the previous assumptions in mind, the editor attempts to move only the source code statement when shifting data. The label and comments are left unchanged. However, if necessary, it shifts the comment also.

Although the editor always uses these assumptions, data shifting is not language-sensitive. It only makes generalities about syntax and individual code entry style.

Finding, seeking, changing, and excluding data

FIND, SEEK, CHANGE, and EXCLUDE allow you to find a specified search string, change one search string to another, or exclude a line containing a specified search string. These commands provide powerful editing functions because they operate on a complete data set rather than on a single line.

The characteristics of each command follow:

FIND
Causes all lines that it finds to be displayed, and moves the cursor (scrolling if necessary) to the first occurrence of the search string.

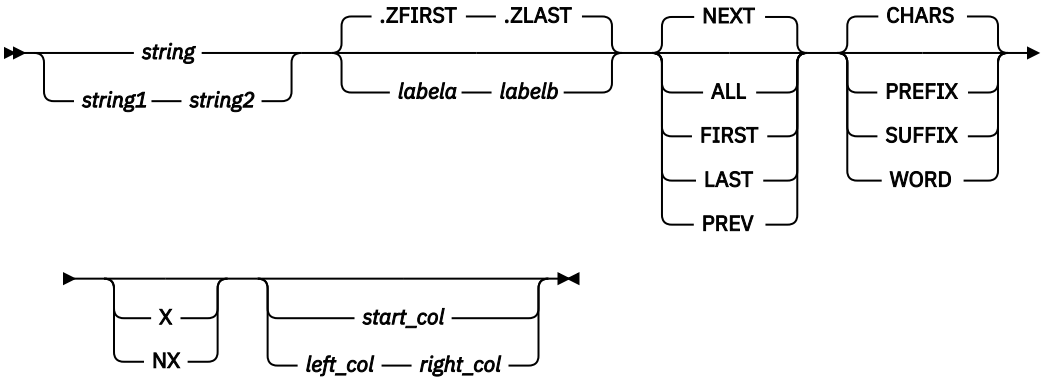
SEEK
A special form of FIND that can only be used in an edit macro. It is different from FIND in that it does not change the exclude status of the lines found.

CHANGE
Causes the same effect as FIND, but it also has a second string operand *(string2)*. During a search, whenever *string1* is found, the editor replaces that string with *string2*. Data to the right is shifted, if necessary.

EXCLUDE
Causes lines that match the search not to be displayed. These lines remain in the data, however. Unlike FIND and CHANGE, it does not require a search string if you use the ALL operand. EXCLUDE ALL is often used with FIND and CHANGE because they cause excluded lines to be redisplayed. Use RESET to cause all lines to be redisplayed.

The scrolling and positioning of the string can be controlled using the Edit_Settings action bar choice or the EDITSET primary command when editing the data. See “EDITSET—Display the Editor Settings Dialog” on page 228 for more information.

The syntax of each command is a variation of that listed here. See the command descriptions in Chapter 10, “Edit primary commands,” on page 193 and Chapter 11, “Edit macro commands and assignment statements,” on page 295 for the exact syntax.



Specifying the search string

The primary control for any search is the search string because it represents the value for which you are looking. Two operands, *string1* and *string2*, are required for the CHANGE command to specify the new value of the string once it is found. The rules for specifying *string1* and *string2* are the same, except that if you type a single asterisk for *string2*, the previous value is used again.

You can define *string*, *string1*, and *string2* to be EBCDIC, DBCS, and mixed strings in any combination. If you delimit a DBCS search string with SO and SI characters, the SO and SI characters are not used as part of the string. If you specify a mixed string that contains no EBCDIC characters, the string is treated as a DBCS string; that is, the SO and SI characters are not used as part of the string.

The editor allows you to specify these kinds of strings:

Simple string
Any series of characters not starting or ending with a quote (' or ") and not containing any embedded blanks, commas, or asterisks.

Delimited string
Any string enclosed (delimited) by either single quotes (') or double quotes ("). The beginning and ending delimiters must be the same character. The string can contain the delimiter character. However, if a delimiter character in the string is followed by a blank ( ) or a comma (,), that delimiter character is processed as the ending delimiter. A delimiter character in the string is also processed as the ending delimiter if it is followed by the letter c, p, r, t, or x. In these cases, the letter is processed as an indication that the preceding string is a character, picture, regular expression, text, or hexadecimal string.

Hexadecimal string
Any delimited string of valid hexadecimal characters, preceded or followed by the character X, such as `X'C27B'`.

Character string
Any delimited string of characters, preceded or followed by the character C, such as `C'conditions for'`. See “Character strings” on page 46 for more information.

Picture string
Any delimited string of picture characters, preceded or followed by the character P, such as `P'.'`. See “Picture strings (string, string1)” on page 46 and “Picture strings (string2)” on page 48 for more information.

Regular expression
Any delimited string of characters, preceded or followed by the character R, such as `R'h[aeiou]d'`, or the characters RC, such as `RC'M[ai]ster'`. Use RC to request a case sensitive search be performed. See "Regular expressions (string, string1)" on page 48 for more information.

Note: The Edit FIND, CHANGE, and EXCLUDE commands do not work with a search argument that contains the command delimiter, even if string delimiters are used. You can specify a hexadecimal search string or use the SETTINGS command to change the command delimiter to a different character.

Simple and delimited strings

If the string is a simple or delimited string, the characters are treated as being both upper and lowercase even if caps mode is off. For example, this command:

```
FIND ALL 'CONDITION NO. 1'
```

successfully finds:

```
CONDITION NO. 1
Condition No. 1
condition no. 1
coNDitION nO. 1
```

Also, all of these commands have the same effect:

```
FIND 'Edit Commands'
FIND 'EDIT COMMANDS'
FIND 'edit commands'
```

You must use delimiters if a string contains embedded blanks or commas, or if a string is the same as a command or keyword. You delimit strings with quotes, either `'` or `"`. For example, to change the next occurrence of EVERY ONE to ALL, type:

```
CHANGE 'EVERY ONE' 'ALL'
```

Note: When using a DBCS terminal, if you specify a text string that contains any SO and SI characters, the string is considered a character string.

Character strings

Use a character string in a FIND, CHANGE, or EXCLUDE command if you want the search to be satisfied by an exact character-by-character match. Lowercase alphabetic characters match only with lowercase alphabetic characters, and uppercase alphabetic characters match only with uppercase.

For example, `FIND C'XYZ'` finds the characters XYZ only, not xyz.

Picture strings (string, string1)

A picture string in a FIND, CHANGE, or EXCLUDE command allows you to search for a particular kind of character without regard for the specific character involved. You can use special characters within the picture string to represent the kind of character to be found, as follows:

- **String**
  Meaning

  P'='
  Any character

  P'¬'
  Any character that is not a blank

  P'.'
  Any character that cannot be displayed

  P'#'
  Any numeric character, 0-9

P'-'
: Any nonnumeric character

P'@'
: Any alphabetic character, uppercase or lowercase

P'<'
: Any lowercase alphabetic character

P'>'
: Any uppercase alphabetic character

P'$'
: Any special character, neither alphabetic nor numeric

If you are using an APL or TEXT keyboard, you can use these additional characters in a picture string:

- **P'□'**
  Any APL-specific or TEXT-specific character

  P'_'
  Any underscored nonblank character

A picture string can include alphanumeric characters, which represent themselves, mixed with other characters. If the character does not have a special meaning (such as @ standing for any alphabetic), the character is treated as itself.

When using a DBCS terminal, you cannot specify a DBCS field as the subject of a picture string for the FIND operation.

When processing a picture string the interpretation of characters is based on the PDF terminal translation tables that are loaded. For example, characters that cannot be displayed are determined using the translation tables for generic string characters and generic string special characters.

Picture string examples

- To find a string of 3 numeric characters:
  - ```
    FIND P'###'
    ```
- To find any 2 characters that are not blanks but are separated by a blank:
  ```
  FIND P'¬ ¬'
  ```
- To find any character that cannot be displayed:
  ```
  FIND P'.'
  ```
- To find a blank followed by a numeric character:
  ```
  FIND P' #'
  ```
- To find a numeric character followed by AB:
  ```
  FIND P'#AB'
  ```
- To find the next character in column 72 that is not a blank:
  ```
  FIND P'¬' 72
  ```
- To change any characters in columns 73 through 80 to blanks:
  ```
  CHANGE ALL P'=' ' ' 73 80
  ```
- To find the next line with a blank in column 1 and a character in column 2 that is not a blank:
  ```
  FIND P' ¬'  1
  ```
  ```

When you use the special characters = or . and a character that cannot be displayed is found, that character's hexadecimal representation is used in the confirmation message that appears in the upper-right corner of the panel. For example, the command `FIND P'..'` could result in the message `CHARS X'0275' FOUND`.

Picture strings (string2)

In a CHANGE command, *string2* can be a picture string with these rules and restrictions:

- The length of *string2* must be the same as the length of *string1*.
- The only valid special characters are =, >, and <.

String
Meaning

P'='
Equal to the corresponding character in *string1*

P'>'
Converts the corresponding character in *string1* to uppercase

P'<'
Converts the corresponding character in *string1* to lowercase

Picture string examples

- To change an alphabetic, alphabetic, numeric, numeric string so that the alphabetic characters become uppercase characters and the numeric characters are unchanged:

```
CHG P'@@##' P'>>=='
```

- To change all characters to uppercase:

```
CHG ALL P'<' P'>'
```

Regular expressions (string, string1)

A regular expression in a FIND, CHANGE, or EXCLUDE command allows you to search for a string matching a *regular expression*.

ISPF uses the IBM C regcomp and regexec functions to compile and execute a regular expression specified with a FIND, CHANGE, or EXCLUDE command. These are supported by the C runtime library and the C runtime library must be available.

ISPF queries the host code page defined for your TN3270 session. If the code page is one of the following:

```
00037   00871   01123   01156
00273   00875   01140   01157
00277   00924   01141   01158
00278   00930   01142   01160
00280   00933   01143   01165
00284   00935   01144   01364
00285   00937   01145   01371
00290   00939   01146   01388
00297   01025   01147   01390
00424   01026   01148   01399
00425   01027   01149   04971
00500   01047   01153   05123
00838   01112   01154   08482
00870   01122   01155   12712
```

ISPF uses the IBM C setlocale function with LC_ALL to set the corresponding C locale. This is done so that the special symbols (such as square brackets) within the regular expression are correctly interpreted when the regcomp function is used to compile the regular expression.

If the TN3270 code page is not one of the listed code pages then the default C locale is used when compiling the regular expression. If a regular expression is encountered on a FIND, CHANGE, or EXCLUDE

command that is specified in an Edit macro that is called from a batch Edit session (where no terminal is attached), code page 1047 is used.

The simplest form of regular expression is a string of characters with no special meaning.

The following characters do have a special meaning; they are used to form extended regular expressions:

Symbol
Description

. (period)
The period symbol matches any one character except the terminal newline character.

For example, the regular expression `d.g` matches "dig", "dug", and "dog", but not "dg", though it matches "dgg".

*** (asterisk)**
The asterisk symbol matches zero or more instances of the previous character.

For example, the regular expression `he*ath` matches "hath" and "heath" and (if it exists) "heeath".

? (question mark)
The question mark symbol matches zero or one instance of the previous character.

For example, the regular expression `behaviou?r` matches "behaviour" and "behavior".

+ (plus)
The plus symbol matches one or more instances of the previous character.

For example, the regular expression `south+ern` matches "southern" and "southhern", but not "soutern". (If you also wanted a match for "soutern", use `south*ern` as the regular expression.)

| (vertical bar)
The vertical bar symbol acts as an OR operator and matches the values to the left and right of the vertical bar.

For example, the regular expression `Jack|Jill` matches "Jack" and "Jill".

\ (backslash)
The backslash symbol acts as an escape sequence. Use it when you want search for a regular expression symbol. The backslash character immediately precedes the symbol in the expression.

For example, the regular expression `a.\+.b` matches the string "a + b".

[string]
A string within square brackets matches any one of the characters in string.

For example, the regular expression `d[iu]g` matches "dig" and "dug", but not "dog".

[character-character]
The hyphen symbol, within square brackets, means *through*. It fills in the intervening characters according to the current collating sequence. For example, [a-z] can be equivalent to [abc...xyz] or, with a different collating sequence, it can be equivalent to [aAbBcC...xXyYzZ].

For example, the regular expression `m[a-z]p` matches "map" and "mop", but not "m9p", since 9 is not in the range a to z.

[^string]
The caret symbol, when the first character inside square brackets, negates the following characters within the square brackets.

For example, the regular expression `d[^iu]g` matches "dog", but not "dig" or "dug".

{m} {m,u} {m,}
Integer values enclosed in {} indicate the number of times to apply the preceding regular expression. *m* is the minimum number, and *u* is the maximum number. {*m*} indicates an exact number of times to apply the regular expression. {*m*,*u*} indicates a range of instances. {*m*,} indicates that there is a minimum, but no maximum.

For example:

- `m[eaiy]{2}n` matches "main", "mien" and "mean", but it does not match "man", because there is only one instance of the letters in the square brackets. Nor does it match "mayan", because this has three instances of the letters in the square brackets.
- `[0-9][a-z]{2,3}[0-9]` matches "7ab5" and "4abc3", but not "7b5", nor "4abcd3".
- `[0-9][a-z]{2,}[0-9]` matches "4ab3", "4abc3", "4abcd3", and so on, but not "4a3".

(expression)
Used to group parts of the expression into sub-expressions. This can be used to limit an operator to a sub-expression.

For example, the regular expression `z/OS.((1\.1[0-3])|(2\.[1-2]))` matches "z/OS 1.13" and "z/OS 2.1".

Note: You can use the] (right square bracket) alone within a pair of square brackets, but only if it immediately follows either the opening left square bracket or if it immediately follows [^. For example: []-] matches the] and - characters.

Regular expressions cannot be used for *string2* for a CHANGE command.

Effect of CHANGE command on column-dependent data

Column-dependent data is groups of nonblank source data separated by two or more blanks, such as a table. When you use CHANGE to change column-dependent data, ISPF attempts to maintain positional relationships. For instance, if you change a long word to a short word, the editor pads the short word with blanks. This padding maintains the column position of any data to the right of the change by preventing it from shifting left.

When only one blank separates words, as in most text data, padding does not occur. Changing a long word to a short word causes data to the right of the change to shift left.

Using the CHANGE command with EBCDIC and DBCS data

If you are editing a data set that contains both EBCDIC and DBCS data, note these rules about CHANGE strings:

- The SO and SI characters that delimit the CHANGE string are used as part of the string only if necessary. If you specify replacement of an EBCDIC string with a DBCS string, they are used. If you specify replacement of a DBCS string with another DBCS string, they are not used.
- If you specify in a CHANGE string that an SO or SI character be changed to another character, the result is unpredictable.
- If you specify a CHANGE string that causes a field length of zero and the boundary falls between the SO and SI characters, the SO/SI or SI/SO character strings that are next to each other are replaced with a DBCS blank. If the boundary does not fall between the SO and SI characters, the SO/SI or SI/SO characters that are next to each other are removed.
- If the lengths of the two strings specified in CHANGE are different, these actions occur:
  - If *string1* is shorter than *string2*, the data to the right of *string1* is shifted to the left up to some breakpoint. Breakpoints include the border between an EBCDIC field and a DBCS field, a double or single blank, or the right boundary set by a BOUNDS command.
  - If *string1* is longer than *string2*, blanks in the record to the right of *string1* are used to make room. When blanks in a DBCS field are used, they are used in units of 2 bytes.
- If a DBCS field crosses the right boundary, CHANGE can cause an odd-length DBCS field. If this happens, the right boundary is ignored and the operation takes place.

Working with ASCII data

When you are working with an ASCII file, you can use the ASCII editing facility to translate data from and to ASCII when displaying and receiving input from the terminal.

The ASCII editing facility converts the ASCII data to the corresponding EBCDIC representation prior to displaying at the terminal. Also, when you enter data from the terminal, the data is converted from the CCSID of the terminal to ASCII before being stored in the file you are editing.

To activate the ASCII editing facility for a PDS member or data set select 1 (ASCII) for the Data Encoding option on the edit entry panel. Otherwise start editing the member or data set and then issue the command `SOURCE ASCII`.

The ASCII editing facility is automatically invoked for a z/OS UNIX file tagged with a CCSID of 819. The following can be used to activate the ASCII editing facility for a z/OS UNIX file that is not tagged with a CCSID of 819:

- Select 1 (ASCII) for the Data Encoding option on the Edit/View entry panel.
- Specify the ASCII parameter when calling the EDIT or VIEW services.
- Specify the EA (Edit - ASCII) or VA (View - ASCII) line commands on the z/OS UNIX directory list display.

When using the ASCII editing facility with a z/OS UNIX file, ISPF ensures the file's CCSID is set to 819 when the file is saved.

Note: If you try to use the ASCII editing facility with a file enabled for z/OS UNIX automatic codeset conversion, ISPF does not invoke the ASCII editing facility and codeset conversion is used to convert the data to EBCDIC.

The ISPF editor then treats the source data as though it is ASCII data and converts it from ASCII to the CCSID of the terminal for display purposes, although the data remains unchanged within the file. When you input or modify data at the terminal, the ISPF editor translates the data entered from the CCSID of the terminal to ASCII before storing the data in the file.

While editing a PDS member or data set, you can revert back to a normal mode, where the data is not translated from and to ASCII when displaying and receiving input from the terminal, by issuing the command:

```
RESET SOURCE
```

Restructuring data based on the linefeed character

ASCII data can contain linefeed characters (X'0A'). If the data has been uploaded from another computing platform, the data may not be correctly structured based on the linefeed characters.

To restructure the data based on the linefeed character, issue the command LF.

Note:

1. There is no reverse process for restructuring the data based on the linefeed character. Consequently, once you have saved the data after an LF command the change is permanent.
2. Do not enter the LF command more than once against the same file as blanks following linefeed characters are interpreted as the leading data of the next record.
3. The ASCII editing facility uses MVS Conversion Services to translate the data between ASCII (CCSID 819) and the CCSID supported by the terminal. It is a requirement that MVS Conversion Services be installed and the required translations specified to it, in order for the ASCII editing facility to be operable.

When using the ASCII edit facility for a z/OS UNIX file, the LF primary command is not available as the editor automatically restructures the data based on the linefeed character.

Working with UTF-8 data

When you are working with an UTF-8 file, you can use the UTF-8 editing facility to translate data from and to UTF-8 when displaying and receiving input from the terminal.

The UTF-8 editing facility converts the UTF-8 data to the corresponding EBCDIC representation prior to displaying at the terminal. Also, when you enter data from the terminal, the data is converted from the CCSID of the terminal to UTF-8 before being stored in the file you are editing.

To activate the UTF-8 editing facility for a PDS member or data set select 2 (UTF8) for the Data Encoding option on the edit entry panel.

The UTF-8 editing facility is automatically invoked for a z/OS UNIX file tagged with a CCSID of 1208. The following can be used to activate the UTF-8 editing facility for a z/OS UNIX file that is not tagged with a CCSID of 1208:

- Select 2 (UTF8) for the Data Encoding option on the Edit/View entry panel.
- Specify the UTF8 parameter when calling the EDIT or VIEW services.
- Specify the EU (Edit - UTF-8) or VU (View - UTF-8) line commands on the z/OS UNIX directory list display.

When using the UTF-8 editing facility with a z/OS UNIX file, ISPF ensures the file's CCSID is set to 1208 when the file is saved.

The ISPF editor then treats the source data as though it is UTF-8 data and converts it from UTF-8 to the CCSID of the terminal for display purposes, although the data remains unchanged within the file. When you input or modify data at the terminal, the ISPF editor translates the data entered from the CCSID of the terminal to UTF-8 before storing the data in the file.

Restructuring data based on the linefeed character

UTF-8 data can contain linefeed characters (X'0A'). If the data has been uploaded from another computing platform, the data may not be correctly structured based on the linefeed characters.

To restructure the data based on the linefeed character, issue the command LF.

Note:

1. There is no reverse process for restructuring the data based on the linefeed character. Consequently, once you have saved the data after an LF command the change is permanent.
2. Do not enter the LF command more than once against the same file as blanks following linefeed characters are interpreted as the leading data of the next record.
3. The UTF-8 editing facility uses MVS Conversion Services to translate the data between UTF-8 (CCSID 1208) and the CCSID supported by the terminal. It is a requirement that MVS Conversion Services be installed and the required translations specified to it, in order for the UTF-8 editing facility to be operable.

When using the UTF-8 edit facility for a z/OS UNIX file, the LF primary command is not available as the editor automatically restructures the data based on the linefeed character.

Controlling the search

After you specify the search string, you can then specify how much of the data you want to search, as well as the starting point and direction of the operation.

Extent of the search

You can limit the lines to be searched by first assigning a label to the first and last lines to be searched, and then specifying the labels on the command (range operand).

If you want to limit the search to a single line, assign a label to it, and then specify the label twice to show the first and last line of the range. For more information about labels, see “Labels and line ranges” on page 59.

Starting point and direction of the search

To control the starting point and direction of the search, use one of these operands:

NEXT
Starts at the first position after the current cursor location and searches ahead to find the next occurrence of *string1*. NEXT is the default.

ALL
Starts at the top of the data and searches ahead to find all occurrences of *string1*. The long verification message, which PDF displays when you enter the HELP command in response to the short verification message, shows the number of occurrences found. If you use this operand with CHANGE, the lines changed are marked with ==CHG> flags, and lines that cannot be changed are marked with ==ERR> flags. The status of these lines can be used by LOCATE and changed by RESET.

FIRST
Starts at the top of the data and searches ahead to find the first occurrence of *string1*.

LAST
Starts at the bottom of the data and searches backward to find the last occurrence of *string1*.

PREV
Starts at the current cursor location and searches backward to find the previous occurrence of *string1*.

If you specify NEXT, ALL, or FIRST, the direction of the search is forward. When you press the assigned function keys, the RFIND or RCHANGE commands find or change the next occurrence of the designated string. If you specify LAST or PREV, the direction of the search is backward. When you specify those operands, the editor finds or changes the previous occurrence of the string.

The search proceeds until the editor finds one or all occurrences of *string1*, or the end of data.

If you omit the ALL operand on the CHANGE command, the editor searches only for the first occurrence of *string1* after the current cursor location. If the cursor is not in the data area of the panel, the search starts at the beginning of the first line currently displayed. Scrolling is performed, if necessary, to bring the string into view.

After you make the change, the cursor is positioned at the end of the changed string; a verification message is displayed in the upper right corner of the panel.

Depending on the direction of the search, if the string is not found between the current cursor location and the end or beginning of data, a message is displayed and an audible alarm, if installed, is sounded.

If *string1* is not found, one of these actions takes place:

- A `NO string1 FOUND` message is displayed in the upper right-hand corner of the panel.
- If CHANGE or EXCLUDE was repeated using RFIND or RCHANGE, either `BOTTOM OF DATA REACHED` or `TOP OF DATA REACHED` is displayed, depending on the direction of the search. When either of these messages is displayed, you can enter RFIND or RCHANGE again to continue the search by wrapping to the top or bottom of the data. If *string1* is still not found, a `NO string1 FOUND` message is displayed.

When you type a primary command, the cursor is, of course, positioned on the command line. In the case of the CHANGE, EXCLUDE, and FIND primary commands, if you specify NEXT or PREV, the search starts at the *current cursor location* in a forward or backward direction respectively:

- If you specify NEXT and then press Enter without repositioning the cursor, the current cursor position is taken to be at the top of the data. The search starts in a forward direction from that point.
- If you specify PREV and then press Enter without repositioning the cursor, the current cursor position is taken to be at the bottom of the data. The search starts in a backward direction from that point.

To obtain the result you want, you may need to reposition the cursor *after* you have typed the primary command, but *before* you press Enter.

Qualifying the search string

You can specify additional characteristics of *string1* by using the operands PREFIX, SUFFIX, CHARS, and WORD. You can abbreviate PREFIX, SUFFIX, and CHARS to PRE, SUF, and CHAR, respectively.

CHARS
Locates *string1* anywhere the characters match. This is the default.

PREFIX
Locates *string1* at the beginning of a word.

SUFFIX
Locates *string1* at the end of a word.

WORD
string1 is delimited on both sides by blanks or other non-alphanumeric characters.

In this example, the editor would find the highlighted strings only:

```
CHARS 'DO'  - DO DONE ADO ADOPT 'DO' +ADO (DONE) ADO-
PREFIX 'DO' - DO DONE ADO ADOPT 'DO' +ADO (DONE) ADO-
SUFFIX 'DO' - DO DONE ADO ADOPT 'DO' +ADO (DONE) ADO-
WORD 'DO'   - DO DONE ADO ADOPT 'DO' +ADO (DONE) ADO-
```

If you do not specify an operand, the default is CHARS.

Limiting the search to specified columns

The *left_col* and *right_col* operands allow you to search only a portion of each line, rather than the entire line. These operands, which are numbers separated by a comma or by at least one blank, show the starting and ending columns for the search. These rules apply:

- If you specify neither *left_col* nor *right_col*, the search continues across all columns within the current boundary columns.
- If you specify only *left_col*, the editor finds the string only if the string starts in the specified column.
- If you specify both *left_col* and *right_col*, the editor finds the string only if it is entirely within the specified columns.

Split screen limitations

When *string1* is not found within the data that is displayed on the screen, the search operation scrolls the data so that *string1* appears on the second displayed line of the data area. If only one line of data is showing in split screen mode, the data on the second line (thus, *string1*) cannot be seen, and so the cursor is placed on the command line.

Limiting the search to excluded or non-excluded lines

You can limit the lines to be searched by first using the X or NX operands:

X
Scan only lines that are excluded from the display.

NX
Scan only lines that are not excluded from the display.

If you omit these operands, both excluded and non-excluded lines are searched. When you issue a FIND or CHANGE command that includes searching excluded lines, all lines found are displayed. EXCLUDE can also find labels assigned to excluded lines.

Using the X (Exclude) line command with FIND and CHANGE

You can use the X (exclude) line command with FIND and CHANGE to display only those lines containing the search string or those lines that have been changed. For example, if your data set contains 99 999 lines or less, type X99999 in the line command field of the first line to exclude all of the lines from the display. Then enter a CHANGE command, such as:

```
CHANGE ALL XYZ ABC
```

All lines containing the search string XYZ are redisplayed with XYZ changed to ABC and with the cursor at the end of the first string changed.

Similarly, you can enter a FIND command:

```
FIND ALL XYZ
```

Here, all lines containing the search string XYZ are redisplayed with the cursor at the beginning of the first string found.

Repeating the FIND, CHANGE, and EXCLUDE commands

The easiest way to repeat FIND, CHANGE, and EXCLUDE without retyping them is to assign those commands to function keys. The defaults are:

F5/17
RFIND

F6/18
RCHANGE

The search begins at the cursor. If the cursor has not moved since the last FIND, CHANGE, or EXCLUDE command, the search continues from the string that was just found. Instead of retyping *string1*, you can type an asterisk to specify that you want to use the last search string. If you decide to type RCHANGE or RFIND on the command line instead of using a function key, position the cursor at the desired starting location before pressing Enter.

All three commands share the same *string1*. Therefore:

```
FIND ABC
```

followed by:

```
CHANGE * XYZ
```

first shows you where ABC is, and then replaces it with XYZ. However, you can do this more easily by typing:

```
CHANGE ABC XYZ
```

Then press F5/17 to repeat FIND. The editor finds the next occurrence of ABC. You can either press F5/17 to find the next ABC, or F6/18 to change it. Continue to press F5/17 to find remaining occurrences of the string.

The previous value of a search string, specified by an asterisk or by use of RFIND or RCHANGE, is retained until you end your editing session.

Examples

See:

- “FIND command example” on page 55
- “CHANGE command example” on page 56
- “EXCLUDE command example” on page 57

FIND command example

To find all occurrences of "MIMIC" in a member such as the one shown in Figure 11 on page 56, type `FIND ALL MIMIC` on the command line.

```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
------------------------------------------------------------------------------
EDIT       SBURNF.PRIVATE.EXEC(FCEXMP) - 01.00            Columns 00001 00072
Command ===> find all mimic                                  Scroll ===> CSR
****** ***************************** Top of Data ******************************
000001 /* REXX */
000002 /* REXX */
000003 ADDRESS TSO
000004 /*                                                                    */
000005 /* RECREATE THE OLD BACKUP DATA SETS                                  */
000006 /*                                                                    */
000007 CALL MIMIC "ALLOC DA('PDFTDEV.SVT2.ARCHDEF')"
000008 CALL MIMIC "ALLOC DA('PDFTDEV.SVT2.CLIST')"
000009 CALL MIMIC "ALLOC DA('PDFTDEV.SVT2.CPP')"
000010 CALL MIMIC "ALLOC DA('PDFTDEV.SVT2.EXEC')"
000011 CALL MIMIC "ALLOC DA('PDFTDEV.SVT2.GIF')"
000012 CALL MIMIC "ALLOC DA('PDFTDEV.SVT2.GMLINC')"
000013 CALL MIMIC "ALLOC DA('PDFTDEV.SVT2.HPP')"
000014 CALL MIMIC "ALLOC DA('PDFTDEV.SVT2.HSAS65')"
000015 CALL MIMIC "ALLOC DA('PDFTDEV.SVT2.LEL')"
000016 CALL MIMIC "ALLOC DA('PDFTDEV.SVT2LMAP')"
000017 CALL MIMIC "ALLOC DA('PDFTDEV.SVT2.LOAD')"
 F1=Help      F2=Split     F3=Exit      F5=Rfind     F6=Rchange   F7=Up
 F8=Down      F9=Swap     F10=Left     F11=Right    F12=Cancel
```

Figure 11. Before FIND command (ISREDDE2)

After you press Enter, the editor searches for the string starting at the top of the data, places the cursor at the beginning of the first occurrence (1), and displays the number of occurrences (2) as shown in Figure 12 on page 56.

```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
------------------------------------------------------------------------------
EDIT       SBURNF.PRIVATE.EXEC(FCEXMP) - 01.00            2  21 CHARS 'MIMIC'
Command ===>                                                 Scroll ===> CSR
****** ***************************** Top of Data ******************************
000001 /* REXX */
000002 /* REXX */
000003 ADDRESS TSO
000004 /*                                                                    */
000005 /* RECREATE THE OLD BACKUP DATA SETS                                  */
000006 /*                                                                    */
000007 CALL 1MIMIC "ALLOC DA('PDFTDEV.SVT2.ARCHDEF')"
⋮
```

Figure 12. After FIND command

CHANGE command example

To change "MIMIC" to "WILLY" enter `C ALL MIMIC WILLY` as shown in Figure 13 on page 56.

```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
------------------------------------------------------------------------------
EDIT       SBURNF.PRIVATE.EXEC(FCEXMP) - 01.00            21 CHARS 'MIMIC'
Command ===> c all mimic willy                               Scroll ===> CSR
****** ***************************** Top of Data ******************************
000001 /* REXX */
⋮
```

Figure 13. Before CHANGE command

The editor changes all occurrences of the string starting at the top of the data and inserts a ==CHG> flag next to each changed line, as shown in Figure 14 on page 57.

```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
EDIT       SBURNF.PRIVATE.EXEC(FCEXMP) - 01.00           CHARS 'MIMIC' changed
Command ===> _____________________________________________ Scroll ===> CSR
****** ***************************** Top of Data ******************************
000001 /* REXX */
000002 /* REXX */
000003 ADDRESS TSO
000004 /*                                                                    */
000005 /* RECREATE THE OLD BACKUP DATA SETS                                  */
000006 /*                                                                    */
==CHG> CALL WILLY "ALLOC DA('PDFTDEV.SVT2.ARCHDEF')"
==CHG> CALL WILLY "ALLOC DA('PDFTDEV.SVT2.CLIST')"
==CHG> CALL WILLY "ALLOC DA('PDFTDEV.SVT2.CPP')"
==CHG> CALL WILLY "ALLOC DA('PDFTDEV.SVT2.EXEC')"
==CHG> CALL WILLY "ALLOC DA('PDFTDEV.SVT2.GIF')"
==CHG> CALL WILLY "ALLOC DA('PDFTDEV.SVT2.GMLINC')"
==CHG> CALL WILLY "ALLOC DA('PDFTDEV.SVT2.HPP')"
==CHG> CALL WILLY "ALLOC DA('PDFTDEV.SVT2.HSAS65')"
==CHG> CALL WILLY "ALLOC DA('PDFTDEV.SVT2.LEL')"
==CHG> CALL WILLY "ALLOC DA('PDFTDEV.SVT2LMAP')"
==CHG> CALL WILLY "ALLOC DA('PDFTDEV.SVT2.LOAD')"
 F1=Help      F2=Split     F3=Exit      F5=Rfind     F6=Rchange   F7=Up
 F8=Down      F9=Swap     F10=Left     F11=Right    F12=Cancel
```

Figure 14. After CHANGE command

EXCLUDE command example

When you enter an EXCLUDE command like `ex /* all` on the command line (Figure Figure 15 on page 57), the editor excludes all lines with that string starting at the top of the data (Figure Figure 16 on page 57).

```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
EDIT       SBURNF.PRIVATE.EXEC(FCEXMP) - 01.00           CHARS 'MIMIC' changed
Command ===> ex /* all                                   Scroll ===> CSR
****** ***************************** Top of Data ******************************
000001 /* REXX */
000002 /* REXX */
000003 ADDRESS TSO
000004 /*                                                                    */
000005 /* RECREATE THE OLD BACKUP DATA SETS                                  */
000006 /*                                                                    */
==CHG> CALL WILLY "ALLOC DA('PDFTDEV.SVT2.ARCHDEF')"
==CHG> CALL WILLY "ALLOC DA('PDFTDEV.SVT2.CLIST')"
⋮
```

Figure 15. Before EXCLUDE command

```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
EDIT       SBURNF.PRIVATE.EXEC(FCEXMP) - 01.00                5 chars '/*/
Command ===> _____________________________________________ Scroll ===> CSR
****** ***************************** Top of Data ******************************
- - -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  2 Line(s) not Displayed
000003 ADDRESS TSO
- - -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  3 Line(s) not Displayed
==CHG> CALL WILLY "ALLOC DA('PDFTDEV.SVT2.ARCHDEF')"
==CHG> CALL WILLY "ALLOC DA('PDFTDEV.SVT2.CLIST')"
⋮
```

Figure 16. After EXCLUDE command

Excluding lines

You can exclude lines from a data set using the X (exclude) line command as well as the EXCLUDE primary command.

When you are editing a program that exceeds the screen size, it can be difficult to determine whether the control structure and indentation levels are correct. Excluding lines allows you to remove one line or a block of lines from the display so that you can see the general control structure. Each block of excluded lines is replaced by a single line containing a message in the form "*n* Line(s) not Displayed". Excluded lines are treated as valid data lines. They are excluded from the display, but are not deleted from the data.

The X line command can be entered in these ways:

```
X
Xn
XX
```

The first two forms allow you to exclude one line or a specified number of lines.

The third form allows you to exclude a block by typing XX on the first and last lines of the block of lines that you want to exclude. The first and last lines do not need to be on the same page; after typing the first XX you can scroll to the second XX.

You can enter any line command that usually operates on a single line in the line command field of the excluded lines message. For example, if you enter the D (delete) line command, the complete block of excluded lines is deleted.

Hiding excluded lines

You can also suppress the lines containing the "*n* Line(s) not Displayed" message by entering HIDE as a primary command or in an edit macro. HIDE removes the excluded lines messages from the display and indicates the location of each block of excluded lines by underscoring the line number field of the previous line.

The RESET HIDE primary command and edit macro command restores the lines containing the "*n* Line(s) not Displayed" message to the display.

Redisplaying excluded lines

To display all excluded lines, enter the RESET EXCLUDED primary command. Alternatively, you can display one or more excluded lines again by entering the S (show), F (first), or L (last) line commands, typing over the dashes in the line command field. If these commands are typed outside the dashes of the command line area, no action is taken.

You can add a number following any of these line commands to cause more than one line to appear again:

```
Sn
```

```
Fn
```

```
Ln
```

FIND and CHANGE also cause any excluded lines that meet the search criteria to appear again.

The S line command causes the editor to scan a block of excluded lines, and one or more lines is selected to be appear again. The selected lines are those with the leftmost indentation levels; that is, the lines that contain the fewest leading blanks. If you type S3, for example, the three lines with the leftmost indentation level are displayed again. If more than three lines exist at this indentation level, only the first three are displayed.

Note: If you enter an S line command to display all but one line of an excluded block, then that line is also displayed. This could result in more lines being displayed than the number you requested. For example, if five lines are excluded in a block, an S4 command causes all five lines to be displayed.

Redisplaying a range of lines

The FLIP command lets you reverse the exclude status of a specified group of lines in a file or of all the lines in the file. This is useful when you have used the `'X ALL;FIND ALL xyz'` command to find lines containing a string (`xyz`) and want to see the lines which do not contain the string. You can also use FLIP to show excluded note, message, and information lines.

You can enter one or two labels to specify the range of lines whose include status you want to reverse. If no labels are specified, the exclude status of all of the lines is reversed.

To reverse the exclude status of all the lines in a file, use this syntax:

```
FLIP
```

To reverse the exclude status of specified lines, use this syntax:

```
FLIP .a .b
```

The lines between labels .a and .b are redisplayed.

Labels and line ranges

A label is an alphabetic character string used to name lines or strings of data for easy reference. Because labels remain with the lines to which they are assigned, they are especially useful in keeping track of lines whose numbers might change. Most labels are assigned in macros, but certain labels are automatically assigned by the ISPF editor.

You can assign a label to a line by typing the label over the line number on the left side of the panel. The label is displayed in place of the number whenever the line is being displayed. If you then move the line, the label moves with it. You cannot type a label on a non-data line or on the line that is displayed to show one or more lines is excluded.

A label must begin with a period, and be followed by no more than 5 alphabetic characters (8 for edit macros), the first of which cannot be a Z. Labels beginning with Z are reserved for use by the editor. No special or numeric characters are allowed.

To eliminate a single label, blank it out. To eliminate all labels, use the RESET LABEL command.

An edit macro can assign labels to lines that the macro references frequently. See “Labels in edit macros” on page 103 for details.

Editor-assigned labels

The editor automatically assigns special labels that begin with the letter Z. Only the editor can assign a special label.

These built-in labels are:

.ZCSR
The data line on which the cursor is currently positioned.

.ZFIRST
The first data line (same as relative line number 1). Can be abbreviated .ZF.

.ZLAST
The last data line. Can be abbreviated .ZL.

Unlike other labels, .ZCSR, .ZFIRST, and .ZLAST do not stay with the same line. Label .ZCSR stays with the cursor, and labels .ZFIRST and .ZLAST remain with the current first and last lines.

Note: Labels that are five characters long and begin with the letter 'O' have special meaning to the HILITE feature of the ISPF editor. When a 5-character label starting with O, such as .OAAAA, is shown on the screen, the language highlighting features are disabled and the lines with these special labels are displayed in blue. This feature is used by the COMPARE command.

Specifying a range

Labels allow you to specify a line or a range of lines on a primary command. You can specify two labels to define a range of lines to be processed on these commands:

```
CHANGE   DELETE   EXCLUDE
FIND     LOCATE   REPLACE
RESET    SORT     SUBMIT
```

The range operand is always optional. If you do not specify a range, it defaults to .ZFIRST and .ZLAST. For example, the command:

```
CHANGE ALL 'TEST' 'FINAL'
```

starts at the first line of the data being edited and scans all lines up to and including the last line, changing all occurrences of TEST to FINAL.

However, the command:

```
CHANGE .ZCSR .ZLAST ALL 'TEST' 'FINAL'
```

specifies a range, and is thus interpreted differently. The command changes only the last part of the data.

When you use labels to specify a range, you must always use two labels to define the first and last lines, inclusively. To process a single line, repeat the label:

```
CHANGE ALL " " "_" .A .A
```

The command in the previous example is interpreted as "Change all blanks to underscores on the .A line."

The order in which you specify the labels is not important. The editor assumes that the line closer to the beginning of the data set is the first line of the range, and the line closer to the end of the data set is the last.

A common error when using a range is to assume that the search begins at the first character of the line with the first label. Remember, however, that the default is NEXT and that the search starts at the cursor location. Lines outside the range are logically the same as the TOP OF DATA and BOTTOM OF DATA lines. Use the FIRST, LAST, or PREV operands to ensure that the search begins within the range.

Using labels and line ranges

The examples shown here show the results of using labels to identify ranges of lines. They show that the order of both labels and other operands is not important, and that you can type both labels and operands in either uppercase or lowercase.

- This command locates the first line flagged: ==CHG> between the line labeled .start and the line with the cursor on it:

  ```
  locate first chg .start .zcsr
  ```

- This command changes the last occurrence of "PRE" to "POST" between the first line and the line marked with the .HERE label:

  ```
  CHANGE LAST PRE POST .HERE .ZFIRST
  ```

- This command changes all occurrences of "PRE" to "POST" from the .MYLAB line to the last line of the data set:

```
CHANGE PRE POST ALL .MYLAB .ZL
```

- This command finds the word "HIGHER" between the .START line and the .END line:

```
FIND HIGHER WORD .START .END
```

Word processing

This topic is a general overview of three line commands for word or text processing: TF (text flow), TS (text split), and TE (text entry). The editor also provides three corresponding edit macro commands: TFLOW, TSPLIT, and TENTER. For the sake of simplicity, only the line commands are referred to. However, the descriptions apply to the macro commands, as well.

TF, TS, and TE assume that the data is grouped in paragraphs. A paragraph is a group of lines that begin in the same column. The first line of a paragraph is excluded from the grouping. The editor interprets any indentation or blank line as representing a new paragraph. It also recognizes word processor control words that are used by the Document Composition Facility as the beginning of a paragraph. These control words begin with a period, a colon, or an ampersand.

If you use text line commands frequently, you can assign both the TS and TF commands to function keys. Use KEYS to reassign the keys. For example:

```
F10 ===> :TS
F11 ===> :TF
```

Now you can split text by moving the cursor to the desired split point within a line and pressing F10. Having typed the new material, press F11 to restructure the text from the line containing the cursor to the end of the paragraph.

Formatting paragraphs

The TF (text flow) line command formats paragraphs. It assumes that the sentences are roughly in paragraph form with a ragged right margin when it attempts to recognize groupings. TF can be followed by a number (TF72 for example) that specifies the desired right side column for the paragraph. If you do not specify a number, the right side of the panel is used unless you have set bounds different from the default. In that case, the right boundary is used. The editor assumes that because the first line of a paragraph may be at a different indentation level than the remainder of the paragraph, the starting column of the second line is the left side of the paragraph.

When formatting paragraphs, the editor:

- Moves text so that each line contains the maximum number of words. TF limits its activity to within the bounds. Thus, it can be used to flow text within a border.
- Keeps any blanks between words.
- Assumes one blank between the word at the end of a line and the word on the next line except when the line ends with a period. In that case, the editor inserts two blanks.

The end of the paragraph is denoted by a blank line, a change in indentation, or the special characters period (.), colon (:), ampersand (&), or left angle bracket (<) in the left boundary column. These special characters are used as Document Composition Facility (SCRIPT/VS) control word delimiters.

The restructure operation removes trailing blanks on a line by using words from the following line. It does not, however, remove embedded blanks within a line. Accordingly, if one or more words in a line are to be removed, delete the words rather than type over them.

The text to be restructured is taken from within the currently defined column boundaries. Any text outside the bounds is not included in the restructuring. The restructured text is also positioned within the current boundaries. If the original text was indented from the left boundary, that indentation is preserved.

Using text flow on a DBCS terminal

You can restructure paragraphs containing lines that include DBCS strings based on these rules:

- If a character in a DBCS string encroaches on the rightmost column position for the restructured text, the string is divided before that character. An SI character is added at the end of the line, and an SO character is added at the beginning of the new line.
- If the boundaries are defined and a DBCS character is on the boundary, the DBCS character is in the text flow operation. An SO or SI character is added to both lines to ensure that DBCS character strings remain enclosed with SO and SI characters.
- If the mask contains DBCS fields and some of the DBCS fields cross the left, right, or both boundaries, the result may be unpredictable.
- If a DBCS string crosses the left, right, or both boundaries, the result may be unpredictable.
- When a text flow operation causes a field length of zero, the SO/SI or SI/SO character strings that are next to each other are removed.

If you use the TF line command while editing a formatted data set, note these points:

- The current boundaries are automatically changed during command processing, and are reset to the original values after processing is complete. Changes are as follows:
  - If the left boundary falls on the second byte of a DBCS character in a DBCS field, the boundary is shifted to the left by 1 byte.
  - If the right boundary does not fall on the same field as the left boundary, it is shifted to the last byte of the field that contains the left boundary. If it falls on the same DBCS field as the left boundary, and it also falls on the first byte of a DBCS character, the right boundary is shifted to the right by 1 byte.
- If you specify the column number with the TF command, and if the column falls on the first byte of a DBCS character in a DBCS field, the column number increases by one.

Splitting lines

The TS (text split) line command splits a line into two lines. The cursor shows where the line is to be split. The editor moves the characters to the right of the cursor or to a new line following the original line and aligns the new line with the left side of the paragraph. As mentioned earlier, the left side of a paragraph is determined by looking for a pattern in the lines preceding or succeeding a paragraph.

If the line being split is the first line in a paragraph, the new line is aligned with the rest of the lines in the paragraph. If there are no other lines in the paragraph, the portion of the line to the right of the cursor aligns itself with the first portion of the line.

One or more blank lines are inserted after the line being split, depending on what you specify when you enter the TS command. Note that the TSPLIT macro command inserts only one blank line.

To rejoin lines, use the TF (text flow) line command. See “Formatting paragraphs” on page 61 for more information.

Splitting lines within a DBCS string

You can split a line within a DBCS string based on these rules:

- When splitting at a DBCS character, an SI character is added to the end of the line and an SO character is added at the beginning of the new line.
- If the cursor is placed at the SO character, the SO character becomes the first character to be moved.
- If the cursor is placed at the SI character, the character following the SI character becomes the first character to be moved.
- If the mask contains DBCS fields and some of the DBCS fields cross the left, right, or both column boundaries, the result is unpredictable.

If you use the TS line command while editing a formatted data set, you make special considerations for the current boundaries. These boundaries are automatically changed during command processing, and are reset to the original values after processing is complete. Changes are as follows:

- If the left boundary falls on the second byte of a DBCS character in a DBCS field, the boundary is shifted to the left by 1 byte.
- If the right boundary does not fall on the same field as the left boundary, it is shifted to the last byte of the field that contains the left boundary. If it falls on the same DBCS field as the left boundary, and it also falls on the first byte of a DBCS character, the right boundary is shifted to the right by 1 byte.

Entering text (power typing)

The TE (text entry) line command allows you to *powertype*. When using this command, the display is filled with blank lines. The line number field normally on the left of the display disappears, so that you can type all of your data as if it were one continuous line. Because the editor is doing the formatting, you can continue typing and ignore the wrap around on the display. Any explicit cursor movement is interpreted as your personal formatting and results in embedded blanks.

The editor assumes that you are typing text as paragraphs. If you explicitly move the cursor down and leave a blank line, the editor assumes that the blank line should be there. The text that follows the blank line is consequently a new paragraph. Similarly, if you leave a specified number of blanks between words, the editor leaves them there. Also, if you tab to the beginning of the next line before completing the current line, the editor does not flow these sentences together. Remember that skipping a line specifies the start of a new paragraph.

Note: You cannot use logical or hardware tabs during text entry.

When you press Enter, the text is flowed in the same manner as the TF (text flow) line command, except that it uses the bounds as the right and left sides of the paragraphs.

Entering text on a DBCS terminal

If you are using the TE line command in a formatted data set, note these points:

- The current boundaries are automatically changed during command processing, and are reset to the original values after processing is complete. Changes are as follows:
  - If the left boundary falls on the second byte of a DBCS character in a DBCS field, the boundary is shifted to the left by 1 byte.
  - If the right boundary does not fall on the same field as the left boundary, it is shifted to the last byte of the field that contains the left boundary. If it falls on the same DBCS field as the left boundary, and it also falls on the first byte of a DBCS character, the right boundary is shifted to the right by 1 byte.
- The attribute of the field where the left boundary falls is used for the text input area attribute. The new input data is reformatted to fit within the current boundaries.

Using tabs

This section discusses hardware, software, and logical tabs, defining and controlling tabs, defining tab positions, and using attribute bytes.

Types of tabs

Software and hardware tabs

The editor uses software and hardware tabs to reposition the cursor within the current display window. You can define tabs with the TABS line command. Use underscores (_) or hyphens (-) to define software tabs and asterisks (*) to define hardware tabs.

Logical tabs

The editor uses logical tabs to reposition strings of data. You can use TABS primary and macro commands, and the TABS assignment statement to define a special character. The tab character locates the beginning of each string. Edit repositions the strings one space to the right of hardware tab positions.

Note:

1. You cannot use the command delimiter that you defined on the Terminal Characteristics panel (option 0.1) as a special tab character.
2. Tabs are not functional when you are using the TE (text entry) line command.

Effect of TABS commands on tab types

If you are using hardware or logical tabs, the TABS line command must be used with one of the other TABS commands or the TABS assignment statement. For example, hardware tab positions defined by the TABS line command do not take effect until tabs mode is turned on, which the line command cannot do. Conversely, a logical tab character defined with the TABS primary or macro command, or the TABS assignment statement, cannot be used to position data strings horizontally unless hardware tab positions are defined with the TABS line command. However, if you are using software tabs, you do not need to turn tabs mode on. The TABS primary and macros commands, and the TABS assignment statement, have no effect on software tabs.

Defining and controlling tabs

Three TABS commands help you to position the cursor where you want to start typing. These commands are the TABS line command, primary command, and macro command. There is also a TABS assignment statement.

You type the TABS line command in the line command field over the line numbers. This command:

- Displays the =TABS> (tab-definition) line
- Defines tab positions for software, hardware, and logical tabs

You type the TABS primary command on the command line. The TABS macro command is processed from within an edit macro. The TABS primary and macro commands can:

- Turn tabs mode on and off
- Define the logical tab character
- Control the insertion of attribute bytes at hardware tab positions that have been defined with the TABS line command

The TABS assignment statement is processed from within an edit macro. It can do everything that the TABS macro command can do. In addition, the TABS assignment statement can retrieve the setting of tabs mode and place it in a variable.

You can use PROFILE to check the setting of tabs mode and the logical tab character.

Defining software tab positions

If you display the =TABS> line and type software tab definitions, they take effect immediately. Each line contains a software tab or a tab field at the designated column positions. The TABS primary command has no effect on software tab definitions.

To define software tab positions:

1. Type TABS in the line command field and press Enter.
2. Type an underscore (_) or a hyphen (-) at each desired column position on the =TABS> line.
3. Press Enter again to start the tabs.

You can move the cursor from one column position to the next by continuing to press Enter. See “Using software and hardware tabs” on page 180 for an example of using software tabs.

Defining hardware tab positions

Hardware tab definitions do not take effect until you turn on tabs mode by using the TABS primary command. The asterisks define the column positions, but the insertion of attribute bytes (hardware tabs) or the repositioning of data strings (logical tabs) does not occur unless tabs mode is on.

To define hardware tab positions:

1. Type TABS in the line command field and press Enter.
2. Type an asterisk (*) at each desired column position on the =TABS> line.
3. Press Enter again.

When tabs mode is turned on using either the ON or ALL operand, the Tab Forward and Tab Backward keys can be used to move the cursor to the space following the next attribute byte.

Note: If the ALL operand is not used, attribute bytes are inserted only in spaces that contain a blank or null character, causing the Tab Forward and Tab Backward keys to recognize only these tab definitions.

When tabs mode is turned on using the *tab-character* operand, the Tab Forward and Tab Backward keys do not recognize hardware tab definitions because no attribute bytes are inserted.

Limiting the size of hardware tab columns

To limit the size of hardware tab columns, type consecutive asterisks between columns to define *hardware tab fields*. The consecutive asterisks:

- Allow you to determine the length of the data string to be typed in a column
- Cause the cursor to automatically move to the next column when the current column is full

This procedure works only with asterisks (hardware tabs). When you type hyphens or underscores (software tabs), PDF does not insert attribute bytes. Because attribute bytes cannot be typed over, they limit the tab column size.

Insert the asterisks from the point where you want the column to end to the point where the next column begins. For instance, suppose you want to limit each tab column to five spaces. You could do so by following these steps:

1. Type COLS in the line command field and press Enter. A partial =COLS> line with positions 9 through 45 is shown in this example:

```
=COLS> -1----+----2----+----3----+----4----+
```

2. Type TABS ALL on the command line and press Enter again. This command causes PDF to insert an attribute byte at each hardware tab position defined by an asterisk (*).
3. Using the TABS line command, change the =TABS> line as follows:

```
=COLS> -1----+----2----+----3----+----4----+
=TABS>           *     *****     *****
```

With the =TABS> line altered as shown, the cursor automatically skips to the next tab column when 5 characters, blank spaces, or a combination of both are typed in each column.

Using attribute bytes

Attribute bytes overlay characters only on the display; the attribute bytes are never recorded in the data. If your data set contains DBCS fields, however, attribute bytes can invalidate them. If you start hardware tabs and insert an attribute byte in the middle of a DBCS field, you invalidate the DBCS field, and it is displayed as an EBCDIC field. When you turn tabs mode off, the attribute bytes are removed and the overlaid character at each tab position is displayed again.

When you are in formatted data edit mode, TABS is ignored.

In tabs mode, you temporarily remove the attribute bytes from a single line. There are two ways to do this:

- Blank out the entire line command field using the Erase EOF key.

- Place the cursor directly under one of the attribute bytes and press Enter. When you press Enter again, the attribute bytes are reinserted.

Undoing edit interactions

If you enter an edit primary, line, or macro command, or type over existing data by mistake, you can restore your data with the UNDO primary command. UNDO has no operands.

Each time you enter UNDO it undoes one interaction. A single interaction might be a data change and Enter key, a data change and function key, or the invocation of an edit macro. All changes caused by an edit macro are considered to be one interaction. You can continue to undo interactions, one at a time, until you have reversed all changes made back to the beginning of your edit session unless you have done a save or undo recycled. If you have done a save or if undo recycled, you can only undo interactions back to that point. At that point, if you enter UNDO again, a message informs you that there are no more interactions to undo.

UNDO has certain limitations. Edit interactions that the command does not undo are:

- Changes that are made by an initial edit macro or recovery edit macro.
- Edit interactions before any data changes are made.
- Edit interactions in previous edit sessions.
- Reset of changed flags (==CHG>) by use of RESET or by typing over the command line area.
- Changes you make to other data sets or members by using the CREATE, REPLACE, or MOVE commands. Because UNDO affects only the member or data set that you are editing, it removes lines from your display if they were inserted there by MOVE. However, it does not put those lines back into the data set or member from which they came.

See "UNDO—Reverse Last Edit Interaction" on page 288 for a discussion of UNDO limitations.

UNDO is reset by SAVE. This means that you can UNDO interactions for the current edit session until you save your data. After the save, you can undo only interactions made following the time you saved your data.

UNDO can be run from data kept in storage or from the recovery file (as in previous releases) depending on what you specify in the Edit Profile for the data you are entering. The SETUNDO primary or macro command is used to control the profile setting. To use UNDO, you must have either RECOVERY on or SETUNDO on. You can undo only those changes made after RECOVERY or SETUNDO was turned on.

SETUNDO allows you to specify how changes you make during your edit session are to be recorded and used by UNDO. You can specify SETUNDO STORAGE (or SETUNDO KEEP or SETUNDO ON) or SETUNDO RECOVER. SETUNDO STORAGE (or SETUNDO KEEP or SETUNDO ON) specifies UNDO from storage. SETUNDO RECOVERY specifies UNDO from recovery and turns recovery on if it is off. See "SETUNDO—Set the UNDO Mode" on page 280 for more details. "Understanding differences in SETUNDO processing" on page 67 explains how the SETUNDO operands differ.

If not enough storage is available to run UNDO from storage but RECOVERY is on, UNDO processing continues to be available by using the recovery file. This makes UNDO available for very large files. It also provides users of machines with less storage with the benefit of UNDO for their larger files.

Note: If you have specified RECOVERY OFF and your installation allows UNDO from storage, the message that UNDO is unavailable does not display when you enter an edit session. If UNDOSIZE = 0, the message appears as before.

The UNDOSIZE specifies the number of kilobytes allowed for saving edit transactions for UNDO and the value is in the configuration table. For more details, refer to *z/OS ISPF Planning and Customizing*.

If UNDOSIZE is set to zero, all undo documented functions work as in ISPF/PDF Version 3.3 and previous releases. This means that the Profile lines do **not** show the status of SETUNDO, and that warning messages will be shown informing you that UNDO is unavailable until RECOVERY is turned on.

UNDO processing

When the storage allocated for changes is exhausted, UNDO *recycles* itself and puts up the message UNDO RECYCLED. Recycling is the process of saving the current image of the file as a new base from which to work. UNDO is then available after the next transaction. No transactions made before the recycling can be undone. This is because UNDO saves an image of the original file and keeps an incremental list of changes to that image.

If there is not enough storage to save the initial image, UNDO attempts to use the recovery file for undo processing. If recovery is off or suspended, the message UNDO SUSPENDED is shown with an alarm, and the profile status line is changed to SETUNDO SUSP. If recovery is available, the message UNDO FROM RECOVERY is shown with an alarm, and the profile status line is changed to SETUNDO REC. This affects the display but does not affect the edit profile values.

To resume SETUNDO STG, enter the SETUNDO primary command. If there is still not enough storage to hold the original copy of the file, the recycling procedure is repeated.

Note: Edit recovery can no longer process edit recovery files created under previous releases of ISPF/PDF. A panel is displayed, but no other action is taken if an old recovery file is used.

Understanding differences in SETUNDO processing

SETUNDO STORAGE (or SETUNDO KEEP or SETUNDO ON) and SETUNDO RECOVERY work essentially the same way; however, there are some important differences. SETUNDO REC is available only after the edit recovery file is initialized, that is, until the first data change is made. Because SETUNDO STG keeps its record of changes in storage, it does not incur the same performance penalty as using the SETUNDO REC.

SETUNDO STG can start to save editing changes earlier than SETUNDO REC, because even non-data changes, such as setting line labels, adding note lines, and inserting blank lines, cause SETUNDO STG to initialize its record of changes. You can undo these changes using UNDO even if no data changes have been made. When SETUNDO REC is in effect, only changes made after and including the first change to edit data can be undone.

UNDO reverses changes made during a single edit transaction. It is important to note, however, that changes to the profile, such as HEX ON, LEVEL, and CAPS, are not undone separately. A data change followed by one or more profile changes is usually considered a single transaction. For example, if you change the data and then the profile, and then enter UNDO, the data and profile return to their status before the data change. Profile changes usually cannot be undone if they are not preceded by a data change. SETUNDO STG and SETUNDO REC may work slightly differently in this regard. Since SETUNDO STG keeps the record of changes in storage, it is not a substitute for recovery. To recover the edit session after a system failure, you must have recovery on during the edit session. SETUNDO STG and RECOVERY ON can be in effect simultaneously, however, after a system crash and a recovery, no transactions can be undone using SETUNDO STG because the in-storage record will be empty.

If you are running both SETUNDO STG and RECOVERY ON, the UNDO command causes the last change to be backed out using the in-storage record of edit changes, and the recovery data set to be reinitialized. If you issue a SETUNDO REC command, after you use UNDO (from storage), there will be no more transactions to UNDO since the recovery file has been reinitialized.

Chapter 4. Using edit models

This topic describes edit models and tells you how to use them.

What is an edit model?

An *edit model* is a predefined set of statements for a dialog element that you can include in the data you are editing and then modify to suit your needs. When you enter the MODEL command, you can select the correct segment for the data type being edited.

ISPF includes an initial set of models for panels, messages, skeletons, and command and program processing of ISPF services. You can add more. There are no models of edit macro commands and assignment statements.

A model has two parts:

Data lines
These are the actual lines that are placed in the data you are editing. For example, the data might be a dialog service call or a panel format. You can update fields in the source statements by inserting names, parameters, and so forth.

The models also include source statement comments for models of dialog service calls to document the meanings of the possible return codes from the service. The comments are in a valid format for the particular kind of model. These comments give you the information you need to develop error-handling logic for your function. Sometimes they provide parameter descriptions for other kinds of models.

Notes
Notes provide tutorial information about how to complete source code statements. You can specify whether you want the notes displayed during the edit session by using the NOTES command or the NOTES or NONOTES operand on the MODEL command. To remove notes from the panel, issue RESET. To convert the notes to data so that they can be saved with your data set, use the MD (make dataline) line command.

How models are organized

Models are organized and named according to a hierarchy based on the type and version of the dialog element they represent. Each part of the model's name corresponds to a level in the hierarchy.

The first part of the logical name is the *model class*. There is a model class for each data set type qualifier that can store a dialog element. The Model Classes panel, Figure 17 on page 70, lists the classes defined for the models distributed with ISPF. This panel prompts you when you need to set the desired model class, if you do not name the class explicitly.

```
                              Model Classes

Enter number or Class of model.
Enter END command to cancel MODEL command.

1  CLIST    - ISPF services in CLIST commands
2  COBOL    - ISPF services in COBOL programs
3  EXEC     - ISPF services in EXEC commands
4  FORTRAN  - ISPF services in FORTRAN programs
5  MSGS     - Message format
6  PANELS   - Panel formats and statements
7  PLI      - ISPF services in PLI programs
8  SKELS    - File tailoring control statements
9  PASCAL   - ISPF services in PASCAL programs
10 REXX     - ISPF services in TSO/REXX commands
11 DTL      - ISPF Dialog Tag Language formats and statements
12 C        - ISPF services in C/370 programs
13 SCLM     - SCLM Project Definition Macros
14 ARCHDEF  - SCLM Architecture Definition templates


Option ===> ____________________________________________________________
 F1=Help      F2=Split     F3=Exit      F7=Backward  F8=Forward   F9=Swap
F12=Cancel
```

Figure 17. Model Classes panel (ISREMCLS)

You can use the default for this part of the logical name whenever the edit profile name matches the class of the model desired.

The second part of the logical name is the model *name*, which identifies the specific model within the model class. Frequently, it uniquely identifies a model and completes the logical name. To uniquely identify a model, you can define optional *qualifiers*. Qualifiers are used, for example, to differentiate among the various kinds of panel verification (VER) statements.

A hierarchy of selection panels defines the hierarchy of models. The different parts of the logical name of a model are selections on the panels that you can choose either by keyword name or option identifier. This allows you to be prompted by selection panels if you do not know the logical name of the model you want or to bypass the display of these panels if you do know the name.

Usually, you do not need to worry about the model class. You must specify it only if you want to use a class that is different from the edit profile name. The model function of the editor recognizes PANELS as a valid type qualifier for panel models, so you do not need to specify the class when requesting a panel model from a data set with a type qualifier of PANELS (assuming you allow the edit profile name to default to panels).

Assume, however, that you call your panels screens and maintain them in a data set with a type of SCREENS. When you want to use a model to develop a new panel, you enter the MODEL command. The model function does not recognize SCREENS as a model class, so you are prompted to identify the class you want, which is the PANELS class in this situation.

Once you have specified a class, whether by panel selection or by use of the MODEL CLASS command, that class remains in effect until you change it. The two ways to change the class specification are by typing a data set name with a different type qualifier, or by leaving the Edit Entry panel.

How to use edit models

You use models to assist you in defining a dialog element. To use a model, first edit your data. Then determine where you want to place the model. If you are editing existing data, define a label or use the A (after) or B (before) line command to show where the model goes. You do not need to use the A or B command when you have a new data set. Then type `MODEL` on the command line and press Enter.

If you know the logical name of the model you want, you can use it to directly access the model. Type `MODEL mmm`, where `mmm` is the name of the model. For example, if you want the model for LMCLOSE, you would specify `MODEL LMCLOSE`. If you enter `MODEL` with no parameters, PDF displays a series of selection panels, from which you select the model name and any qualifiers.

The original data is then displayed with the model in place. You can type over or use line commands to change the data lines in the model to meet your needs.

As an example, assume that you are writing a dialog function using CLIST commands and you want to have the CLIST display a panel. You are editing your CLIST member, called USERID.PRIVATE.CLIST(DEMO1). Since your data set type, CLIST, matches the class of models you want, you can allow the model class to default. If you enter MODEL without a model name, the CLIST Models panel, Figure 18 on page 71, appears.

```
                                CLIST Models

Enter number or service name.
Enter END command to cancel MODEL command.

Variables                 Workstation                 Library Access
V1  VGET                  X1  FILESTAT               L0  LIBACC
V2  VPUT                  X2  FILEXFER
V3  VERASE                X3  WSCON                   Miscellaneous
                          X4  WSDISCON                M0  MISC
Display
D1  DISPLAY
D2  TBDISPL               File Tailoring
D3  SETMSG                F1  FTOPEN
D4  PQUERY                F2  FTINCL
D5  ADDPOP                F3  FTCLOSE
D6  REMPOP                F4  FTERASE

Tables
T1  TABLES

Option ===> ____________________________________________________________
 F1=Help      F2=Split     F3=Exit      F7=Backward  F8=Forward   F9=Swap
F12=Cancel
```

Figure 18. CLIST Models panel (ISREMCMD)

If you select option D1 (DISPLAY), the editor inserts the model for the DISPLAY service in your CLIST, as shown in Figure 19 on page 71. The lines are inserted at the location you specify with a label or an A or B line command. Notes are identified by the characters =NOTE= in the line command field.

```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
──────────────────────────────────────────────────────────────────────
EDIT       LSACKV.PRIVATE.CLIST(EDITOLD) - 01.01          Columns 00001 00072
****** ***************************** Top of Data ******************************
000100   ISPEXEC  DISPLAY  PANEL(PANELNAM)     MSG(MSG-ID)        +
000200                      CURSOR(FIELDNAM)    CSRPOS(POS#)       +
000300                      COMMAND(COMMANDS)   RETBUFFR(BUF-NAME) +
000400                      RETLGTH(LNG-NAME)   MSGLOC(MSG-FIELD)
=NOTE=
=NOTE=      PANELNAM  - OPTIONAL, NAME OF THE PANEL TO BE DISPLAYED.
=NOTE=      MSG-ID    - OPTIONAL, IDENTIFIER OF A MESSAGE TO BE DISPLAYED ON
=NOTE=                  THE PANEL.
=NOTE=      FIELDNAM  - OPTIONAL, NAME OF THE FIELD WHERE THE CURSOR IS TO BE
=NOTE=                  POSITIONED.
=NOTE=      POS#      - OPTIONAL, POSITION OF CURSOR IN FIELD. DEFAULT IS 1.
=NOTE=      COMMANDS  - OPTIONAL, NAME OF A VARIABLE WHICH CONTAINS THE CHAIN
=NOTE=                  OF COMMANDS.
=NOTE=      BUF-NAME  - OPTIONAL, NAME OF A VARIABLE WHICH CONTAINS THE
=NOTE=                  REMAINING PORTION OF THE COMMAND CHAIN TO BE STORED
=NOTE=                  IF AN ERROR OCCURS.
=NOTE=      LNG-NAME  - OPTIONAL, NAME OF A VARIABLE WHICH CONTAINS THE LENGTH
Command ===> ____________________________________________ Scroll ===> PAGE
 F1=Help      F2=Split     F3=Exit      F5=Rfind     F6=Rchange   F7=Up
 F8=Down      F9=Swap     F10=Left     F11=Right    F12=Cancel
```

Figure 19. DISPLAY Service Model

With the notes as a guide, you can edit the CLIST to change the DISPLAY service call parameters for your function. The error-handling source code shown serves as a skeleton which you can update. Finally, use

RESET to eliminate the notes from the panel, leaving the service call, the error-handling logic, and the comments. Some models also include examples in NOTE lines. Use the MD line command to turn NOTE lines into data lines.

Adding, finding, changing, and deleting models

Models are implemented in a general fashion, so your installation can apply and use the concept for other tasks besides dialog development. You can create a set of PL/I call models for your IMS applications, or a set of report format models for your sales forecasting application. You can also create models for the JCL statements that you use most frequently.

Similarly, you may find that the models provided for panel formats do not correspond to the standards for your local installation or for your particular application. You can change the distributed panel models to match your own requirements.

This topic describes how you can add a new model to your skeleton library, change an existing model, or delete an existing model.

Adding models

To create a new model, you must:

1. Determine the data set name and member name for the model. For actual use, the model must be in a skeleton library.
2. Create the source code for the model. Consider whether you should create all new source code or change an existing model under a new name.

   When you create a COBOL model, make sure number mode is on. Then, when you save the model, turn number mode off.
3. Make the model accessible from a model selection panel by having its selection call the program ISRECMBR with the actual model member name as its parameter. This involves:
   - Changing an existing model selection panel to add the new panel.
   - Creating a new model selection panel. If you do this, you must add the new panel to the hierarchy of selection panels by changing one of the higher-level panels.
   - No change, if you are replacing an existing model with an updated model with the same name.
   - Adding the word NOSEQ after the model member name if you wish to check that model data is not being overlaid by editor sequence numbers.

As an example of adding a model, assume that you want to create a model for multiple-line block letters. Since you intend to use these block letters on panels, the model becomes part of the panel model class. To build each model block letter, use the editor to create a new member in your skeleton library. For example, you could create a member called BLKI containing this model for the letter I:

```
        IIIIIIIIII
            II
            II
            II
            II
            II
        IIIIIIIIII
)N
)N   the letter I for logo
```

Once the model for each letter is built, you must update the selection panel in the prompting sequence that deals with panel model selection. This panel is named ISREMPNL and is stored in the system panel library. Figure 20 on page 73 shows the last few lines in ISREMPNL:

```
                              Panel Models
Option ===> _____________________________________________________________

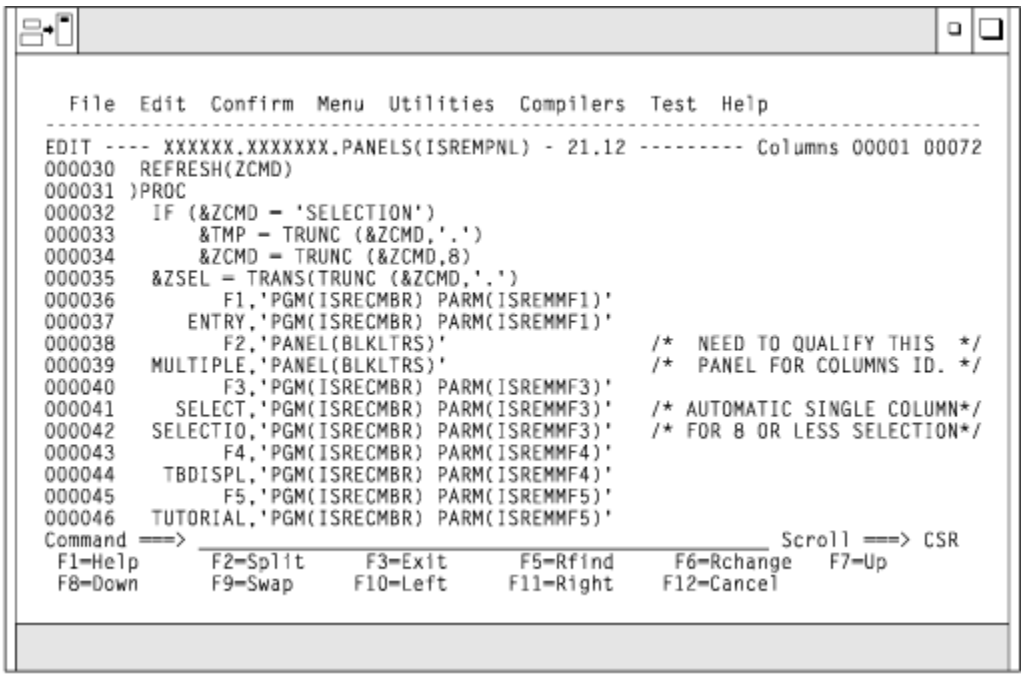
Enter number or statement name.
Enter END command to cancel MODEL command.
                                                                More:   -
⋮
S18 CUAATTR  - CUA attributes
S19 *REXX    - Rexx in panel procedures

P0  PANSECT  - Panel Sections - Other definitions

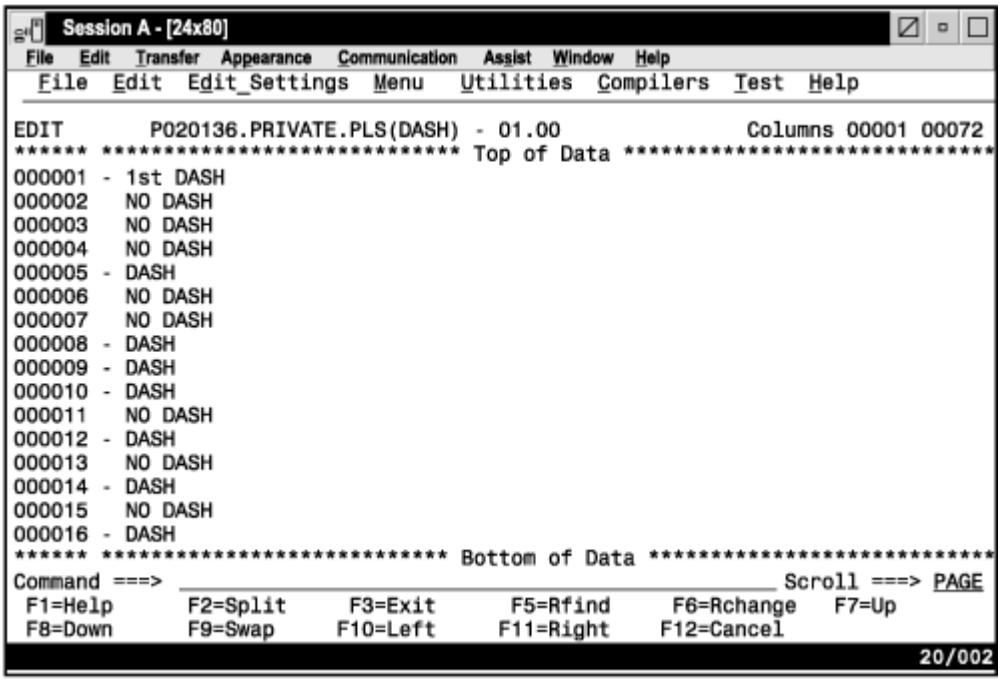
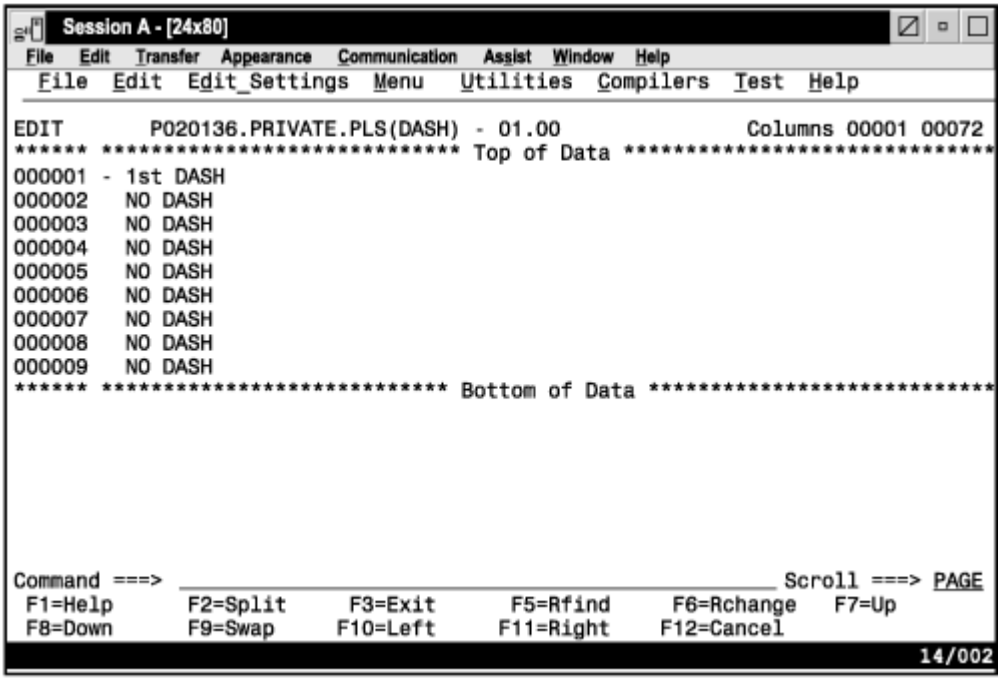
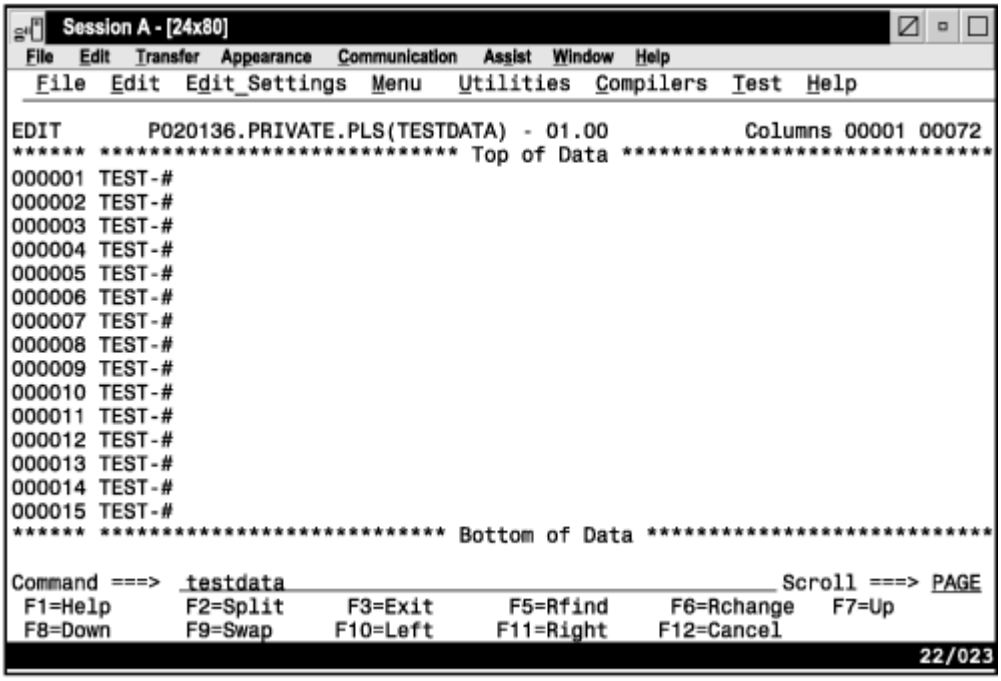
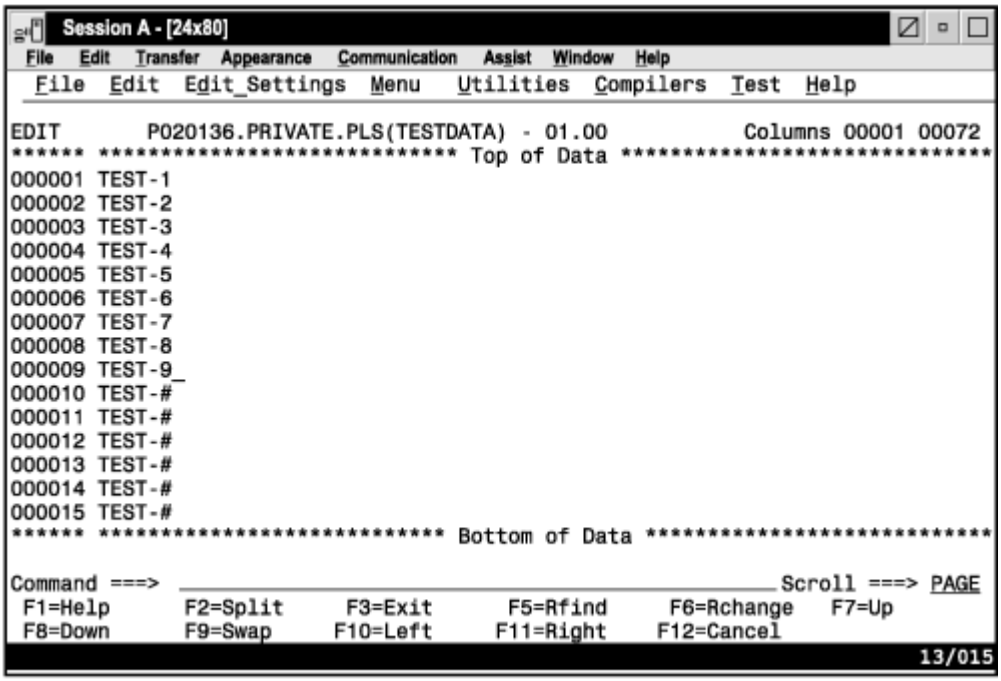
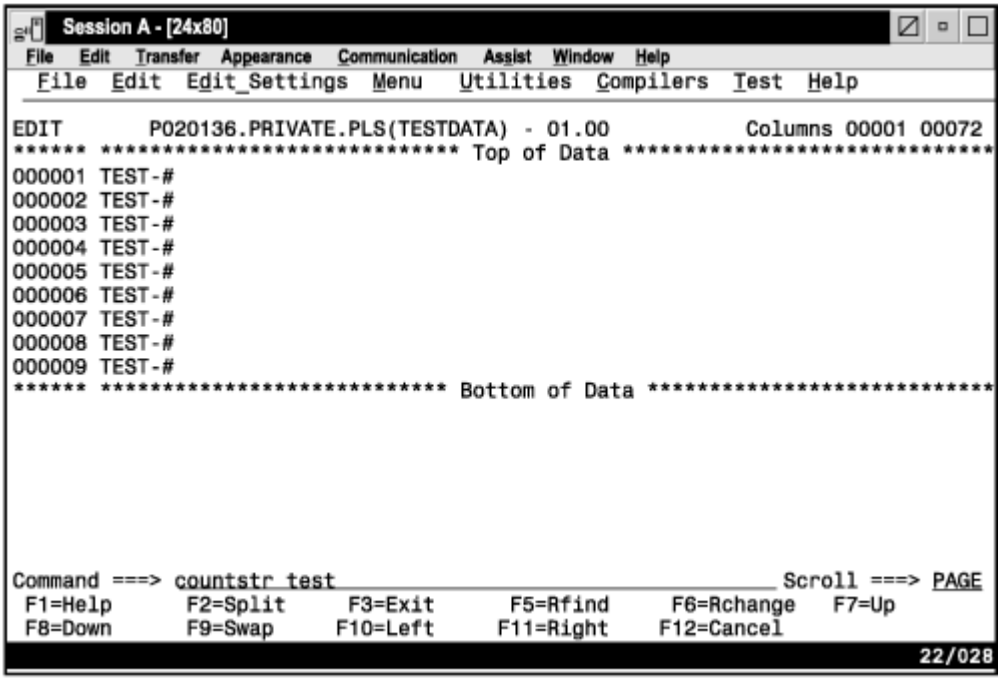
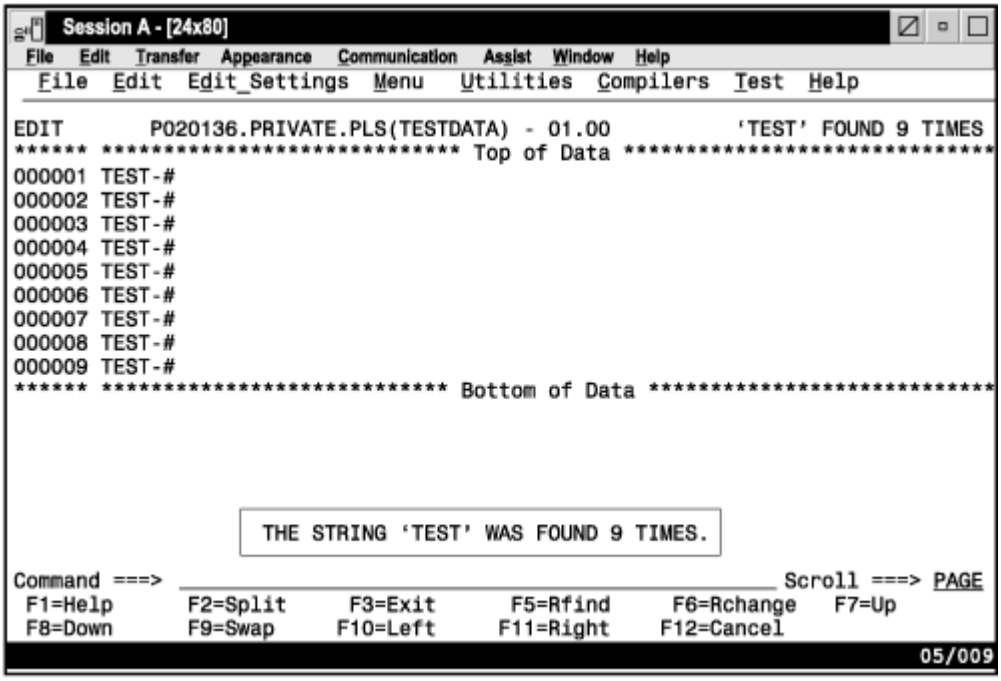
Panel Formats:
F0  PANFORM
 F1=Help      F2=Split     F3=Exit      F7=Backward  F8=Forward   F9=Swap
F12=Cancel
```

Figure 20. Panel Models panel (ISREMPNL)

Copy the panel shown in Figure 20 on page 73 into your panel data set and change it by adding a format F1, BLOCKLTR. See Figure 21 on page 73 for an example.

```
                              Panel Models
Option ===> _____________________________________________________________

Enter number or statement name.
Enter END command to cancel MODEL command.
                                                                More:   -
⋮
S18 CUAATTR  - CUA attributes
S19 *REXX    - Rexx in panel procedures

P0  PANSECT  - Panel Sections - Other definitions

Panel Formats:
F0  PANFORM
F1  BLOCKLTR
 F1=Help      F2=Split     F3=Exit      F7=Backward  F8=Forward   F9=Swap
F12=Cancel
```

Figure 21. Changed Panel Models panel (ISREMPNL)

If there are several new models, this panel should be updated so that when you select F2, a new Block Letter selection panel is displayed. Therefore, you should change the)PROC section of panel ISREMPNL to include item F2. See Figure 22 on page 74 for an example.



Figure 22. Changed)PROC section of Panel Models panel (ISREMPNL)

This concept allows you and other users to have sets of individual models, and allows the installation to have its own set of general models, without having multiple copies of the PDF model selection panels. For each model class, the installation could provide two additional entries on the selection panel: one for installation-wide models and one for your models. Each entry could point to a selection panel, with each user having a copy of the selection panel to customize for individual use.

Note that the entry for F2, BLOCKLTR, points to a new panel, ISRBLOCK, which you would now build.

You can change an existing panel model to create the new panel. Figure 23 on page 75 shows how the new panel might be typed. Note particularly the)INIT and)PROC sections of the coding. In the)PROC section of panel ISRBLOCK, the target for all valid selections is the program ISRECMBR. The parameter passed to this program is different for each separate, but valid, selection and is the name of the model for that selection. Thus, for our example, the model name for selection 1 or I is BLKI.

You should follow the)INIT source code and the end source code in the)PROC section shown in Figure 23 on page 75 for all new panels.

```
)PANEL
/*  ISRBLOCK                                                        */
/*  5647-A01 (C) COPYRIGHT IBM CORP 1995, 2003                      */
/*  Sample source code for the Block Letter Model selection panel.  */
)ATTR
)BODY
%------------------------  BLOCK LETTER  -----------------------

%OPTION  ===>_ZCMD                     +
%
%   1 +I           - Block letter I
%   2 +J           - Block letter J
%   3 +K           - Block letter K
%
%
+
+ Enter %END+command to cancel MODEL command. +

%
)INIT
  .CURSOR = ZCMD
  .HELP = ISRxxxxx
  IF (&ISRMDSPL = 'RETURN  ')
       .RESP = END
)PROC
  &ZSEL = TRANS(TRUNC (&ZCMD,'.')
          1,'PGM(ISRECMBR) PARM(BLKI)'
          I,'PGM(ISRECMBR) PARM(BLKI)'
          2,'PGM(ISRECMBR) PARM(BLKJ)'
          J,'PGM(ISRECMBR) PARM(BLKJ)'
          3,'PGM(ISRECMBR) PARM(BLKK)'
          K,'PGM(ISRECMBR) PARM(BLKK)'
          *,'?' )
  IF (&ZSEL = '?')
     .MSG = ISRYM012
  &ISRMMEND = 'N'                    /* SET THE END INDICATOR TO NO   */
  IF (.RESP = END )                  /* IF ENDING, WHY ... WHO CAUSED */
    IF (&ISRMONCL = 'Y')             /* MAKE SURE ITS NOT A CLASS OP. */
      IF (&ISRMDSPL = 'RETURN  ')    /* MAKE SURE ITS NOT END ON MBR. */
        &ISRMMEND = 'Y'              /* NO - ITS BECAUSE USER HIT END */
)END
```

Figure 23. Source code for Block Letter Model Selection panel

Finding models

Before you change or delete a model, you must determine the physical name of the model in the skeleton library. See *z/OS ISPF Planning and Customizing* for a list of the names of the models of dialog elements distributed with PDF. In addition, you can use the method shown here to find the member name for any model.

You can find the member name for any model in the)PROC section of the final selection panel used to get it. The member name is the parameter passed to ISRECMBR, the program called when you choose that selection.

To determine the name of the model selection panel so that you can look at it to find the model member name, use the PANELID command when that panel is displayed. Then use the Browse or Edit options to look at the member of the panel library with that name.

Changing models

To change a model that currently exists, copy the existing model from the skeleton data set into your own data set. Then use the editor to change the model in the same way you would change any text data set.

Note: Any lines that are to contain notes must have)N in positions 1 and 2, followed by one or more blanks, as shown in this example.

```
    VARIABLE = VALUE
)N        VARIABLE - A DIALOG VARIABLE OR A CONTROL VARIABLE.
)N        VALUE    - A LITERAL VALUE CONTAINING: SUBSTITUTABLE
)N                   VARIABLES, A DIALOG VARIABLE, A CONTROL
```

```
 )N                   VARIABLE, OR AN EXPRESSION CONTAINING A
 )N                   BUILT-IN FUNCTION.
 )N         EXAMPLES:  &DEPT = 'Z59'   &A = &B    &C = ' '
```

When the model is later accessed using MODEL, the lines with)N indicators are flagged with =NOTE= in the line command field (Figure 19 on page 71).

Deleting models

You can delete models by deleting the references to them. To delete the references, remove the entry referencing the model in both the)BODY and)PROC sections of the model selection panel.

Generally, you can leave the model itself in the skeleton library. However, if you are deleting a substantial number of models, you can delete those members from the library and then compress it.

Part 2. Edit macros

Chapter 5. Using edit macros

This topic documents general-use programming interfaces and associated guidance information. It also describes edit macros and describes several examples of their use.

What are edit macros?

You can use edit macros, which look like ordinary editor commands, to extend and customize the editor. You create an edit macro by placing a series of commands into a data set or member of a partitioned data set. Then you can run those commands as a single macro by typing the defined name in the command line or, if you have specified a user line command table to the editor, by entering a user line command in the line command field of one or more lines of the data set.

Edit macros can be either CLISTs or REXX EXECs written in the CLIST or REXX command language, or program macros written in a programming language (such as FORTRAN, PL/I, or COBOL). This documentation uses the CLIST command language for most of its examples, with a few examples in REXX. Examples of program macros are in "Program macros" on page 89.

Edit macros can also contain edit assignment statements that communicate between a macro and the editor. These statements are made up of two parts, keyphrases and values, that are separated by an equal sign. Edit assignment statements are described in "Edit assignment statements" on page 96.

Edit macros have access to the dialog manager and system services. Because edit macros are CLISTs, or REXX EXECs, programs, they have unlimited possibilities.

Note: All edit macros must have an ISREDIT MACRO statement as the first edit command. For more information see "Syntax" on page 366.

You can use edit macros to:

- Perform repeated tasks
- Simplify complex tasks
- Pass parameters
- Retrieve and return information

The remainder of this topic presents examples of these tasks.

Note: To run an edit macro against all members of a PDS you can use a program containing a loop that uses a LMMLIST service to obtain the names of PDS members. For each member issue an ISPEXEC edit command with the initial macro keyword. For an example, see Figure 47 on page 125.

Performing repeated tasks

You can use an edit macro to save keystrokes when you frequently perform a task. A simple example would be using a macro to delete every line that begins with a dash (-) in column 1. You could scan the data and manually delete each line, or you could write a macro that does the same thing much faster. The edit macro in Figure 24 on page 80 processes the commands necessary to delete the lines and requires only that you enter the ISRDASH macro.

```
/********************************************************************/
/*                                                                  */
/* 5647-A01 (C) COPYRIGHT IBM CORP 1995, 2003                       */
/*                                                                  */
/* ISRDASH  Delete lines with a '-' in column 1                     */
/*          except the first '-'                                    */
/*                                                                  */
/********************************************************************/
ISREDIT MACRO
  ISREDIT RESET EXCLUDED        /* Ensure no lines are excluded  */
  ISREDIT EXCLUDE ALL '-' 1     /* Exclude lines with '-' in col1*/
  ISREDIT FIND FIRST '-' 1      /* Show the first such line      */
  ISREDIT DELETE ALL EXCLUDED   /* Delete all lines left excluded*/
EXIT CODE (0)
```

Figure 24. ISRDASH macro

When you run this macro, it deletes all lines beginning with a dash, except the first one. To run the macro, type `isrdash` on the command line (Figure 25 on page 80). The dash macro deletes all lines that began with a dash except the first one (Figure 26 on page 81).



Figure 25. ISRDASH macro - before running



Figure 26. ISRDASH macro - after running

Simplifying complex tasks

If you need to perform an involved task, you can include logic in your edit macro. For instance, the ISRTDATA macro shown in Figure 27 on page 81 creates variations of the same line by first finding the succeeding test string number, and then changing each occurrence, using ascending numbers one through nine.

```
/*********************************************************************/
/*                                                                   */
/* 5647-A01 (C) COPYRIGHT IBM CORP 1995, 2003                        */
/*                                                                   */
/* ISRTDATA generates test data                                      */
/*                                                                   */
/*********************************************************************/
ISREDIT MACRO
  SET &COUNT = 1                    /* Start loop counter         */
  DO WHILE &COUNT <= 9              /* Loop up to 9 times         */
    ISREDIT FIND 'TEST - # '        /* Search for 'TEST-#'        */
    SET &RETCODE = &LASTCC          /* Save the FIND return code  */
    IF &RETCODE = 0 THEN            /*                            */ -
      DO                            /* If the string is found,    */
        ISREDIT CHANGE '#' '&COUNT' /* change '#' to the value    */
        SET &COUNT = &COUNT + 1     /* of '&COUNT', increment     */
      END                           /* the counter by one, and    */
    ELSE                            /* continue the loop.         */ -
      SET &COUNT = 10               /* If the string is not       */
  END                               /* found, set the counter to  */
EXIT CODE (0)                       /* exit the loop.             */
```

Figure 27. ISRTDATA macro

To run the test macro, type `isrtdata` on the command line (Figure 28 on page 82). The macro numbers the first nine lines of data (Figure 29 on page 82).



Figure 28. ISRTDATA macro - before running



Figure 29. ISRTDATA macro - after running

Passing parameters, and retrieving and returning information

You can also write macros to get information from other users and from the editor, and to display messages to other users. The ISRCOUNT macro, as shown in Figure 30 on page 83, finds occurrences of the string TEST from the previous example, counts them, and prepares a return message.

```
/*********************************************************************/
/*                                                                   */
/* 5647-A01 (C) COPYRIGHT IBM CORP 1995, 2003                        */
/*                                                                   */
/* ISRCOUNT counts the number of occurrences of a string, and        */
/*          returns a message.                                       */
/*                                                                   */
/*********************************************************************/
ISREDIT MACRO (PARMSTR)
  ISREDIT SEEK ALL &PARMSTR
  IF &LASTCC > 12 THEN DO
    SET &ZEDSMSG = &STR(SEEK ERROR )
    SET &ZEDLMSG = &STR(STRING NOT FOUND )
  END
  ELSE DO
    ISREDIT (COUNT) = SEEK_COUNTS
    SET &COUNT = &COUNT
    SET &ZEDSMSG = &STR("&PARMSTR" FOUND &COUNT TIMES)
    SET &ZEDLMSG = &STR(THE STRING "&PARMSTR " WAS FOUND +
                      &COUNT TIMES.)
  END
  ISPEXEC SETMSG MSG(ISRZ000)
EXIT CODE (0)
```

Figure 30. ISRCOUNT macro

To run the ISRCOUNT macro, type `isrcount TEST` on the command line (Figure 31 on page 83). The macro does not change the data but displays return messages to show the number of times it found the string. The editor always displays the short message in the upper right corner of the screen. Enter HELP (the default is F1) to produce the long message (Figure 32 on page 84).



Figure 31. ISRCOUNT macro - before running



Figure 32. ISRCOUNT macro - after running

Working with an edit line command table

You can create an edit line command table to store your own line commands. Each line command is associated with a macro that you want to run when you enter the specified line command. The associated macro uses the PROCESS macro statement to determine the lines the command applies to and the destination to be used by the macro.

You can edit an existing line command table to add, delete, or modify your line commands.

When you invoke an Edit or View session, you can specify the name of the edit line command table to be used for that session. For information on specifying the name of the edit line command table on the EDIF, EDIT, VIEW, or VIIF service, refer to the topics describing those services in *z/OS ISPF Services Guide*. For information on specifying the name of the edit line command table on the View Entry Panel or on the Edit Entry Panel, refer to topics "View (option 1) " and "Edit (option 2) " in *z/OS ISPF User's Guide Vol II*.

For each line command you add to the table, you specify:

- The name of the line command you want to add.
- The name of the associated edit macro.
- If it supports a multiple line format. That is, if a numeric suffix can be included on the command to indicate the number of lines that the command applies to.
- If it is a block format. That is, if the command applies to a block of lines.
- If it requires a destination line command as well.

If you have specified an edit line command table to be used in an Edit or View session, when you enter a line command that is in the edit line command table, ISPF invokes the associated macro.

To create a new edit line command table or to edit an existing edit line command table, in the table editor (3.16):

1. Specify the name of the table in the **Table Name** field.
2. Select the **Table is an EDIT line command table** option.
3. Type "E" on the command line and press Enter.

   For a *new* table, ISPF displays the entry fields for the first line command.

a. In the **User Command** field, type the name of the line command you want to create. This can be a 1 to 6 character value. The name must not conflict with any of the ISPF edit internal line commands.

b. In the **MACRO** field, type the name of the program, REXX, or CLIST edit macro that you want to execute when the specified edit line command is entered. The macro can VGET variable ZLMACENT to obtain the edit line command that was entered to run the macro, excluding any suffixes that were entered to indicate a block command. For more information about edit line command tables, see *Line command table support* under *ISPF table utility (option 3.16)* in *z/OS ISPF User's Guide Vol II*.

c. In the **Program Macro** field, type one of these values:

Y
The macro is a program.

N
The specified macro is CLIST or REXX.

d. In the **Block Format** field, type one of these values:

Y
The macro permits a block format for the line command by you repeating the last character of the line command. This is not possible if the line character is 6 characters long.

N
The macro does not permit a block format.

e. In the **Multi line** field, type one of these values:

Y
The macro allows a multiple line format where you can indicate a range of lines by providing a numeric suffix on the line command.

N
The macro does not allow a multiple line format.

f. In the **Dest Used** field, type one of these values:

Y
The line command requires a destination line command as well.

N
The line command does not use a destination line command.

The value Y causes a return code of 8 to be returned by the PROCESS DEST macro command when a destination line command is not specified. See "PROCESS—Process Line Commands" on page 383 for more information.

For an *existing* table, ISPF displays each line command and its associated fields on a separate line.

a. You can modify the details for existing line commands in the table, or add or delete lines by entering any of these commands in the table row selection field:

I
Insert one or more rows after the row where the command was entered.

B
Insert one or more rows before the row where the command was entered.

R
Repeat a row by creating one or more copies of the row where the command was entered.

D
Delete one or more rows.

Note: An optional number from 1 to 9 can be added as a suffix to each of these command characters to cause processing against multiple rows.

4. Press PF3 to save the new or updated table.

Chapter 6. Creating edit macros

This topic documents general-use programming interfaces and associated guidance information.

Edit macros are ISPF dialogs that run in the ISPF editor environment.

CLIST edit macros must be in partitioned data sets in at least one of these concatenations: SYSUPROC, ALTLIB (for data sets activated as CLISTs), or SYSPROC. Data sets in these concatenations can contain CLIST edit macros, REXX edit macros, or a combination of both. However, REXX edit macros in these concatenations must include a REXX comment line (`/* REXX */`) as the first line of each edit macro to distinguish them from CLIST edit macros. This comment line can contain other words or characters if necessary, but it must include the string REXX.

Note: For more information about the ALTLIB concatenation, refer to *z/OS TSO/E Command Reference*.

REXX edit macros must also be in partitioned data sets. Besides the concatenations in the previous list for CLIST edit macros, REXX edit macros can exist in these concatenations: SYSUEXEC, ALTLIB (for data sets activated as EXECs), and SYSEXEC. Data sets in these concatenations can contain only REXX EXECs.

For example, if an application activates an application-level library with these commands:

```
ALTLIB ACTIVATE APPLICATION(EXEC) DA(DS1 DS2 DS3)
ALTLIB ACTIVATE APPLICATION(CLIST) DA(DSA DSB DSC)
```

then data sets DS1, DS2, and DS3 must contain only REXX EXECs. However, DSA, DSB, and DSC can contain either REXX EXECs or CLISTs; if these data sets contain REXX EXECs, the first line of each EXEC must be a REXX comment line.

As in an ISPF dialog, program macros must be made available as load modules in either the ISPLLIB, STEPLIB, or LINKLST library.

CLIST and REXX edit macros

A CLIST edit macro is made up of CLIST statements. A REXX edit macro is made up of REXX statements. Each statement falls into one of these categories:

- Edit macro commands
- CLIST or REXX command procedure statements and comments
- ISPF and PDF dialog service requests
- TSO commands

All statements are initially processed by the TSO command processor, which scans them and does symbolic variable substitution. It is important to recognize the different kinds of CLIST and REXX statements listed because:

- They are processed by different components of the system
- They have different syntax rules and error handling
- Their descriptions are in different manuals

Edit macros are invoked by the editor using the ISPF SELECT service. For REXX macros, the BARRIER keyword is specified to ensure the REXX data stack is preserved across macro invocations.

Edit macro commands and assignment statements

Any statement in an edit macro that begins with ISREDIT is assumed to be an edit macro command or assignment statement. When such a statement is found, the CLIST or REXX command processor does symbolic substitution and then passes it to the editor. The editor processes it, performing any requested functions. Examples of two edit macro commands are:

Table 4. Example edit macro commands

CLIST Statements	REXX Statements

`ISREDIT FIND "TEST475"`
`ISREDIT PROCESS`
	`ADDRESS ISPEXEC`
	`'ISREDIT FIND TEST475'`
	`'ISREDIT PROCESS'`

Examples of two edit macro assignment statements are:

Table 5. Example edit macro assignment statements

CLIST Statements	REXX Statements

`ISREDIT BOUNDS = 1,60`
`ISREDIT (WIDTH) = LRECL`
	`ADDRESS ISPEXEC`
	`'ISREDIT BOUNDS = 1,60'`
	`'ISREDIT (WIDTH) = LRECL'`

A description of each edit macro command and assignment statement is in Chapter 11, “Edit macro commands and assignment statements,” on page 295.

Using the REXX ADDRESS instruction

If you have several edit macro commands within a REXX exec, you can change the command environment to the PDF editor with the instruction ADDRESS ISREDIT. All subsequent commands in the exec are passed directly to the editor. These examples show how you can pass the same edit macro commands using different environments:

Table 6. Passing commands using a different environment

ISPEXEC Environment	ISREDIT Environment

`ADDRESS ISPEXEC`
`'ISREDIT BOUNDS = 1,60'`
`'ISREDIT (WIDTH) = LRECL'`
	`ADDRESS ISREDIT`
	`'BOUNDS = 1,60'`
	`'(WIDTH) = LRECL'`

For information on using the REXX ADDRESS instruction, refer to *z/OS TSO/E REXX Reference*.

Command procedure statements

Command procedure statements handle CLIST and REXX variables and control flow within a CLIST or REXX exec. Command procedure statements are processed by the TSO command processor. Some command procedure statements commonly used in edit macros are:

- Assignment statements
- IF-THEN-ELSE statements
- DO-WHILE-END statements
- EXIT statements

For a complete list and description of command procedure statements for CLIST and REXX, refer to *z/OS TSO/E CLISTs*, *z/OS TSO/E REXX Reference*, and *z/OS TSO/E REXX User's Guide*.

ISPF and PDF dialog service requests

Any statement in an edit macro beginning with ISPEXEC is assumed to be a request for an ISPF service. When such a statement is found, the TSO command processor does symbolic substitution. It then passes

the command to the specified ISPF service to be processed. Some examples of service requests that might be in an edit macro are:

Table 7. Service requests in an edit macro

CLIST Statements	REXX Statements
ISPEXEC SETMSG ... ISPEXEC VPUT ... ISPEXEC DISPLAY ... ISPEXEC EDIT ... ISPEXEC LMINIT ...	ADDRESS ISPEXEC 'SETMSG ...' 'VPUT ...' 'DISPLAY ...' 'EDIT ...' 'LMINIT ...'

For more information on ISPF services, refer to *z/OS ISPF Services Guide*.

TSO commands

Any statement that is not recognized as a command procedure statement and does not begin with ISPEXEC or ISREDIT is assumed to be a TSO command. TSO commands can be either CLISTs, REXX EXECs, or programs. When the command processor finds a TSO command, it processes the command. Examples of TSO commands are:

Table 8. TSO commands

CLIST Statements	REXX Statements
ALLOCATE ... FREE ... DELETE ... RENAME ...	ADDRESS TSO 'ALLOCATE ...' 'FREE ...' 'DELETE ...' 'RENAME ...'

For more information on TSO commands, refer to *z/OS TSO/E Command Reference*.

Program macros

Not all edit macros are written in CLIST or REXX. You can also write edit macros in a programming language such as PL/I, COBOL, FORTRAN, APL2®, Pascal, or C. These are called *program macros*.

There are four basic reasons to write and debug a program macro:

- A macro runs faster in a language that can be precompiled than in CLIST or REXX. This can be valuable for macros that you run many times.
- A macro that must read data containing symbols can confuse an interpretive language processor. Particularly, ampersands in the data can cause problems.
- Complex logic can be handled better in a programming language.
- To pass mixed data or strings (those that contain both EBCDIC and DBCS characters) as parameters, you must use a program macro. Although CLIST does not allow mixed data strings, these edit macro commands and assignment statements allow you to supply data or string operands:

```
CHANGE     EXCLUDE      FIND
LINE       LINE_AFTER   LINE_BEFORE
MASKLINE   SEEK         TABSLINE
```

Differences between program macros, CLISTs, and REXX EXECs

Program macros have special characteristics that you should consider before coding:

- Variables are not self-defining in program macros, as they are in CLISTs and REXX EXECs. The VDEFINE, VCOPY, and VREPLACE dialog services must be called to identify variables looked at or set by the program.
- If you write a REXX exec or a program macro that accepts parameter input, the macro must be aware that the input may be in lowercase. Variable values are automatically converted to uppercase by the CLIST processor.
- Program macros are not implicitly defined, while CLIST and REXX macros are. When you use a command name that is not a built-in or previously defined primary command, the editor searches the SYSUEXEC, SYSUPROC, ALTLIB, SYSEXEC, and SYSPROC concatenations (for CLISTs and REXX EXECs) for a member with the same name. If it exists, it is assumed to be a macro.

  No automatic search is done for program macros. Therefore, there are two ways to tell the editor to run a macro as a program macro. You can precede the name with an exclamation point (`!`) if it is less than 8 characters, or you can use the DEFINE command to define the name as a program macro. Program macros are treated as ISPF dialogs, and must be made available as load modules in either the ISPLLIB, STEPLIB, or LINKLST library.
- Program macros can run without being verified as macros; the MACRO statement can follow calls to dialog services.
- The editor scans edit statements within program macros to do variable substitution similar to the CLIST processor. Only one level of substitution is done. This is the default; use the SCAN assignment statement to prevent it.

Passing parameters in a program macro

Program macros process edit commands by using the ISPLINK or ISPEXEC interface. ISPLNK and ISPEX are the interface names used in FORTRAN and Pascal programs. Parameters are passed to the ISREDIT service as follows:

- ```
  CALL ISPLINK ('ISREDIT',length,buffer)
  ```

  ```
  CALL ISPEXEC (length,'ISREDIT command')
  ```

where these definitions apply:

'ISREDIT'
The service name.

length
A fullword number indicating the length of the command buffer. When a zero length is passed, the maximum buffer length is 255 bytes.

buffer
Can contain any edit command that is valid from a macro, typed with the same syntax used in a CLIST or REXX exec.

command
Any PDF edit command that is valid from a macro, typed with the same syntax used in a CLIST or REXX exec.

Program macro examples

These examples show three different methods of coding a FIND command for a program macro. They are typed using PL/I syntax:

```
CALL ISPLINK ('ISREDIT',LEN0,'¢FIND XYZ¢')
CALL ISPLINK ('ISREDIT',LEN8,'FIND XYZ')
CALL ISPEXEC (LEN16,'ISREDIT FIND XYZ')
```
```

where:

- **LEN0**
  A fullword program variable with a value of 0.

  LEN8
  A fullword program variable with a value of 8.

  LEN16
  A fullword program variable with a value of 16.

In each of these examples, the rest of the command is typed as a literal value.

The first two examples use the ISPLINK syntax. In the ISPLINK call, ISREDIT is passed as the first parameter and is omitted from the command buffer.

The first example uses a special interface. A zero length can be passed, but only when the command is delimited by a special character. A special character cannot be an alphanumeric character. If the length is zero and if a valid delimiter is the first character in the command buffer, a scan of the command is done to find the next occurrence of that character. The command length is the number of characters between the two delimiters. Here, the cent sign (¢) is used as a delimiter. When a zero length is passed, the maximum buffer length is 255 bytes.

In the second example, an explicit length of 8 is used and the command buffer contains the command without delimiters.

The third example uses the ISPEXEC syntax. This syntax always requires the length of the command buffer to be passed. The command buffer includes the ISREDIT prefix, and is typed the same way as a CLIST or REXX command.

Writing program macros

When you write a program macro, it can help to first type it as a CLIST or REXX macro to debug the logic and the command statements. The example that follows is a simple macro that separates each line in a set of data with a line of dashes. The REXX version, called ISRSLREX, is shown in Figure 33 on page 92. The PL/I program is shown in Figure 34 on page 93, and the COBOL program is shown in Figure 35 on page 94. Notice that a VDEFINE is not required for the variable SAVE, which is referenced only by the ISPF editor.

```
/* Rexx ************************************************************/
/***** Sample Edit Macro ********************************************/
/*                                                                  */
/* 5647-A01 (C) COPYRIGHT IBM CORP 1995, 2003                       */
/*                                                                  */
/* ISRSLREX - separates lines with a line of dashes.                */
/*                                                                  */
/********************************************************************/
TRACE
ADDRESS ISPEXEC
'ISREDIT MACRO'

   'ISREDIT (SAVE) = USER_STATE'
   'ISREDIT RESET'
   'ISREDIT EXCLUDE ----- 1 ALL'
   'ISREDIT DELETE ALL X'
   LASTL = 1
   LINE = 0
   LINX = COPIES('-',70)

   LL = LASTL + 1
   DO WHILE LINE < LL
     'ISREDIT LINE_AFTER 'LINE' = (LINX)'
     'ISREDIT (LASTL) = LINENUM .ZLAST'
     LL = LASTL + 1
     LINE = LINE + 2
   END
   'ISREDIT USER_STATE = (SAVE)'
EXIT
```

Figure 33. ISRSLREX REXX macro

```
/*                                                                   */
/* 5647-A01 (C) COPYRIGHT IBM CORP 1995, 2003                        */
/*                                                                   */
/* ISRSEPP - EDIT MACRO PROGRAM TO INSERT SEPARATOR LINES            */
/*           PL/I                                                    */
/*                                                                   */
ISRSEPP: PROC OPTIONS (MAIN);
/*                                                                 */
DECLARE
  LINEX CHAR (70) INIT ((70)'-') ,     /* SEPARATOR LINE ---  */
  LASTL FIXED BIN(31,0) INIT (0),      /* LAST LINE OF TEXT   */
  LINE  FIXED BIN(31,0) INIT (0),      /* CURRENT LINE NUMBER */
  LEN0  FIXED BIN(31,0) INIT (0),      /* LENGTHS - 0         */
  LEN1  FIXED BIN(31,0) INIT (1),      /* LENGTHS - 1         */
  LEN4  FIXED BIN(31,0) INIT (4),      /* LENGTHS - 4         */
  LEN70 FIXED BIN(31,0) INIT (70);     /* LENGTHS - 70        */
                                       /*                     */
DECLARE                                /*                     */
  ISPLINK ENTRY OPTIONS(ASM,INTER,RETCODE); /* LINK TO ISPF   */
                                       /*                     */
  CALL ISPLINK('VDEFINE','(LASTL)',LASTL,'FIXED',LEN4);
  CALL ISPLINK('VDEFINE','(LINE)', LINE, 'FIXED',LEN4);
  CALL ISPLINK('VDEFINE','(LINEX)',LINEX,'CHAR', LEN70);

  CALL ISPLINK('ISREDIT',LEN0,'¢ MACRO ¢');
  CALL ISPLINK('ISREDIT',LEN0,'¢ (SAVE) = USER_STATE ¢');
  CALL ISPLINK('ISREDIT',LEN0,'¢ RESET ¢');
  CALL ISPLINK('ISREDIT',LEN0,'¢ EXCLUDE ------ 1 ALL ¢');
  CALL ISPLINK('ISREDIT',LEN0,'¢ DELETE ALL X ¢');

  LASTL = 1;
  LINE = 0;

DO WHILE (LINE < (LASTL + 1));
  CALL ISPLINK('ISREDIT',LEN0,'¢ LINE_AFTER &LINE = (LINEX) ¢    ');
  CALL ISPLINK('ISREDIT',LEN0,'¢ (LASTL) = LINENUM .ZLAST ¢');
  LINE = LINE + 2;
END;

  CALL ISPLINK('ISREDIT',LEN0,'¢ USER_STATE = (SAVE) ¢');

END IISRSEPP;
```

Figure 34. ISRSEPP PL/I macro

```
       ID DIVISION.
       PROGRAM-ID. ISRSEPC.
      *
      *           EDIT MACRO PROGRAM TO INSERT SEPARATOR LINES
      *
       ENVIRONMENT DIVISION.
       DATA DIVISION.
       WORKING-STORAGE SECTION.

       01  LINEX   PIC  X(70) VALUE  ALL  "-".
      *   SEPARATOR LINE ------
       01  LASTL   PIC  9(6)  VALUE    0  COMP.
      *   LAST LINE OF TEXT
       01  LYNE    PIC  9(6)  VALUE    0  COMP.
      *   CURRENT LINE NUMBER

       01  ISREDIT PIC  X(8)  VALUE  "ISREDIT ".
       01  VDEFINE PIC  X(8)  VALUE  "VDEFINE ".
       01  ZLASTL  PIC  X(8)  VALUE  "(LASTL )".
       01  ZLINE   PIC  X(8)  VALUE  "(LINE  )".
       01  ZLINEX  PIC  X(8)  VALUE  "(LINEX )".
       01  FIXED   PIC  X(8)  VALUE  "FIXED   ".
       01  CHAR    PIC  X(8)  VALUE  "CHAR    ".
       01  LEN0    PIC  9(6)  VALUE    0  COMP.
       01  LEN4    PIC  9(6)  VALUE    4  COMP.
       01  LEN70   PIC  9(6)  VALUE   70  COMP.

       01  EM1     PIC  X(10) VALUE  "¢  MACRO ¢".
       01  EM2     PIC  X(24) VALUE  "¢  (SAVE) = USER_STATE ¢".
       01  EM3     PIC  X(10) VALUE  "¢  RESET ¢".
       01  EM4     PIC  X(25) VALUE  "¢  EXCLUDE ------ 1 ALL 0".
       01  EM5     PIC  X(18) VALUE  "¢  DELETE ALL X ¢".
       01  EM6     PIC  X(30) VALUE  "¢ LINE_AFTER &LINE = (LINEX) ¢".
       01  EM7     PIC  X(28) VALUE  "¢ (LASTL) = LINENUM .ZLAST ¢".
       01  EM8     PIC  X(23) VALUE  "¢ USER_STATE = (SAVE) ¢".

       PROCEDURE DIVISION.
           CALL  "ISPLINK" USING  VDEFINE  ZLASTL  LASTL  FIXED LEN4.
           CALL  "ISPLINK" USING  VDEFINE  ZLINE   LYNE   FIXED LEN4.
           CALL  "ISPLINK" USING  VDEFINE  ZLINEX  LINEX  CHAR  LEN70.
           CALL  "ISPLINK" USING  ISREDIT  LEN0    EM1.
           CALL  "ISPLINK" USING  ISREDIT  LEN0    EM2.
           CALL  "ISPLINK" USING  ISREDIT  LEN0    EM3.
           CALL  "ISPLINK" USING  ISREDIT  LEN0    EM4.
           CALL  "ISPLINK" USING  ISREDIT  LEN0    EM5.

           MOVE  1 TO LASTL.
           MOVE  0 TO LYNE.
           PERFORM LOOP UNTIL LYNE IS NOT LESS THAN (LASTL + 1).
           CALL  "ISPLINK" USING  ISREDIT  LEN0    EM8.
           GOBACK.
       LOOP.
           CALL  "ISPLINK" USING  ISREDIT  LEN0    EM6.
           CALL  "ISPLINK" USING  ISREDIT  LEN0    EM7.
           ADD 2 TO LYNE.
```

Figure 35. ISRSEPC COBOL macro

Running program macros

The ISPF editor assumes that any unknown primary command is a macro, and it also assumes that the macro has been implemented as a CLIST or REXX exec. You can define a macro as a program macro either by entering a DEFINE command or by prefixing the macro name with an exclamation point (!) when you type the macro name on the command line.

If a macro named FINDIT is a CLIST or REXX exec macro, for example, you can run it by typing FINDIT on the command line and pressing Enter. If it is a program macro, you can type !FINDIT, or FINDIT if it had previously been defined as a program macro by the DEFINE command. The first time you enter a macro with an exclamation point (!) prefix implicitly defines that macro as a program macro. Thereafter, you can omit the prefix.

To use the DEFINE command to define a program as a macro, type this command and press Enter:

```
DEFINE name PGM MACRO
```

The operands can be typed in either order. That is, `DEFINE name MACRO PGM` is also valid.

Using commands in edit macros

You can use most primary commands in an edit macro if you precede it with ISREDIT. Table 18 on page 295 shows the macro commands available to use. There are differences, though, between entering a command on the command line and processing the same command in a macro as one of a series:

- When you enter a command on the command line, the result of the command is displayed in either an informational or an error message. If you process the same command in a macro, messages are not displayed, and the lines actually displayed may be different from a command entered on the command line.
- When you issue a series of commands as a macro, the display does not change with each command. The lines displayed are the end result of the macro running, not the individual commands.
- Some commands have additional operands permitted in a macro that cannot be used interactively.

Besides these differences, there are certain guidelines to remember when creating edit macros. These topics apply to CLIST, REXX, and program macros.

Naming edit macros

Edit macro names can be any valid CLIST, REXX, or program name. Using the DEFINE ALIAS command, you can assign command names for running the edit macros that are different from the actual name.

When choosing names and aliases, avoid defining names that might conflict with the DEFINE command operands and their abbreviations. You can do this by implicitly defining the macros: precede program macros with an exclamation point (!); do not use explicit definitions for CLIST or REXX macros.

Variables

Variables function in edit macros in the same way as in CLISTs and REXX EXECs. The only exceptions are dialog variables—variables that communicate with ISPF and the PDF component—which can only have names from 1 to 8 characters in length. This topic presents a brief introduction on using variables; for more detailed information on variables in CLISTs, refer to *z/OS TSO/E CLISTs*. For information on variables in REXX EXECs, refer to *z/OS TSO/E REXX Reference* and *z/OS TSO/E REXX User's Guide*.

When coding macros in CLIST or REXX, remember that all ISREDIT statements are processed for variable substitution before the editor sees the statements. Enclose the variables in parentheses when variable substitution should not occur, such as in cases when ISREDIT statements expect a variable name and not its value. For CLIST variables, omit the ampersand; for REXX variables, use quotes.

Variable substitution

Scan mode controls the automatic replacement of variables in command lines passed to the editor. Use the SCAN assignment statement either to set the current value of scan mode (for variable substitution), or to retrieve the current value of scan mode and place it in a variable.

When scan mode is on, command lines are scanned for ampersands (&). If an ampersand followed by a nonblank character is found, the characters between the ampersand and the next blank or period are treated as the name of a dialog variable. The value from the variable pool is substituted in the command for the variable name before the command is processed. For example, &DVNAME. and &DVNAME are both interpreted as a dialog variable called DVNAME.

The period after the variable allows concatenation of the variable value without an intervening blank delimiter. Remember this when using program macros that do not have the CLIST processor to substitute variable values.

Character conversion

A CLIST automatically converts all character strings to uppercase before passing them to the editor. Therefore, if you want an edit macro command or assignment statement that you process from a CLIST to find a character string in lowercase, you must precede the command or statement with the TSO CONTROL ASIS statement. This statement passes lowercase characters to the editor.

Edit assignment statements

You use edit assignment statements to communicate between macros and the editor. An assignment statement consists of two parts, *values* and *keyphrases*, which are separated by an equal sign. The value segment represents data that is in the macro, and the keyphrase segment represents data in the editor. You can use assignment statements to pass data from the edit macro to the editor, or to transfer data from the editor to the edit macro.

Data is always transferred from the right side of the equal sign in an assignment statement to the left side. Therefore, if the keyphrase is on the right, data known to the editor is put into CLIST or REXX variables on the left. In this situation, the `yyy` would be a keyphrase, and the `xxx` would be the value.

Table 9. Edit assignment statements

CLIST Statement	REXX Statements
`ISREDIT  xxx = yyy`	`ADDRESS ISPEXEC` `'ISREDIT xxx = yyy'`

Value

The value part of an edit macro assignment statement can be:

- A *literal* character string can be one of these types:

  Simple
  Any series of characters not enclosed within quotes (either ' or "), parentheses, or less-than (<) and greater-than signs (>), and not containing any embedded blanks or commas.

  Delimited
  Any string starting and ending with a quote (either ' or "), but not containing embedded quotes. The delimiting quotes are not considered to be part of the data.

- A *dialog variable name* enclosed in parentheses (varname). If the dialog variable name is on the right, the entire contents of the variable are considered part of the data, including any quotes, apostrophes, blanks, commas, or other special characters. If the dialog variable name is on the left, its content is totally replaced.

  Note:

  1. In the CLIST environment, the CLIST variable pool and the dialog function variable pool are merged. Therefore, variables in parentheses are the same as ampersand variables, except that the editor does the symbolic substitution rather than the CLIST processor.
  2. In the REXX environment, the REXX variable pool and the dialog function variable pool are also merged. Therefore, quoted variable names in parentheses are the same as unquoted variable names, except that the editor does the symbolic substitution rather than the REXX processor.
  3. In a program macro, you must use the VDEFINE service for any variables that are passed to the editor.

Keyphrase

A keyphrase is either a single keyword, or a keyword followed by a line number or label. The keyphrase can be either a single-valued keyphrase or a double-valued keyphrase.

Keyphrase syntax

Single-valued keyphrases can have this syntax:

```
ISREDIT keyphrase = keyphrase
ISREDIT keyphrase = value
ISREDIT keyphrase = keyphrase + value
ISREDIT keyphrase = value + value
```

Double-valued keyphrases can have this syntax:

```
ISREDIT (varname,varname) = keyphrase
ISREDIT keyphrase = value-pair
```

where *value-pair* is one of these:

- Two literals, which can be separated by a comma or blank. For example:

Table 10. Separating two literals

CLIST Statements	REXX Statements

`ISREDIT CURSOR = 1,40`
`ISREDIT CURSOR = 1 40`
	`ADDRESS ISPEXEC`
	`'ISREDIT CURSOR = 1,40'`
	`'ISREDIT CURSOR = 1 40'`

  Apostrophes or quotes cannot be used when specifying two numeric values. All of these, for example, are incorrect:

  CLIST Statements
  REXX Statements
  `ISREDIT CURSOR = '1','40' ISREDIT CURSOR = '1,40'`

```
ADDRESS ISPEXEC
"ISREDIT CURSOR = '1','40'"
"ISREDIT CURSOR = '1,40'"
```

- Two variable names enclosed in parentheses and separated by a comma or blank, where each variable contains a single value:

  (*varname,varname*) or (*varname varname*)

In any edit assignment statement containing a two-valued keyphrase, either of the variables or values in a pair can be omitted. The general syntax then becomes:

```
ISREDIT  (varname) = keyphrase
ISREDIT  keyphrase = single-value
ISREDIT  (,varname) = keyphrase
ISREDIT  keyphrase = ,single-value
```

Note: Even though you can use blanks instead of commas to separate paired variables or values, you must use a leading comma whenever the first variable or value has been omitted.

Overlays and templates

The transfer of information from one side of the equal sign to the other can involve combining several variables or values. This transfer is called an *overlay*. When you perform overlays, there are certain guidelines to remember.

When two values (or a keyphrase and a value) are on one side of an equal sign and separated by a plus sign (+), only nonblank characters in the value on the right overlay corresponding positions in the value on the left. For example:

CLIST statements

```
ISREDIT LINE .ZCSR = LINE + '//'
ISREDIT MASKLINE = MASKLINE + <40 '&STR(/*)' 70 '&STR(*/)'>
```

REXX statements

```
ADDRESS ISPEXEC
"ISREDIT LINE .ZCSR = LINE + '//'"
"ISREDIT MASKLINE = MASKLINE + <40 '/*' 70 '*/'>"
```

The first example causes two slashes to replace the first two column positions of the current line (the line containing the cursor). The remainder of the line is unchanged. The second example uses a *template* to cause columns 40-41 of the current mask line to be replaced with /* and columns 70-71 to be replaced with */. Again, remember that the template replaces the corresponding positions on the left only if those left positions are blank. The template shown in the preceding example has the form:

```
<col1 literal1 col2 literal2 ... >
```

It can be designed with *col1* and *col2* indicating a starting column position, and *literal1* and *literal2* indicating the data to start in that column. The entire template is delimited with less-than (<) and greater-than (>) signs. A template can be designed by using variable names (enclosed in parentheses) for either *col1*, *col2*, *literal1*, *literal2*, or for all four. All of these forms are valid:

```
<(colvar1) (datavar1) (colvar2) (datavar2) ... >
<(colvar1,datavar1)   (colvar2,datavar2)   ... >
<(colvar1)  literal1   col2     (datavar2) ... >
```

Using edit assignment statements

You can use an assignment statement to pass edit parameters to a macro or to allow a macro to set an edit parameter. If the edit parameter keyphrase is on the right of the assignment statement, the edit parameter is passed to the macro. If the edit parameter keyphrase is on the left of the assignment statement, the edit parameter is changed to the value on the right. In the assignment statement shown, the edit parameter keyphrase is CAPS. The editor assigns the current CAPS edit mode status (ON or OFF) to the variable CAPMODE.

Table 11. Assigning a value to a variable

CLIST Statement	REXX Statements
`ISREDIT (CAPMODE) = CAPS`	`ADDRESS ISPEXEC` `'ISREDIT (CAPMODE) = CAPS'`

In the preceding example statements, the parentheses around CAPMODE indicate to the ISPF editor that the enclosed name is the name of a symbolic variable. If the name happened to be preceded by an ampersand (&), rather than enclosed in parentheses, the CLIST processor would replace the name of the variable with its actual value, and the editor would not see the name. In a REXX statement, the variable name must be within quotes so that the name, not the value, is passed. Only names with 8 or fewer characters are allowed by the ISPF editor.

When the editor finds a variable name in parentheses in a position where a value is required, it substitutes the value assigned to that variable. In these examples the edit macro sets the edit CAPS mode:

Table 12. Substituting a value in a variable

CLIST Statements	REXX Statements

`ISREDIT CAPS = ON`
`ISREDIT CAPS = (CAPMODE)`
`ISREDIT CAPS = &CAPMODE`
	`ADDRESS ISPEXEC`
	`'ISREDIT CAPS = ON'`
	`'ISREDIT CAPS = (CAPMODE)'`
	`'ISREDIT CAPS = '`*capmode*

The CLIST and REXX command processors replace the variable CAPMODE with its assigned value before the ISPF editor processes the statement. This makes the last statement equivalent to the first statement; in this case, the variable has a value of ON.

The second statement differs in that the editor receives the variable name and retrieves its value from the dialog variable pool.

Passing values

Some information can best be passed back and forth between the editor and the macro in pairs. These examples show assignment statements that pass two values:

CLIST Statements	REXX Statements

`ISREDIT (LB,RB) = BOUNDS`
`ISREDIT BOUNDS = (LB,RB)`
	`ADDRESS ISPEXEC`
	`'ISREDIT (LB,RB) = BOUNDS'`
	`'ISREDIT BOUNDS = (LB,RB)'`

In the first statement, the current left and right boundaries are stored into the variables LB (LEFTBND) and RB (RIGHTBND). In the second statement, the values from the variables LB and RB are used to change the current boundaries.

For more information on which edit macro commands take one variable and which take two, see Chapter 11, “Edit macro commands and assignment statements,” on page 295.

Manipulating data with edit assignment statements

You can use assignment statements to obtain, replace, or add data being edited.

To copy a line, use:

CLIST Statement	REXX Statements
`ISREDIT LINE_AFTER 5 = LINE 2`	`ADDRESS ISPEXEC` `'ISREDIT LINE_AFTER 5 = LINE 2'`

To copy line 1 from the data set into the variable LINEDATA, use:

CLIST Statement	REXX Statements
`ISREDIT (LINEDATA) = LINE 1`	`ADDRESS ISPEXEC` `'ISREDIT (LINEDATA) = LINE 1'`

To replace the first line in the data set, using the data from the variable LINEDATA, use:

CLIST Statement	REXX Statements
`ISREDIT LINE 1 = (LINEDATA)`	`ADDRESS ISPEXEC` `'ISREDIT LINE 1 = (LINEDATA)'`

To add a new line after line 1 in the data set using the variable NEWDATA, use:

CLIST Statement	REXX Statements
`ISREDIT LINE_AFTER 1 = (NEWDATA)`	`ADDRESS ISPEXEC` `'ISREDIT LINE_AFTER 1 = (NEWDATA)'`

Differences between edit, CLIST, and REXX assignment statements

- Edit assignment statements are preceded by ISREDIT. CLIST assignment statements are preceded by SET. If the ADDRESS ISREDIT command is in effect, edit assignment statements within a REXX exec do not need to be preceded by ISREDIT.
- In edit assignment statements, a keyphrase must appear on either the left or right side of the equal sign. A keyphrase is either a single keyword, or a keyword followed by a line number or label. See "Keyphrase" on page 96 if you need more information.
- When coding edit assignment statements, variable names to be passed to the editor are enclosed in parentheses so that the PDF component is passed the name of the variable, not its value. Sometimes two variable names may appear within the parentheses.
- Arithmetic expressions are not allowed in an edit assignment statement, but in certain cases a plus sign (+) can be used to show partial overlay of a line. See "Overlays and templates" on page 97 if you need more information.

Performing line command functions

You cannot issue line commands directly from an edit macro. For example, you cannot use the M (move) line command within an edit macro.

However, you can perform most of the functions provided by line commands by writing an edit macro. By using edit assignment statements or by issuing primary commands, you can perform functions such as move, copy, or repeat. For example, if you want to move a line, you can assign the line to a CLIST or REXX variable, delete the original line using the DELETE command, and assign the variable to a new line in the data.

Some commands can be processed only from within a macro. These commands provide functions done with line commands from the keyboard. Table 13 on page 100 identifies the commands, the corresponding line commands, and the functions performed.

Table 13. Edit macro commands corresponding to line commands

Edit Macro Statement	Corresponding Line Command	Function
INSERT	I	Inserts temporary lines
SHIFT (	(	Shifts columns left
SHIFT)	)	Shifts columns right
SHIFT <	<	Shifts data left
SHIFT >	>	Shifts data right
TENTER	TE	Starts text entry mode
TFLOW	TF	Performs text flow
TSPLIT	TS	Performs text split

For example:

CLIST Statement	REXX Statements
`ISREDIT TFLOW 1`	`ADDRESS ISPEXEC` `'ISREDIT TFLOW 1'`

causes the paragraph starting on line 1 to be flowed in the same way as a TF (text flow) line command would if entered on the first line.

For more information on line command functions in edit macros, see Chapter 11, “Edit macro commands and assignment statements,” on page 295.

Parameters

If you want to supply information to a macro as parameters, you must identify these parameters on the ISREDIT MACRO statement by enclosing them in parentheses. For example, if you have this macro command in an edit macro named FIXIT:

CLIST Statement	REXX Statements
`ISREDIT MACRO (MEMNAM)`	`ADDRESS ISPEXEC` `'ISREDIT MACRO (MEMNAM)'`

when you enter:

```
Command ====> FIXIT ABCD
```

the value ABCD is assigned to the variable MEMNAM.

Passing parameters to a macro

A parameter can be either a simple string or a quoted string. It can be passed by using the standard method of putting variables into shared and profile pools (use VPUT in dialogs and VGET in initial macros). This method is best suited to parameters passed from one dialog to another, as in an edit macro.

You can enter parameters along with an edit macro name as a primary command by using the MACRO command. This command allows you to identify the names of one or more variables to contain any passed parameters.

Note: For edit line macros, only one parameter is passed to the macro. This parameter is the line command, including any repetition, as it was entered on the line.

For more information, see “Working with an edit line command table” on page 84.

It is an error to enter parameter values for a macro without parameter variables. If you make this mistake, the editor displays a message. It is not an error if you supply more or fewer parameters than the number of variables that are included on the MACRO command. When you are writing a macro, check for omissions and the order of parameters.

Multiple parameters are placed into one or more variables based on the number of variables specified in the MACRO command. If you include more than one variable name, the editor stores the parameters in order (the first parameter in the first variable, the second in the second, and so on). Note that assignment to variables is by position only.

If there are more parameters entered than there are variables available, the editor stores the remaining parameters as 1 character string in the last variable. If you include only one variable name on the MACRO command, that variable contains all the parameters entered with the macro name. If there are more variable names than parameters, the unused variables are set to nulls.

Multiple parameters are separated by a blank or comma, or a quoted string that is separated by a blank or comma. Quotes can be single (') or double ("). If you want your FIXIT macro to accept two parameters, for example, you can include this command:

CLIST Statement	REXX Statements
`ISREDIT MACRO (PARM1,PARM2,REST)`	`ADDRESS ISPEXEC` `'ISREDIT MACRO (PARM1,PARM2,REST)'`

This means that if you enter:

```
FIXIT GOOD BAD AND UGLY
```

variable PARM1 is assigned the value "GOOD", PARM2 is assigned the value "BAD", and REST is assigned the value "AND UGLY".

If the parameters passed were GOOD BAD, variable REST would be null. Also, if the parameters are enclosed in quotation marks, such as:

```
FIXIT 'GOOD BAD' 'AND UGLY'
```

PARM1 would be set to "GOOD BAD", PARM2 would be set to "AND UGLY", and REST would be null.

For another example, see the ISRTRYIT macro (Figure 38 on page 115). If the MACRO statement contains two variables (`ISREDIT MACRO (`*`command`*`,`*`parm`*`)`), entering:

```
ISRTRYIT RESET
```

sets the variables *command* to "RESET" and *parm* to null. Conversely, this command:

```
ISRTRYIT FIND A
```

sets *command* to "FIND" and *parm* to "A". To find out what was actually typed on the command line, a macro may examine the variable ZEDITCMD, which is in the shared variable pool. ZEDITCMD is a character variable, the length if which depends on the length of the command entered. Therefore, you should either VDEFINE ZEDITCMD to be sufficiently large to hold the expected command, or use the VCOPY service to get the length.

Using edit macros in batch

You can run edit macros in batch by submitting JCL which allocates all of the necessary ISPF libraries (refer to *z/OS ISPF Dialog Developer's Guide and Reference*), and runs a command which calls the EDIT service with an initial macro. This initial macro can do anything that can be done by an initial macro in an interactive session. However, in batch, the macro should end with an ISREDIT END or ISREDIT CANCEL statement. These statements ensure that no attempt is made to display the edit screen in batch.

A simple initial macro to change strings in batch might look like this:

```
ISREDIT MACRO
ISREDIT CHANGE JANUARY FEBRUARY ALL
ISREDIT END
```

Edit macro messages

You can display messages from an edit macro the same way you do from an ISPF dialog.

- Use SETMSG, which causes the message to appear on whatever panel is displayed next.
- Use DISPLAY with the MSG keyword. This is useful if the macro displays panels of its own.

ISPF provides three generic messages for use in dialogs where you want to generate the message text or when you do not want a separate message library.

```
ISRZ000  '&ZEDSMSG'  .ALARM = NO   .HELP = ISR2MACR
'&ZEDLMSG'
```

```
ISRZ001  '&ZEDSMSG'  .ALARM = YES  .HELP = ISR2MACR
'&ZEDLMSG'
```

```
ISRZ002  '&ZERRSM'  .ALARM = &ZERRALRM  .HELP = &ZERRHM
'&ZERRLM'
```

For example, if you want your macro to sound an alarm and to issue the short message `INVALID PARAMETER` and the long message `PARAMETER MUST BE 4 DIGITS`, use these statements:

CLIST statements

```
SET &ZEDSMSG = &STR(INVALID PARAMETER)
SET &ZEDLMSG = &STR(PARAMETER MUST BE 4 DIGITS)
ISPEXEC SETMSG MSG(ISRZ001)
```

REXX statements

```
ADDRESS ISPEXEC
zedsmsg = 'Invalid Parameter'
zedlmsg = 'Parameter must be 4 digits'
'SETMSG MSG(ISRZ001)'
```

Note: ZEDLMSG only displays when you enter the HELP command.

Macro levels

Each macro operates on a separate and unique level. A person at the keyboard always operates at level 0. If that person starts a macro, it operates at level 1; the macro started by a level-1 macro operates at level 2, and so on. The level is the degree of macro nesting. Edit macros are primary commands; thus, nested macros are started by prefixing them with ISREDIT.

A macro can determine its own level with this assignment statement:

```
ISREDIT (varname) = MACRO_LEVEL
```

The current level number is stored in the specified variable. ISPF supports up to 255 levels of macro nesting.

Labels in edit macros

A label is an alphabetic character string used to name lines. It is especially useful for keeping track of a line whose relative line number may change because labels remain set on a line even when relative line numbers change. The special labels shown are automatically assigned by the editor. A label must begin with a period (.) and be followed by no more than 8 alphabetic characters, the first of which cannot be Z. No special characters or numeric characters are allowed.

The special labels that are automatically assigned by the editor all begin with the letter Z. Labels beginning with Z are reserved for editor use only.

The editor-assigned labels are:

.ZCSR
The data line on which the cursor is currently positioned.

.ZFIRST
The first data line (same as relative line number 1). Can be abbreviated .ZF.

.ZLAST
The last data line. Can be abbreviated .ZL.

.ZFRANGE
The first line in a range specified by you.

.ZLRANGE
The last line in a range specified by you.

.ZDEST
The destination line specified by you.

Note: Unlike other labels, .ZCSR, .ZFIRST, and .ZLAST do not stay with the same line. Label .ZCSR stays with the cursor, and labels .ZFIRST and .ZLAST point to the current first and last lines, respectively.

Using labels

In a macro, you can assign a label to a line by using the LABEL assignment statement. For example:

CLIST Statements

```
SET &LNUM = 10
ISREDIT LABEL &LNUM = .HERE
```

REXX Statements

```
ADDRESS ISPEXEC
lnum = 10
'ISREDIT LABEL' lnum '= .HERE'
```

This assigns the label .HERE to the line whose relative line number is contained in variable LNUM (line 10 here). The .HERE label allows the macro to keep track of a line whose relative line number may change. When the macro finishes running, the .HERE label is removed.

Labels can be used as part of a keyphrase instead of a line number. For example:

CLIST Statements

```
ISREDIT LINE .NEXT = (DATAVAR)
ISREDIT LINE_AFTER .XYZ = (DATAVAR)
```

REXX Statements

```
ADDRESS ISPEXEC
'ISREDIT LINE .NEXT = (DATAVAR)'
'ISREDIT LINE_AFTER .XYZ = (DATAVAR)'
```

The first example stores new data into the line that currently has the label .NEXT. The second example creates a new line after the line whose label is .XYZ, and stores data into the new line.

A macro can determine if a label exists. Using the LINENUM assignment statement, you can obtain the current relative line number of a labeled line. If the label does not exist, the return code (&LASTCC for CLIST or RC for REXX) is 8. For example:

CLIST Statements

```
ISREDIT (LNUM2) = LINENUM .ABC
IF &LASTCC = 8 THEN WRITE NO .ABC LABEL
```

REXX Statements

```
ADDRESS ISPEXEC
'ISREDIT (LNUM2) = LINENUM .ABC'
IF RC = 8 THEN SAY 'No .ABC label'
```

This example stores the relative line number of the line with label .ABC into variable LNUM2 and tests to see if that label did exist.

Labels have a variety of uses. For example, because both the FIND and SEEK commands position the cursor at the search string after the macro has been started, you may want to assign the data from the line on which the cursor is positioned to the variable CSRDATA. To do so, use this statement:

CLIST Statements

```
ISREDIT FIND 'IT'
ISREDIT (CSRDATA) = LINE .ZCSR
```

REXX Statements

```
ADDRESS ISPEXEC
'ISREDIT FIND IT'
'ISREDIT (CSRDATA) = LINE .ZCSR'
```

The label .ZCSR names the line in which the cursor is positioned. The .ZCSR label is moved to a new line when one of these commands moves the cursor: FIND, CHANGE, SEEK, EXCLUDE, TSPLIT or CURSOR. The labels .ZFIRST and .ZLAST can also move when data is added or deleted.

If you assign a labeled line a new label that is blank, the previous label becomes unassigned (if both labels are at the same level). For example:

CLIST Statement	REXX Statements
`ISREDIT LABEL .HERE = ' '`	`ADDRESS ISPEXEC` `"ISREDIT LABEL .HERE = ' '"`

removes the label from the line.

If a label in use is assigned to another line, the label is moved from the original line to the new line (if the new assignment is at the same level as the original).

Referring to labels

A nested macro can refer to all labels assigned by higher-level macros and to labels that you assign. When a macro assigns labels, they are associated by default with the assigning macro level. The labels are automatically removed when the macro finishes running. The labels belong to the level at which they are assigned and can have the same name as the labels at other levels without any conflict.

When a macro ends, the labels at the current nesting level are deleted. To set a label for the next higher level, the macro can issue the MACRO_LEVEL assignment statement to obtain the current level and decrease the level by 1.

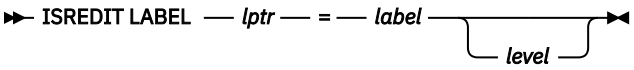
A macro can determine the level of a label with the LABEL assignment statement, as shown in this syntax:

```
ISREDIT (varname1,varname2) = LABEL lptr
```

The label assigned to the referenced line is stored in the first variable and its level is stored in the second variable. If a label is not assigned to the line, a blank is stored in both variables.

Passing labels

You can create a label at any level above its current level by explicitly stating the level:



Here, if the label previously existed at the explicitly specified level, its old definition is lost. A label assigned at a higher level remains after the macro ends and is available until the level at which it was assigned ends or the label is explicitly removed.

If a macro sets a label without indicating a level, or if its value is equal to or greater than the level at which the macro is running, the label is set at the macro level that is currently in control and does not affect any labels set in a higher level.

If a macro queries a label without specifying a level, or uses the label as a line pointer, the search for the label starts at the current macro level and goes up, level by level, until the label defined closest to the current level is found.

If you specify a level parameter that is outside the currently active levels, it is adjusted as follows: a value less than zero is set to zero; a value greater than the current nesting level is set to the current nesting level. This means that a higher-level macro cannot set a label at the level of the macro that it is going to start.

Referring to data lines

You can refer to data lines either by a relative line number or by a symbolic label. Note that special lines (MASK lines, TABS lines, COLS lines, BOUNDS lines, MSG lines, and others) are not considered data lines. You cannot assign labels to them, and they do not have relative line numbers. Also, you cannot directly reference these lines in a macro, even though they are displayed. Excluded lines are regarded as data lines.

Relative line numbers are not affected by sequence numbers in the data, nor are they affected by the current setting of number mode. The first line of data is always treated as line number 1, the next line is line number 2, and so on. The TOP OF DATA line is considered line number 0.

When you insert or delete lines, the lines that follow change relative line numbers. If you insert a new line after line 3, for example, it becomes relative line 4 and what was relative line 4 becomes relative line 5, and so on. Similarly, if line 7 is deleted, the line that was relative line 8 becomes relative line 7, and so on.

Referring to column positions

Column positions in edit macros are not the same as they appear on the panel; they refer only to the editable portions of the data. When number mode is on, sequence numbers are not part of the data, and thus are not editable. For example, if NUMBER COBOL ON mode is in effect, the first six positions of each line contain the sequence number. The first data character is in position 7, which is considered relative column 1. When number mode is off, the line number portion is editable, so here position 1 becomes column 1 and position 7 becomes column 7. These are not the column values displayed on the edit panel. This discrepancy can influence the use of column numbers as parameters from the keyboard. Column numbers must be converted according to number mode. See “Edit boundaries” on page 23 for the conversions.

If your macro must access the sequence numbers as data, include statements that save the current number mode, set number mode off, and then restore the original number mode.

When a macro retrieves the current cursor position, a relative column number of zero is returned if the cursor is outside the data portion of the line. When a macro sets the cursor column to zero, the cursor is placed in the Line Command field on the left side of the designated line.

Defining macros

You can use DEFINE to give macros names that are different from their data set names, make aliases for built-in edit commands, identify macros as program macros, or set a command as disabled. DEFINE commands are usually issued in an initial macro.

For more information, refer to the description of the DEFINE command in Chapter 11, “Edit macro commands and assignment statements,” on page 295.

Defining an alias

To establish an alias or alternate name for a primary command, enter a DEFINE followed by the new name, the ALIAS operand, and then the original command name. For example, this command:

```
DEFINE FILE ALIAS SAVE
```

establishes FILE as an alias for SAVE, allowing you to enter FILE to save the data currently being edited instead of SAVE.

Resetting definitions

To reset the last definition for a command and return the command to its previous status, use the DEFINE command with the RESET operand. For example, having established FILE as an alias for SAVE, you can enter this command to cause FILE to be flagged as an invalid command:

```
DEFINE FILE RESET
```

When defining a command as DISABLED, you cannot reset the disabled function.

Replacing built-in commands

You also use DEFINE to replace an existing edit command with a macro. This links the command name to an edit macro. For example:

CLIST Statement	REXX Statements
`ISREDIT DEFINE FIND ALIAS MYFIND`	`ADDRESS ISPEXEC` `'ISREDIT DEFINE FIND ALIAS MYFIND'`

To use the built-in edit command, precede the command with BUILTIN. For example, to process the built-in FIND command, include this statement:

	REXX Statements
`ISREDIT BUILTIN FIND …`	`ADDRESS ISPEXEC` `'ISREDIT BUILTIN FIND …'`

The ellipses (...) represent other FIND command operands such as the search string.

Implicit definitions

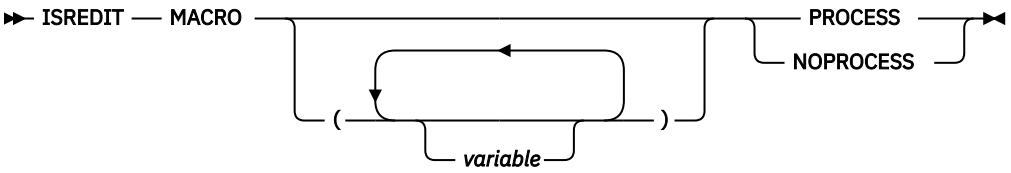
When you or your macro issue a command unknown to the editor, PDF searches for a CLIST or REXX exec with that name. If the editor finds the command, it is implicitly defines it as an edit macro.

Program macros can be implicitly defined by preceding the name of the macro with an exclamation point (!). Remember that the name must be 7 characters or less, excluding the exclamation point. Program macros are similar to ISPF dialogs in that they must be made available as load modules in either the ISPLLIB, STEPLIB, or LINKLST library. See “Program macros” on page 89 for more information.

Using the PROCESS command and operand

The PROCESS command provides a way to alter the usual sequence of events in an edit macro. It is related to the PROCESS operand on the MACRO command. PROCESS is the default for the MACRO command. PROCESS specifies that display data and line commands be processed before another statement is processed. If you specify NOPROCESS, the editor defers processing the panel data and line commands until it finds an ISREDIT PROCESS command later in the macro, or until the macro ends. You can use PROCESS to create a "before-and-after" effect. If you specify NOPROCESS at the beginning of a macro, edited data appears without the changes made from the keyboard—creating a "before" effect. Once you specify PROCESS, changes that were made from the keyboard appear—creating an "after" effect.

The syntax of the ISREDIT MACRO statement is:



Using the PROCESS command with edit line macros

The PROCESS command is used within edit line macros to set the .ZFRANGE, .ZLRANGE, and .ZDEST labels for use by the macro. For edit user line commands, you must specify NOPROCESS on the MACRO statement and include a PROCESS statement within the macro. For macros run by your own line commands, the PROCESS statement does not delay or control the execution of other line commands, as the editor executes all the line commands in sequential order and executes any preceding line commands prior to invoking the user line macro.

Specifying NOPROCESS in the macro statement

NOPROCESS is useful if you want to process statements before the display data or line commands are processed. It enables you to perform initial verification of parameters or capture lines before they are changed from the panel.

It is also useful if you want to include an ISREDIT PROCESS command to specify whether the macro expects, and handles, line commands that identify either a range of lines, a destination line, or both. This linking is the method by which the editor allows a macro command to interact with line commands in the same way that the built-in MOVE and REPLACE commands do. With the ISREDIT PROCESS command, the editor can process line commands that you have entered, performing significant error and consistency checking.

Specifying a destination

If you include this process statement in an edit macro:

CLIST Statement	REXX Statements
`ISREDIT PROCESS DEST`	`ADDRESS ISPEXEC` `'ISREDIT PROCESS DEST'`

the macro expects you to specify a destination line. A destination line is always specified using either A (after) or B (before). The editor sets the dialog variable .ZDEST to the line preceding the destination. However, if neither A nor B is specified, .ZDEST is set to the last data line. In this situation, a return code shows that no destination was specified.

Specifying a range

If you use this syntax for a PROCESS macro command in an edit macro:

```
ISREDIT PROCESS RANGE operand
```

the macro expects to receive a specified range of lines to process. The operand following the RANGE operand identifies either one or two commands that are to be accepted. For example, the command PROCESS RANGE Q Z allows the line commands Q or Z (but not both) to be processed with this macro. The line commands could take any of these forms:

- Q or Z, to specify a single line.
- QQ or ZZ, to specify a block of lines. This form is obtained by doubling the last letter of the single-line command.
- Q*n* or Z*n* where *n* is a number that specifies a series of lines.

After the PROCESS command is completed, the dialog variable .ZFRANGE is automatically set to the first line of the specified range. The dialog variable .ZLRANGE is set to the last line of the specified range. These labels can refer to the same line. If no range is entered, the range defaults to the entire data set. In this situation, a return code shows that no range was specified.

Two line command names can be specified for PROCESS In this situation, use the RANGE_CMD assignment statement to return the value of the command entered. For example, if you issue this PROCESS command:

CLIST Statement	REXX Statements
`ISREDIT PROCESS RANGE Z $`	`ADDRESS ISPEXEC` `'ISREDIT PROCESS RANGE Z $'`

The RANGE_CMD assignment statement returns either a Z or a $.

The names of line commands that define the range can be 1 to 6 characters, but if the name is 6 characters long, it cannot be used as a block format command by doubling the last character. The name can contain any alphabetic or special character except blank, hyphen (-), apostrophe ('), or period (.). It cannot contain any numeric characters.

Example

In the example that follows, the NOPROCESS operand on the MACRO command defers processing of the panel data until the line with the cursor is assigned to a variable. After the PROCESS command, the line contains any changes that you made.

CLIST Statements

```
ISREDIT MACRO NOPROCESS
ISREDIT (BEFORE) = LINE .ZCSR
ISREDIT PROCESS
ISREDIT (AFTER) = LINE .ZCSR
IF &STR(&BEFORE) = &STR(&AFTER) THEN -
     ...
ELSE -
     ...
```

REXX Statements

```
ADDRESS ISPEXEC
'ISREDIT MACRO NOPROCESS'
'ISREDIT (BEFORE) = LINE .ZCSR'
'ISREDIT PROCESS'
'ISREDIT (AFTER) = LINE .ZCSR'
IF BEFORE = AFTER THEN
     ...
ELSE
     ...
```

See "PROCESS—Process Line Commands" on page 383.

Recovery macros

After a system failure, you might want to restore the command definitions and aliases that you were using when the system failed, but you do not want to destroy the profile changes you made during the edit session before the failure.

To help to recover after a system failure, you can provide a recovery macro which can restore command definitions and aliases while not destroying profile changes made before the failure. The recovery macro, like an initial macro, runs after the data has been read but before it is displayed. However, the macro is run whenever the recovery data set is being edited.

You can specify a recovery macro:

- By entering the RMACRO primary command:

  ```
  RMACRO name
  ```

- In your initial macro by using the RMACRO assignment statement:

  ```
  ISREDIT RMACRO = name
  ```

  where *name* sets the name of the macro for the edit session. The name operand is used to specify the name of the macro to be run after a data set has been recovered.

Note: Recovery macros are only in effect for the duration of a particular Edit session. They must be specified again each time a new member or data set is edited.

Return codes from user-written edit macros

A macro can issue the return codes shown here. These return codes affect the command line and cursor position on the next display of edit data:

0
Shows normal completion of the macro. The cursor position is left as set by the macro. The command line is blanked.

1
Shows normal completion of the macro. The cursor is placed on the command line and the line is blanked. Use this return code to make it easy to enter another macro or edit command on the command line.

4 and 8
Treated by the ISPF editor as return code 0. No special processing is done.

12 and higher
Error return codes. The cursor is placed on the command line and the macro command remains. When used with these return codes, the dialog manager SETMSG service prompts you for an incorrect or omitted parameter.

Any invocation of a disabled macro command issues a return code of 12. See the DEFINE command for more information on disabled commands.

20 and higher
Indicate a severe error. The meanings of the severe return codes are:

20
Command syntax error or Dialog service routine error.

24
Macro nesting limit of 255 exceeded (possible endless loop; see the BUILTIN macro command).

28
Command found either preceding the ISREDIT MACRO command, or following the ISREDIT END or ISREDIT CANCEL command.

Each command description in Chapter 11, "Edit macro commands and assignment statements," on page 295 includes a list of return codes that are possible for the command. Because &LASTCC (CLIST) or RC (REXX) is set for every statement, you must either test it in the statement immediately following the command that sets it, or you must save its value in another variable. Use a command such as:

```
SET &RETCODE = &LASTCC
```

The variable (&RETCODE or RETCODE) can then be tested anywhere in the macro until it is changed.

Return codes from PDF edit macro commands

Every CLIST edit macro command sets variable &LASTCC with a return code. REXX edit macros set variable RC. The return codes range from 0 to 20.

0
Shows normal completion of the command.

2, 4, and 8
Information return codes. They show a special condition that is not necessarily an error. These return codes can be tested or ignored, depending on the requirements of the macro.

For some cases of RC=8, the ISPF system variables ZERRSM (short error message text) and ZERRLM (long error message text) are set. For more information on ZERRSM and ZERRLM, see *z/OS ISPF Dialog Developer's Guide and Reference*.

12 and higher
Error return codes. Normally an error return code causes the macro to end abnormally and an error panel to appear. The error panel shows the kind of error and lists the statement that caused the error condition.

The ISPF system variables ZERRSM (short error message text) and ZERRLM (long error message text) are set for error return codes. For more information on ZERRSM and ZERRLM, see *z/OS ISPF Dialog Developer's Guide and Reference*.

Often, the only two possible return codes are 0 and 20. The CAPS command is an example of such a command. Any valid form of CAPS issues a return code of 0.

The dialog variables ZEDMSGNO (message identifier), ZEDISMSG (short message text) and ZEDILMSG (first 240 bytes of the long message text) are available to be tested for or displayed within edit macros. These variables contain values relating to any message that would have been displayed at the terminal had the user issued the command directly from the command line. They can be useful in situations where the return code does not provide enough detail.

Selecting control for errors

As explained in “Return codes from PDF edit macro commands” on page 110, every edit macro statement causes variable &LASTCC (CLIST) or RC (REXX) to be set to a return code. Return codes of 12 or higher are considered errors (except for the PROCESS edit macro command return code of 12), and the default is to end macros that issue those return codes.

Sometimes you need to handle errors at the time that they occur. The error is expected and the edit macro logic can handle the problem. If you want to handle all errors that might occur in your macro, you can include this statement:

```
ISPEXEC CONTROL ERRORS RETURN
```

If errors occur, control returns to the macro. On the other hand, to return error handling to the default mode, include this statement:

```
ISPEXEC CONTROL ERRORS CANCEL
```

If an error occurs, the macro ends.

If you want to do both, you can include any number of ISPEXEC CONTROL statements in your macro to turn error handling on and off.

Chapter 7. Testing edit macros

This chapter documents general-use programming interfaces and associated guidance information. It also tells you how to include statements in your edit macros to capture and handle error conditions.

Using the information in the preceding chapters , you should be able to write and run an edit macro that uses CLIST or REXX logic and processes simple edit commands. However, even an experienced edit macro writer occasionally includes a bug that causes a macro to end abnormally (ABEND), or writes a macro that does not work as expected. When this occurs, you must debug your macro, just as you would debug any other kind of program you write.

Handling errors

There are two kinds of errors that you may encounter when you debug macros: edit command errors and dialog service errors. Both kinds of errors are controlled by the ISPEXEC CONTROL ERRORS RETURN command. For more information about the CONTROL service, refer to *z/OS ISPF Services Guide*.

Edit command errors

The editor detects edit command errors and displays either an edit macro error panel with an error message, or a return code. If an edit command error occurs, the macro ends abnormally with these results:

- When you are using the ISPF editor with ISPF test mode off, you return to the edit session.
- If ISPF test mode is on, the PDF component is also in test mode. You can override the abnormal end and attempt to continue by typing YES on the PDF edit macro error panel and pressing Enter. If ISPEXEC CONTROL ERRORS RETURN has been processed, the error panel does not appear, and the macro automatically continues.

Dialog service errors

ISPF detects dialog service errors and displays a message identifying the error with the statement which caused the error. If a dialog service error occurs, the edit session ends abnormally with these results:

- When you are using the PDF component with ISPF test mode off, the ISPF Primary Option Menu is displayed.
- If you are using the PDF component with ISPF test mode on, you can override the abnormal end and attempt to continue by typing YES on the ISPF dialog error panel and pressing Enter. In either case, if ISPEXEC CONTROL ERRORS RETURN has been processed, no panel appears and the editor sends a return code instead of ending the dialog.

Note: If you enter ISPF with TEST as an operand, or use Dialog Test (option 7), ISPF remains in test mode until you end the ISPF session.

Using CLIST WRITE statements and REXX SAY statements

The CLIST WRITE statement and the REXX SAY statement can be valuable tools in tracking down edit macro problems. A WRITE statement or a SAY statement is simply a line of text inserted into your macro that creates a message on your screen while the macro is running. With these statements, you can identify the position of the statement within the macro, and display the value of variables.

For example, if you are having trouble debugging the CLIST ISRTDATA macro from Figure 27 on page 81, adding some WRITE statements may help locate the problem.

```
/*********************************************************************/
/*                                                                   */
/* 5647-A01 (C) COPYRIGHT IBM CORP 1995, 2003                        */
/*                                                                   */
/* ISRTDWRI - generates test data                                    */
/*                                                                   */
/*********************************************************************/
ISREDIT MACRO
  SET &COUNT = 1                   /* Initialize loop counter    */
  DO WHILE &COUNT <= 9             /* Loop up to 9 times         */
    ISREDIT FIND 'TEST-#'          /* Search for 'TEST-#'        */
    SET &RETCODE = &LASTCC         /* Save the FIND return code  */
    WRITE RESULT OF FIND, RC = &RETCODE
    IF &RETCODE = 0 THEN           /* If string was found,       */ -
      DO                           /*                            */
        ISREDIT CHANGE '#' '&COUNT' /*   Change # to a digit and  */
        SET &COUNT = &COUNT + 1    /*   increment loop counter   */
        WRITE COUNT IS NOW UP TO &COUNT
      END                          /*                            */
    ELSE                           /* If string is not found,    */ -
      SET &COUNT = 10              /*   Set counter to exit loop */
  END
EXIT CODE (0)
```

Figure 36. ISRTDATA macro with CLIST WRITE statements

Remember that the macro ISRTDATA creates test data with variations of the same line by putting ascending numbers 1 through 9 in the data. When WRITE statements are included in the data, a step-by-step breakdown of the procedure appears on your screen.

If there are no errors in the ISRTDATA macro, the return codes and count appear on your screen in TSO line mode. Asterisks at the bottom of the screen prompt you to press Enter and return to ISPF full-screen mode (Figure 37 on page 114).

```
RESULT OF FIND, RC = 0
COUNT IS NOW UP TO 2
RESULT OF FIND, RC = 0
COUNT IS NOW UP TO 3
RESULT OF FIND, RC = 0
COUNT IS NOW UP TO 4
RESULT OF FIND, RC = 0
COUNT IS NOW UP TO 5
RESULT OF FIND, RC = 0
COUNT IS NOW UP TO 6
RESULT OF FIND, RC = 0
COUNT IS NOW UP TO 7
RESULT OF FIND, RC = 0
COUNT IS NOW UP TO 8
RESULT OF FIND, RC = 0
COUNT IS NOW UP TO 9
RESULT OF FIND, RC = 0
COUNT IS NOW UP TO 10
 ***_
```

Figure 37. Results of ISRTDATA macro with CLIST WRITE statements

Using CLIST CONTROL and REXX TRACE statements

You can display a statement from a macro as it is being interpreted and run. Use either of these:

- A CLIST CONTROL statement with the LIST, SYMLIST, or CONLIST operand
- A REXX TRACE statement with the A, I, L, O, R, or S operand

These statements produce messages on your display screen similar to the WRITE and SAY statements discussed in the previous section. However, several differences should be noted:

- For the CLIST CONTROL statement:
  - LIST displays commands and subcommands (including ISREDIT statements) after substitution but before processing. This allows you to see an ISREDIT statement in the form that the editor sees the statement.
  - CONLIST displays a CLIST statement (for example, IF, DO, SET) after substitution but before processing. You might be able to tell why an IF statement did not work properly by using CONLIST.
  - SYMLIST displays both CLIST and command lines before symbolic substitution, allowing you to see the lines as written.

  Use the NOLIST, NOSYMLIST, and NOCONLIST operands to prevent the display of statements. See *z/OS TSO/E CLISTs* for more details.
- For the REXX TRACE statement:
  - The A operand traces all clauses displaying the results of each clause.
  - The I operand traces the intermediate results, displaying both the statement and the results.
  - The L operand traces labels in your edit macro.
  - The O operand stops, or turns off, the trace.
  - The R operand, which is used most often, traces all clauses and expressions.
  - The S operand scans each statement, displaying it without processing it.

See *z/OS TSO/E REXX Reference* and *z/OS TSO/E REXX User's Guide* for more details.

Experimenting with macro commands

Use the ISRTRYIT macro (Figure 38 on page 115) to experiment with edit macros. ISRTRYIT is handy when you want to see how a command or assignment statement works but do not actually want to write an entire macro. ISRTRYIT processes the command and issues return codes that show whether it succeeded. To start the macro, type ISRTRYIT on the command line, followed by a command, and press Enter. If you enter ISRTRYIT with the RESET operand, the variable &COMMAND is set to RESET; if you enter it as ISRTRYIT FIND A, the variable &COMMAND is set to FIND A.

```
/*********************************************************************/
/*                                                                   */
/* 5647-A01 (C) COPYRIGHT IBM CORP 1995, 2003                        */
/*                                                                   */
/* ISRTRYIT - a simple macro for trying out edit macro statements.   */
/*                                                                   */
/*********************************************************************/
ISREDIT MACRO (COMMAND)
  SET &RETCODE = 0                    /* Initialize return code  */
  IF &STR() = &STR(&COMMAND) THEN     /* If no command specified */ -
    WRITE MISSING COMMAND PARAMETER   /*   indicate problem      */
  ELSE DO                             /* Else parameter exists;  */
    ISREDIT &COMMAND                  /* Invoke edit command,    */
    SET &RETCODE = &LASTCC            /*   save the return code  */
    WRITE &COMMAND RETURN CODE IS &RETCODE  /*   and display it   */
  END                                 /*   and the command name  */
EXIT CODE(&RETCODE)
```

Figure 38. ISRTRYIT macro

The ISRTRYIT macro tests both the SEEK and AUTONUM commands (Figure 39 on page 116). When you run the macro, it displays the return codes from the commands on your screen (Figure 40 on page 116).

```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
_____________________________________________________________________
EDIT       SBURNF.PRIVATE.DATA(TESTDATA) - 01.00           Columns 00001 00072
Command ===> isrtryit seek "test"; isrtryit autonum on       Scroll ===> CSR
****** ***************************** Top of Data ******************************
000100 TEST-#
000200 TEST-#
000300 TEST-#
000400 TEST-#
000500 TEST-#
000600 TEST-#
000700 TEST-#
000800 TEST-#
000900 TEST-#
001000 TEST-#
001100 TEST-#
001200 TEST-#
001300 TEST-#
****** **************************** Bottom of Data ****************************


 F1=Help      F2=Split     F3=Exit      F5=Rfind     F6=Rchange   F7=Up
 F8=Down      F9=Swap     F10=Left     F11=Right    F12=Cancel
```

Figure 39. ISRTRYIT macro - before running

```
ISREDIT SEEK "TEST"
RETURN CODE IS 0
ISREDIT AUTONUM ON
RETURN CODE IS 0
 ***_
```

Figure 40. ISRTRYIT macro - after running

Debugging edit macros with ISREMSPY

When you run an edit macro, the editor screen is not displayed until the macro completes. To view the status of the data being edited during execution of the edit macro, invoke the program ISREMSPY from within the running macro.

ISREMSPY displays a simulated editor panel in which the data is presented as it exists at the time ISREMSPY is started. You can also see the cursor location and the last edit macro command executed. In most cases, the line that has the cursor on it is indicated by an arrow in the line command field.

Within an ISREMSPY display you can issue the commands RESET and FIND. RESET restores the display to the current editor state, including scroll and cursor location. FIND locates a string within the data being display.

FIND does not support all the operands of the FIND command of the real editor; it only supports the search string as an operand. The string may be in quotes, and embedded quotes should not be doubled. Pressing the RFIND key will repeat the last search. Only the first 256 bytes of each line are searched by the FIND command.

Because ISREMSPY is a *simulated* edit session, it may not display precisely as the editor would. For example, the numbers in the line command field are always incremented by one, and may not accurately reflect the numbers displayed in the real edit session. Similarly, there are some cases such as TENTER and INSERT, where the cursor location may not be correct.

ISREMSPY can be invoked in several ways:

- You can invoke it as a TSO command directly from within an edit macro.

  CLIST example:

  ```
  ISREMSPY
  ```

REXX example:

```
Address TSO 'ISREMSPY'
```

- You can define a breakpoint for ISREDIT in dialog test (option 7.8) and then run the macro under dialog test (option 7.1). When the breakpoint is triggered, you can type TSO ISREMSPY to view the current state of the edit data. This technique can be used to look at edit data during execution of a macro without having to modify the edit macro source and is particularly useful for debugging program macros (macros not written in CLIST or REXX).
- You can define ISREMSPY as a program macro using the editor DEFINE command and then use ISREMSPY as an editor command.

Chapter 8. Sample edit macros

This chapter documents general-use programming interfaces and associated guidance information.

ISRBOX macro

The ISRBOX macro draws a box with its upper left corner at the cursor position. This macro comes in handy when you want to make a note to yourself or others reading the data. You can start the ISRBOX macro in one of two ways:

- Type ISRBOX on the command line as an edit primary command and press Enter.
- Type KEYS on the command line, press Enter, set a function key to the ISRBOX macro, and enter the END command.

If you have defined a function key for ISRBOX, position the cursor on a data line where you want the box drawn. Press the function key that you have defined to start the ISRBOX macro. After the box is drawn, the cursor is positioned inside, ready for you to type enough text to fill the box.

If any of the macro commands fail, a warning message appears.

```
/*********************************************************************/
/*                                                                   */
/* 5647-A01 (C) COPYRIGHT IBM CORP 1995, 2003                        */
/*                                                                   */
/* ISRBOX - Draw a box with its upper left corner at the             */
/*          cursor position                                          */
/*                                                                   */
/*********************************************************************/
ISREDIT MACRO
ISREDIT (ROW,COL) = CURSOR             /* Get cursor position     */

ISPEXEC CONTROL ERRORS RETURN          /* No macro error panel    */
                                       /* Draw box over existing  */
                                       /*    lines                */

ISREDIT LINE &ROW          = LINE + < &COL '+--------------------+'>
ISREDIT LINE &EVAL(&ROW+1) = LINE + < &COL '|                    |'>
ISREDIT LINE &EVAL(&ROW+2) = LINE + < &COL '|                    |'>
ISREDIT LINE &EVAL(&ROW+3) = LINE + < &COL '|                    |'>
ISREDIT LINE &EVAL(&ROW+4) = LINE + < &COL '|                    |'>
ISREDIT LINE &EVAL(&ROW+5) = LINE + < &COL '+--------------------+'>

IF &MAXCC > 0 THEN DO                  /* If error occurred while */
   SET ZEDSMSG = &STR(INCOMPLETE BOX)  /*    overlaying lines     */
   SET ZEDLMSG = &STR(NOT ENOUGH LINES/COLUMNS TO DRAW COMPLETE BOX)
   ISPEXEC SETMSG MSG(ISRZ001)         /* Issue error message     */
END

SET &COL = &COL + 2                    /* Position cursor within  */
SET &ROW = &ROW + 1                    /*    the box              */
ISREDIT CURSOR = (ROW,COL)
EXIT CODE(0)
```

Figure 41. ISRBOX macro

This list explains the logical sections of the ISRBOX macro:

1. The variables &ROW and &COL are set to the cursor position.

   ```
   ISREDIT (ROW,COL) = CURSOR
   ```

2. The dialog service allows the macro to handle severe errors, allowing a message to be displayed when the cursor is placed too close to the end of the data. The LINE assignment statement fails if the row it is setting does not exist.

```
ISREDIT CONTROL ERRORS RETURN
```

3. The LINE assignment statements overlay existing data on a line with the characters which form a box. LINE uses a merge format to include the existing line data and then a template to put the overlaying data at the cursor column position. The CLIST &EVAL function increments the relative line numbers before the statement is passed to the editor.

```
ISREDIT LINE &ROW          = LINE + < &COL '+----------------+'>
ISREDIT LINE &EVAL(&ROW+1) = LINE + < &COL '|                |'>
ISREDIT LINE &EVAL(&ROW+2) = LINE + < &COL '|                |'>
ISREDIT LINE &EVAL(&ROW+3) = LINE + < &COL '|                |'>
ISREDIT LINE &EVAL(&ROW+4) = LINE + < &COL '|                |'>
ISREDIT LINE &EVAL(&ROW+5) = LINE + < &COL '+----------------+'>
```

4. The CLIST IF statement checks the &MAXCC variable, and if it is nonzero, calls the dialog service SETMSG to display a message. &MAXCC is a variable updated by the CLIST processor to contain the highest condition code.

```
IF &MAXCC > 0 THEN
```

5. The message used in SETMSG is one of two messages (ISRZ000 and ISRZ001) reserved for macro use. Each message uses two variables:

   - &ZEDSMSG to set the text for the short message (up to 24 characters) that is displayed when the macro ends.
   - &ZEDLMSG to set the text for the long message that appears when the HELP command is entered.

   Message ISRZ001 sounds the alarm to indicate an error; message ISRZ000 does not sound the alarm.

```
  DO
    SET ZEDSMSG = &STR(INCOMPLETE BOX)
    SET ZEDLMSG = &STR(NOT ENOUGH LINES/COLUMNS +
    TO DRAW COMPLETE BOX)
    ISPEXEC SETMSG MSG(ISRZ001)
  END
```

6. These statements position the cursor within the box to simplify entering text when the panel is redisplayed.

```
SET &COL = &COL + 2
SET &ROW = &ROW + 1
ISREDIT CURSOR = (ROW,COL)
```

This example shows the cursor placed on line `000009` next to the number 9 before starting the macro.

```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
------------------------------------------------------------------------
EDIT       SBURNF.PRIVATE.DATA(TESTDATA) - 01.00           Columns 00001 00072
Command ===> isrbox                                         Scroll ===> CSR
****** ***************************** Top of Data ******************************
000100 TEST-1
000200 TEST-2
000300 TEST-3
000400 TEST-4
000500 TEST-5
000600 TEST-6
000700 TEST-7
000800 TEST-8
000900 TEST-9_
001000 TEST-#
001100 TEST-#
001200 TEST-#
001300 TEST-#
001400 TEST-#
001500 TEST-#
****** **************************** Bottom of Data ****************************

 F1=Help      F2=Split     F3=Exit      F5=Rfind     F6=Rchange   F7=Up
 F8=Down      F9=Swap     F10=Left     F11=Right    F12=Cancel
```

Figure 42. ISRBOX macro - before running

When you press Enter, a box appears beside the cursor.

```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
------------------------------------------------------------------------
EDIT       SBURNF.PRIVATE.DATA(TESTDATA) - 01.00           Columns 00001 00072
Command ===>                                                Scroll ===> CSR
****** ***************************** Top of Data ******************************
000100 TEST-1
000200 TEST-2
000300 TEST-3
000400 TEST-4
000500 TEST-5
000600 TEST-6
000700 TEST-7
000800 TEST-8
000900 TEST-9+--------------------+
001000 TEST-#| _                  |
001100 TEST-#|                    |
001200 TEST-#|                    |
001300 TEST-#|                    |
001400 TEST-#+--------------------+
001500 TEST-#
****** **************************** Bottom of Data ****************************

 F1=Help      F2=Split     F3=Exit      F5=Rfind     F6=Rchange   F7=Up
 F8=Down      F9=Swap     F10=Left     F11=Right    F12=Cancel
```

Figure 43. ISRBOX macro - after running

ISRIMBED macro

The ISRIMBED macro (Figure 44 on page 122) builds a list of imbed (.im) statements found in the member that is entered as an operand. The list is created at the end of the member currently being edited. The imbed statements are indented under a MEMBER identifier line.

You can start this macro by editing a member, typing ISRIMBED and the name of the member that contains the imbed statements as the operand, and pressing Enter.

```
/********************************************************************/
/*                                                                  */
/* 5647-A01 (C) COPYRIGHT IBM CORP 1995, 2003                       */
/*                                                                  */
/* ISRIMBED - creates a list of imbed statements.                   */
/*                                                                  */
/********************************************************************/
ISREDIT MACRO (MEMBER)                  /* Member name passed  */
                                        /*   as input          */
ISREDIT LINE_AFTER .ZL='MEMBER &MEMBER'  /* Add member ID line  */
ISREDIT (LINENBR) = LINENUM .ZL         /* Get line number     */

ISREDIT COPY AFTER .ZL &MEMBER          /* Copy member at end  */
ISREDIT (NEWLL) = LINENUM .ZL           /* Get new last line#  */

IF &LINENBR = &NEWLL THEN               /* If no data was      */ -
   EXIT CODE(8)                         /*   copied, then exit */
ELSE DO
   ISREDIT LABEL &EVAL(&LINENBR + 1) = .FIRST  /* Label first   */
                                        /*   line copied       */
   ISREDIT RESET EXCLUDED               /* Make sure there are */
                                        /*   no previously     */
                                        /*   excluded lines    */

   ISREDIT EXCLUDE ALL .FIRST .ZL       /* Exclude newly       */
                                        /*   copied lines      */
   ISREDIT FIND ALL .IM 1 .FIRST .ZL    /* Show lines          */
   SET FINDRC = &LASTCC                 /*   containing ".im"  */
                                        /*   in column 1       */
   ISREDIT DELETE ALL X .FIRST .ZL      /* Delete any lines    */
                                        /*   still excluded    */
   ISREDIT (NEWLL) = LINENUM .ZL        /* Update last line    */
                                        /* number after delete */
   IF &FINDRC = 0 THEN                  /* If ".im" was found  */ -
     DO WHILE (&LINENBR < &NEWLL)       /*   for all remaining */
                                        /*   copied lines      */
       SET LINENBR = &LINENBR + 1       /*   Shift all .im     */
       ISREDIT SHIFT &LINENBR ) 8       /*   lines right 8     */
     END
END
EXIT CODE(1)                            /* Place cursor on     */
                                        /*   command line      */
```

Figure 44. ISRIMBED macro

This list explains the logical sections of the ISRIMBED macro:

1. Add a line that identifies the member to be searched at the end of ISRIMBED. The .ZL (or .ZLAST) is always associated with the last line in the data.

   ```
   ISREDIT LINE_AFTER .ZL = 'MEMBER &MEMBER'
   ```

2. Retrieve the line number of the identifier line just added into &LINENBR.

   ```
   ISREDIT (LINENBR) = LINENUM .ZL
   ```

3. Now copy, at the end of ISRIMBED, the member name that was passed as an input parameter.

   ```
   ISREDIT COPY AFTER .ZL &MEMBER
   ```

4. &NEWLL is set to the new last line number of ISRIMBED.

   ```
   ISREDIT (NEWLL) = LINENUM .ZL
   ```

5. Check to see if any lines were added by the copy. Exit from the macro if no lines were added.

   ```
   IF &LINENBR = &NEWLL THEN
       EXIT CODE(8)
   ```

6. Set the .FIRST label on the first line copied. This label is available only to this macro; you do not see it.

```
ISREDIT LABEL &EVAL(&LINENBR + 1) = .FIRST
```

7. Excluded lines are deleted later. Therefore, make sure that no lines in the data set are excluded.

```
ISREDIT RESET EXCLUDED
```

8. Exclude all lines that were just copied: all the lines in the range .FIRST to .ZL.

```
ISREDIT EXCLUDE ALL .FIRST .ZL
```

9. The FIND command is used to find all occurrences of .im starting in column 1 of the copied lines. This shows (unexcludes) the lines to keep. If .im was not found on any line, &FINDRC will be 4.

```
ISREDIT FIND ALL .IM 1 .FIRST .ZL
SET FINDRC = &LASTCC
```

10. All the lines still excluded are now deleted.

```
ISREDIT DELETE ALL X .FIRST .ZL
```

11. Obtain the last line number again, because it will have changed if lines were deleted.

```
ISREDIT (NEWLL) = LINENUM .ZL
```

12. If .im lines were found, loop using a column shift to indent them under the member identifier line. Note that &LINENBR is still associated with the identifier line.

```
IF &FINDRC = 0 THEN
   DO WHILE (&LINENBR < &NEWLL)
      SET LINENBR = &LINENBR + 1
      ISREDIT SHIFT &LINENBR ) 8
           END
```

LIST is a member with several imbed statements.

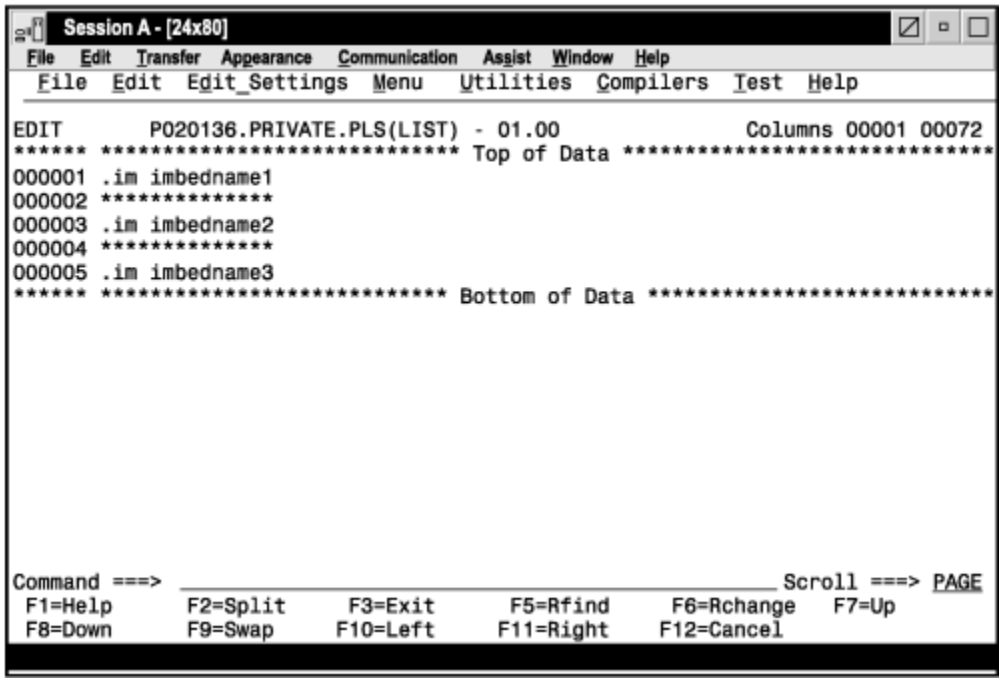


Figure 45. LIST with imbed statements

When you run the ISRIMBED macro by typing `ISRIMBED  LIST` on the command line of ISRTDATA, a list of the imbeds in LIST appears at the end of the data. See Figure 46 on page 124.

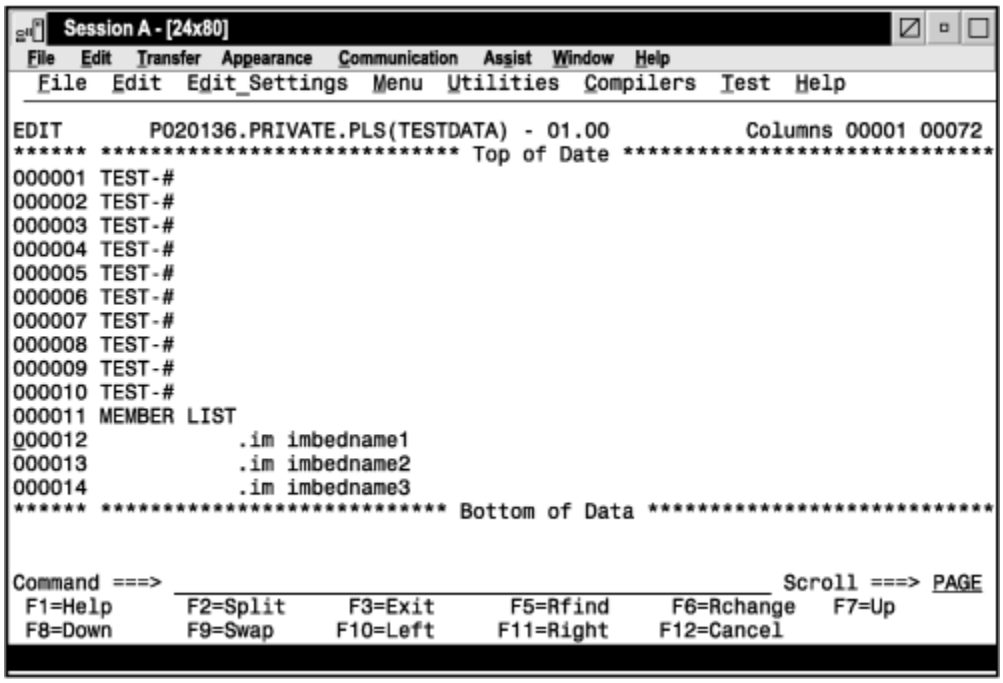


Figure 46. ISRIMBED macro - after running

ISRMBRS macro

The ISRMBRS macro (Figure 47 on page 125) uses PDF library access services to determine each member name in the library being edited.

This macro invokes the edit service for each member in the library, except the member currently being edited, passing a user-specified edit macro on the edit service invocation. The ISRMBRS *macname* command, where *macname* is the name of the macro to be invoked against each member, starts the service.

This macro can aid in making repetitive changes to all members of a data set, or in searching all members for a specific string of data.

```
/*REXX***************************************************************/
/*   ISPF edit macro to process all members of partitioned data set, */
/*   running a second, user-specified, ISPF edit macro against each  */
/*   member.                                                         */
/*                                                                   */
/*   To run:                                                         */
/*    Enter "ISRMBRS macname" on the command line, where macname is  */
/*    the macro you want run against each member.                    */
/*********************************************************************/

'ISREDIT MACRO (NESTMAC)'

/*********************************************************************/
/* Get dataid for data set and issue LMOPEN                          */
/*********************************************************************/
'ISREDIT (DATA1) = DATAID'
'ISREDIT (CURMEM) = MEMBER'
Address ispexec 'LMOPEN DATAID('data1') OPTION(INPUT)'
member = ' '
lmrc = 0

/*********************************************************************/
/* Loop through all members in the PDS, issuing the EDIT service for */
/* each.  The macro specified on the ALLMEMS invocation is passed as */
/* an initial macro on the EDIT service call.                        */
/*********************************************************************/
Do While lmrc = 0
  Address ispexec 'LMMLIST DATAID('data1') OPTION(LIST),
                  MEMBER(MEMBER) STATS(NO)'
  lmrc = rc
  If lmrc = 0 & member ^= curmem Then
    do
      Say 'Processing member' member
      Address ispexec 'EDIT DATAID('data1') MEMBER('member')
                       MACRO('nestmac')'
    end
End

/*********************************************************************/
/* Free the member list and close the dataid for the PDS.            */
/*********************************************************************/
Address ispexec 'LMMLIST DATAID('data1') OPTION(FREE)'
Address ispexec 'LMCLOSE DATAID('data1')'

Exit 0
```

Figure 47. ISRMBRS macro

To start the ISRMBRS macro, edit a new or existing member and enter ISRMBRS *macname*, where *macname* is the name of the macro you wish to invoke against each member of the data set. For example, if the macro is named ISRIMBED, enter: `ISRMBRS  ISRIMBED`

This list explains the logical sections of the ISRMBRS macro:

1. The MACRO command identifies NESTMAC as the variable to contain the name of the macro that is passed on the edit service invocation for each member. If no parameter is passed to ISRMBRS, NESTMAC is blank.

   ```
         ISREDIT MACRO (NESTMAC)
   ```

2. The DATAID assignment statement returns a data ID in the variable DATA1. The data ID identifies the concatenation of data sets currently being edited.

   ```
         ISREDIT (DATA1) = DATAID
   ```

3. The name of the member currently being edited is returned in CURMEM.

   ```
         ISREDIT (MEMBER) = CURMEM
   ```

4. The data set (or sets) identified by the data ID obtained earlier is opened for input to allow the LMMLIST service to be called later. No return code checking is done because it is presumed that if the data set is being edited, it can be successfully processed by LMOPEN.

```
Address ispexec 'LMOPEN DATAID('data1') OPTION(INPUT)'
```

5. The variable to hold the name of the next member to be processed, and the return code from the LMMLIST service are initialized.

```
member = ' '
lmrc = 0
```

6. The exec loops to process all members returned by LMMLIST. Variable LMRC is set to 4 when the end of the member list is reached, stopping the loop.

```
Do While lmrc = 0
```

7. Obtain the next member in the list. If this is the first invocation of LMMLIST, the first member in the list is returned. The member name is returned in variable MEMBER, and variable LMRC is set to the return code from LMMLIST.

```
Address ispexec 'LMMLIST DATAID('data1') OPTION(LIST),
                MEMBER(MEMBER) STATS(NO)'
lmrc = rc
```

8. If LMMLIST returns a 0, indicating a member name was returned, and if the member returned is not the member currently being edited, the member is processed.

```
If lmrc = 0 Then
  do
```

9. The REXX SAY statement is used to write line-I/O messages. As the macro processes each member, the member name appears on the terminal to keep you informed about what is happening. An alternative to the SAY statement would be to display a panel showing the member name after issuing the ISPEXEC CONTROL DISPLAY LOCK service.

```
Say 'Processing member' member
```

10. The EDIT service is invoked on the member returned by LMMLIST. The macro specified on invocation of ISRMBRS is passed as an initial macro on the edit service.

```
Address ispexec 'EDIT DATAID('data1') MEMBER('member')
                MACRO('nestmac')'
```

11. When the LMMLIST service returns a nonzero value, the loop is exited and the cleanup begins. LMMLIST is called to free the member list, and the LMCLOSE service is called to close the data sets associated with the data ID.

```
Address ispexec 'LMMLIST DATAID('data1') OPTION(FREE)'
Address ispexec 'LMCLOSE DATAID('data1')'
```

ISRCHGS macro

The ISRCHGS macro (Figure 48 on page 127) identifies the lines most recently changed by showing only those lines and excluding all others. When no level is passed, the latest level is assumed. A label range can also be passed to ISRCHGS to limit the search. This macro relies on the modification level maintained by the editor for members with numbers and ISPF statistics.

Operands can also be specified. For example, to show lines with level 8 or greater on a line range:

```
Command ===> ISRCHGS 8 .FIRST .LAST
```

```
/*********************************************************************/ 00010003
/*                                                                   */ 00020003
/* 5647-A01 (C) COPYRIGHT IBM CORP 1995, 2003                        */ 00030003
/*                                                                   */ 00040003
/* ISRCHGS - shows the most recent changes to a data set             */ 00050003
/*                                                                   */ 00060003
/*********************************************************************/ 00070003
 ISREDIT MACRO (SEARCH,PARMS)                                          00080003
                                                                       00090003
 ISREDIT (SAVE) = USER_STATE                                           00100003
 ISREDIT (NUMBER, NUMTYPE) = NUMBER                                    00110003
 SET SYSDVAL = &NUMTYPE                                                00120003
 READDVAL STD COBOL DISPLAY                                            00130003
 ISREDIT (STATS) = STATS                                               00140003
 ISREDIT (LEVEL) = LEVEL                                               00150003
                                                                       00160003
 IF &SEARCH = &STR() | &SUBSTR(1:1,&STR(&SEARCH. )) = &STR(.) THEN DO  00170008
   SET PARMS = &STR(&SEARCH &PARMS)                                    00180003
   SET SEARCH = &LEVEL                                                 00190003
 END                                                                   00200003
                                                                       00210003
 IF &STATS = OFF | &NUMBER = OFF | &STD = NOSTD THEN DO                00220003
   SET ZEDSMSG = &STR(INVALID DATA)                                    00230003
   SET ZEDLMSG = &STR(BOTH NUMBER AND STATS MODE MUST BE ON)           00240003
   ISPEXEC SETMSG MSG(ISRZ001)                                         00250003
   EXIT CODE(8)                                                        00260003
 END                                                                   00270003
                                                                       00280003
 IF &DATATYPE(&SEARCH) = CHAR THEN DO                                  00290003
   SET ZEDSMSG = &STR(INVALID ARG)                                     00300003
   SET ZEDLMSG = &STR(SEARCH STRING MUST BE FIRST)                     00310003
   ISPEXEC SETMSG MSG(ISRZ001)                                         00320003
   EXIT CODE(8)                                                        00330003
 END                                                                   00340003
                                                                       00350003
 ISREDIT NUMBER = OFF                                                  00360007
 ISREDIT (RECFM) = RECFM                                               00370003
 IF &RECFM = F THEN DO                                                 00380003
   ISREDIT (LRECL) = LRECL                                             00390003
   SET COL1 = &LRECL - 1                                               00400003
   SET COL2 = &LRECL                                                   00410003
 END                                                                   00420003
 ELSE DO                                                               00430003
   SET COL1 = 7                                                        00440003
   SET COL2 = 8                                                        00450003
 END                                                                   00460003
                                                                       00470003
 ISREDIT EXCLUDE ALL                                                   00480003
                                                                       00490003
 DO WHILE &SEARCH <= &LEVEL                                            00500003
   ISREDIT FIND ALL '&SEARCH' &COL1 &COL2 &PARMS                       00510003
   SET SEARCH = &SEARCH + 1                                            00520005
 END                                                                   00530003
                                                                       00530107
 ISREDIT NUMBER = ON                                                   00531007
 ISREDIT USER_STATE = (SAVE)                                           00550003
 EXIT CODE(1)                                                          00560003
```

Figure 48. ISRCHGS macro

This list explains the logical sections of the ISRCHGS macro:

1. ISRCHGS allows three optional parameters to be passed: a search level and two labels (a label range). If all three are passed, PARMS contains two labels.

   ```
   ISREDIT MACRO (SEARCH,PARMS)
   ```

2. The statements shown here save user information, number mode and type, last find string, cursor location, and other profile and status information. Also, stats mode and the current modification level for parameter checking are retrieved, and the three-part number type is divided into three variables.

   ```
   ISREDIT (SAVE) = USER_STATE
   ISREDIT (NUMBER, NUMTYPE) = NUMBER
   ```

```
SET SYSDVAL = &NUMTYPE
READDVAL STD COBOL DISPLAY
ISREDIT (STATS) = STATS
ISREDIT (LEVEL) = LEVEL
```

3. ISRCHGS requires that the modification level be entered first if it is specified. This check allows the level to default to the current (highest) modification level. A label range can be specified without a level number; PARMS is reset to capture both labels.

```
IF &SEARCH = &STR() | &SUBSTR(1:1,&SEARCH) = &STR(;) THEN -
  DO
    SET PARMS = &STR(&SEARCH &PARMS)
    SET SEARCH = &LEVEL
  END
```

4. Check to see if the member modification level is maintained. If not, issue an error message and exit the macro.

```
IF &STATS = OFF | &NUMBER = OFF | &STD = NOSTD THEN -
  DO
    SET ZEDSMSG = &STR(INVALID DATA)
    SET ZEDLMSG = &STR(BOTH NUMBER AND STATS MODE MUST BE ON)
    ISPEXEC SETMSG MSG(ISRZ001)
    EXIT CODE(8)
  END
```

5. A CLIST DATATYPE function is used to check if the first parameter is valid (a number). If it is not valid, issue an error message and exit from the macro.

```
IF &DATATYPE(&SEARCH) = CHAR THEN -
  DO
    SET ZEDSMSG = &STR(INVALID ARG)
    SET ZEDLMSG = &STR(SEARCH STRING MUST BE FIRST)
    ISPEXEC SETMSG MSG(ISRZ001)
    EXIT CODE(8)
  END
```

6. Now that validity checks have been passed you can set number mode off. This allows you to treat the number field, which contains the level number, as data.

```
ISREDIT NUMBER = OFF
```

7. Set &COL1 and &COL2 to the columns containing the level numbers.

```
ISREDIT (RECFM) = RECFM
IF &RECFM = F THEN -
  DO
    ISREDIT (LRECL) = LRECL
    SET COL1 = &LRECL - 1
    SET COL2 = &LRECL
   END
  ELSE DO
    SET COL1 = 7
    SET COL2 = 8
   END
```

8. Exclude all lines.

```
ISREDIT EXCLUDE ALL
```

9. For each level, find all occurrences of the current modification level. If a label range was specified, it is in the PARMS variable. All lines with matching levels are excluded.

```
DO WHILE &SEARCH <= &LEVEL
  ISREDIT FIND ALL '&SEARCH' &COL1 &COL2 &PARMS
  SEARCH = &SEARCH + 1
END
```

10. Restore user values, especially number mode.

```
ISREDIT USER_STATE = (SAVE)
```

In the example in Figure 49 on page 129 the data contains lines that you have changed.

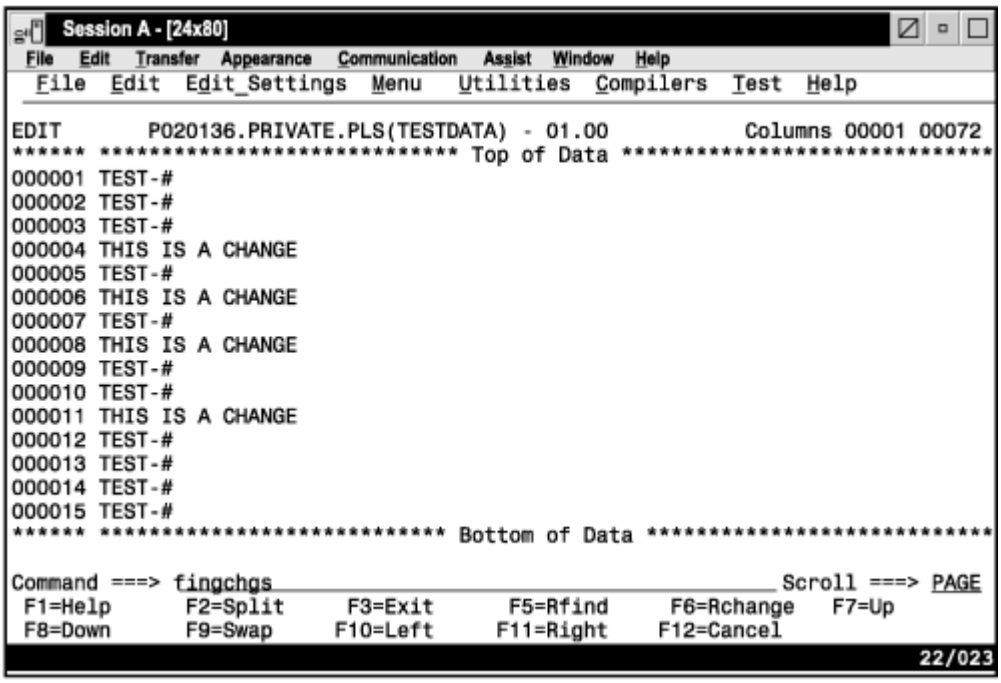


Figure 49. ISRCHGS macro - before running

When you press Enter, the FINDGHGS macro displays the changed lines and excludes the others, as shown in Figure 50 on page 129.

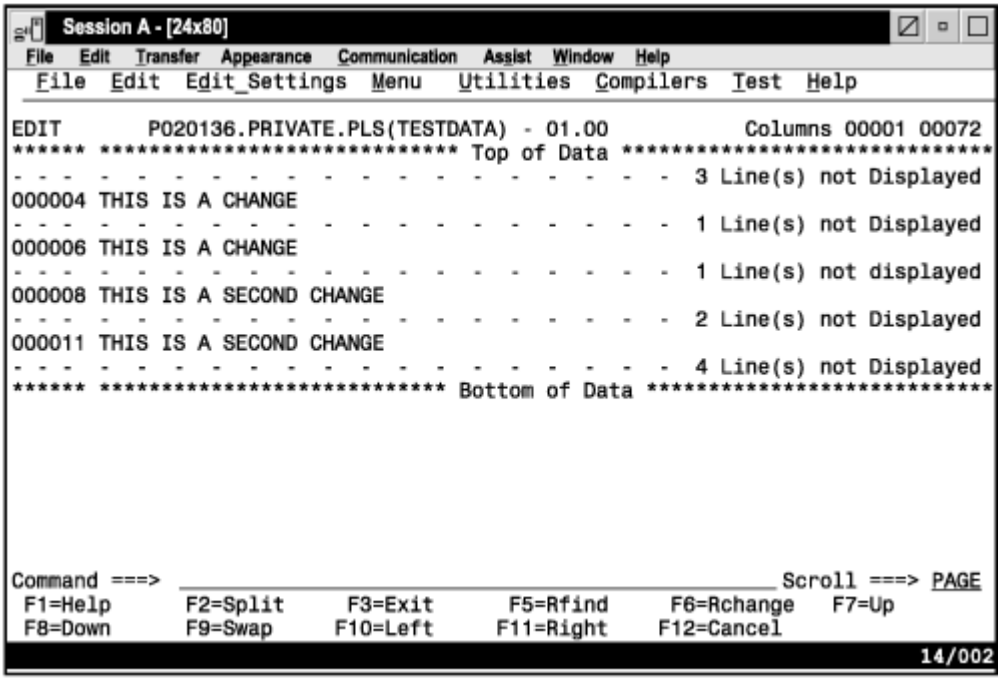


Figure 50. ISRCHGS macro - after running

ISRMASK macro

The ISRMASK macro (Figure 51 on page 130) allows data in the mask line to overlay lines. It can be used to place a comment area over existing lines in a member.

Before starting this macro, you must specify two things: a mask line and the range of lines it overlays. See “MASKLINE—Set or Query the Mask Line” on page 368 for information on creating mask lines.

Specify the range of lines by using either an OO or $$ line command. You can use O, OO, O*n*, or $, $$, $*n*, where *n* is the number of lines.

An O line command specifies that mask line data overlays only blanks in the line data. A $ line command specifies that nonblank mask line data overlays the line data. Once the mask line and range of lines have been specified, type ISRMASK on the command line and press Enter.

```
/*********************************************************************/
/*                                                                   */
/* 5647-A01 (C) COPYRIGHT IBM CORP 1995, 2003                        */
/*                                                                   */
/* ISRMASK - Overlay a line with data from the mask line.            */
/*           Use either line command 0 or $ to indicate              */
/*           which line to overlay. 0 causes nondestructive          */
/*           overlay, and $ causes a destructive overlay.            */
/*                                                                   */
/*********************************************************************/
ISREDIT MACRO NOPROCESS                 /* Wait to process       */
ISREDIT PROCESS RANGE O $               /* "O" and "$" reserved  */
IF &LASTCC = 0 THEN                     /*   for macro           */ +
  DO                                    /* If specified, get     */
    ISREDIT (CMD)   = RANGE_CMD         /*   command entered and */
    ISREDIT (FIRST) = LINENUM .ZFRANGE  /*   line number range   */
    ISREDIT (LAST)  = LINENUM .ZLRANGE
    DO WHILE &FIRST LE &LAST            /* Loop to merge data    */
                                        /*   based on which line */
                                        /*   command was entered.*/
       IF &CMD = $ THEN                 /*  If $ overlay data    */ +
          ISREDIT LINE &FIRST = (LINE) + MASKLINE
       ELSE                                     /*  - else */ +
          ISREDIT LINE &FIRST = MASKLINE + (LINE)
                                        /*     do not overlay    */
       SET FIRST = &FIRST + 1           /* Increment line num    */
      END
    SET RC = 0
  END
ELSE                                    /* Set prompt messages   */ +
  DO
    SET ZEDSMSG = &STR(ENTER "O"/"$" LINE CMD)
    SET ZEDLMSG = &STR("ISRMASK" REQUIRES AN "O" OR +
        "$" CMD TO INDICATE LINE(S) MERGED WITH MASKLINE)
    ISPEXEC SETMSG MSG(ISRZ001)
    SET RC = 12                         /* Set return code to 12 */
  END                                   /*   to keep command in  */
EXIT CODE(&RC)                          /*   command area        */
```

Figure 51. ISRMASK macro

This list explains the logical sections of the ISRMASK macro:

1. The NOPROCESS keyword on the MACRO command allows the macro to control when user input (changes to data and line commands) is processed.

   ```
   ISREDIT MACRO NOPROCESS
   ```

2. Now process user input and check if certain line commands are entered. The O and $ following the RANGE keyword specify the line commands to be processed by this macro.

   ```
   ISREDIT PROCESS RANGE O $
   ```

3. A zero return code shows that you entered an O or $ in any of its valid forms: 00-00, 0*n*, and so forth.

```
IF &LASTCC = 0 THEN
```

4. &CMD is set to O or $, whichever command was entered.

```
ISREDIT (CMD) = RANGE_CMD
```

5. &LINE1 and &LINE2 contain the first and last line numbers of the lines specified by the user line commands.

```
ISREDIT (FIRST) = LINENUM .ZFRANGE
ISREDIT (LAST) = LINENUM .ZLRANGE
DO WHILE &FIRST LE &LAST
```

6. Each line that you specify is merged with data from the mask line. Note the use of the LINE keyphrase on both sides of the assignment. The line command entered controls how the data is merged. An O specifies that the mask line data only overlays where the line contains blanks. A $ specifies that nonblank mask line data overlays line data.

```
IF &CMD = $ THEN
  ISREDIT LINE &FIRST = (LINE) + MASKLINE
ELSE
  ISREDIT LINE &FIRST = MASKLINE + (LINE)
```

7. When no line command is entered, issue a prompt message. Set a return code of 12 to keep ISRMASK displayed on the command line.

```
SET ZEDSMSG = &STR(ENTER "O"/"$" LINE CMD)
SET ZEDLMSG = &STR("ISRMASK" REQUIRES AN "O" OR +
    "$" CMD TO INDICATE LINE(S) MERGED WITH MASKLINE)
ISPEXEC SETMSG MSG(ISRZ001)
SET RC = 12
```

In the example shown in Figure 52 on page 131, the mask line is specified and the range of lines is set with the destructive $$ line command.

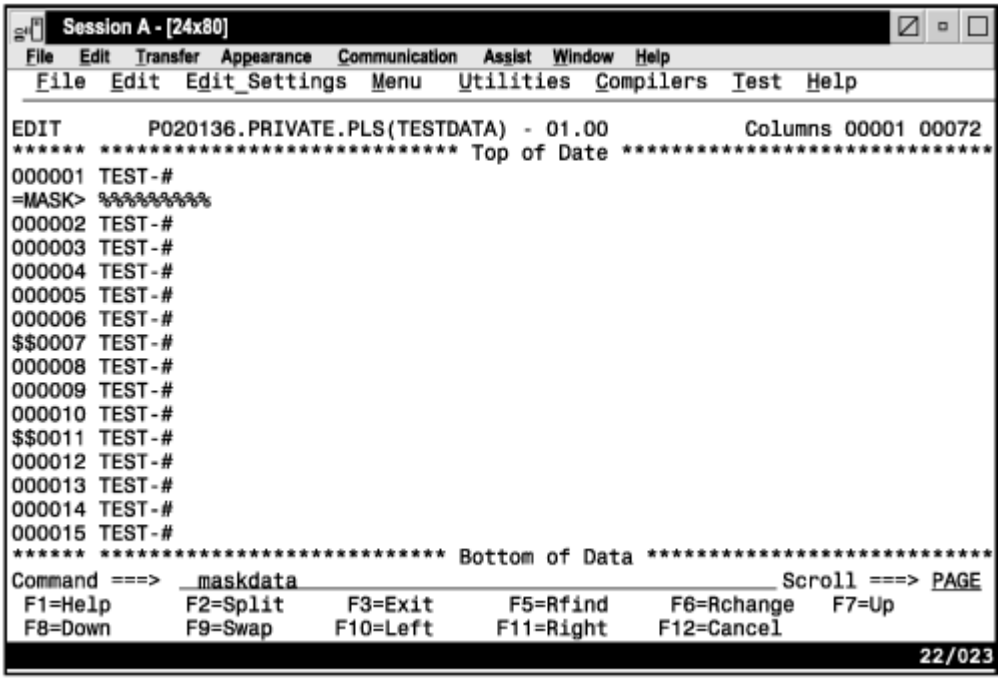


Figure 52. ISRMASK macro - before running

When you press Enter, the macro overlays the mask line onto the specified range of lines, as shown in Figure 53 on page 132.

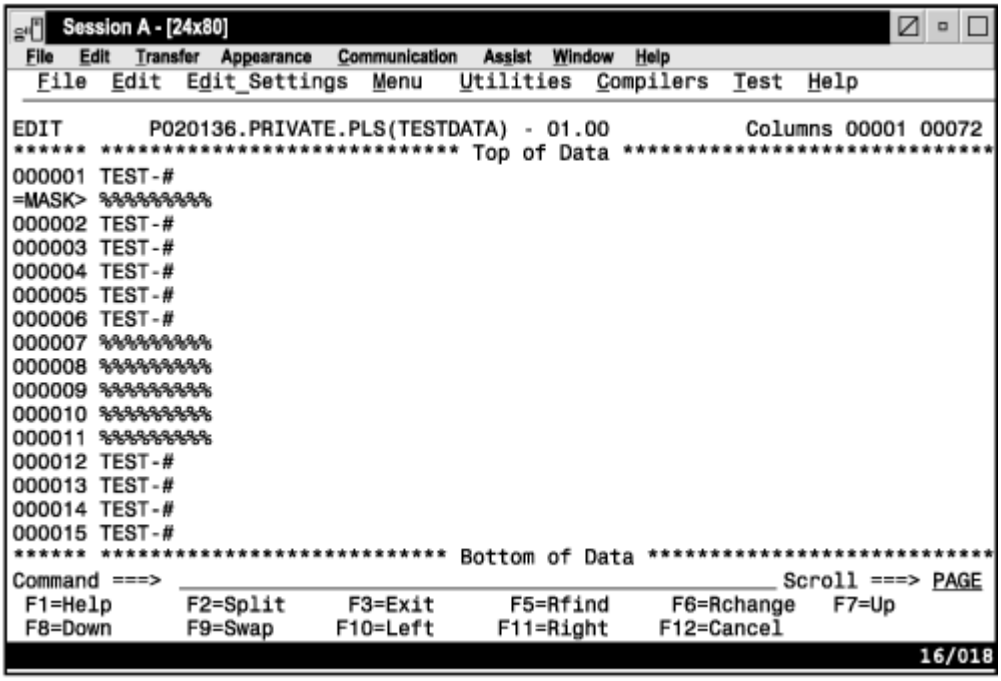


Figure 53. ISRMASK macro - after running

Part 3. Command reference

Chapter 9. Edit line commands

Edit line commands affect only a single line or block of lines. You enter line commands by typing over the 6-digit number in the line command field on one or more lines and pressing Enter. Most command definitions in this documentation consist of:

Syntax
A syntax diagram is how you type the command. It includes a description of any required or optional operands.

Description
A description explains the function and operation of the command. This description may also refer to other commands that can be used with this command.

Example
An example gives a sample usage of the line command.

Rules for entering line commands

Enter a line command by performing one of these actions:

- Typing the command in the line command field and pressing Enter.
- Placing the cursor in the data or line command field and pressing a function key to which the command is assigned.

These rules apply to all line commands:

- You can type several line commands and make multiple data changes before you press Enter. The editor displays an error message if the line command is ambiguous. Because the line commands are processed from top to bottom, it is possible to have one error message appear that masks a later error condition. Only the first error condition found is displayed. After you have corrected that error condition, processing can continue and the next error condition, if any, is displayed. If you type a line command incorrectly, you can replace it before you press Enter by retyping it, blanking it out, or entering RESET.
- Generally, you need to type over only the first 1 or 2 characters of the line number to enter a line command. Sometimes, however, typing a single character can be ambiguous. In this example, it is unclear whether the intended line command is R to repeat line 31700, or R3 to repeat the line three times:

  ```
  031600
  R31700
  031800
  ```

  In such cases, the ISPF editor assumes that you have not typed a number following the line command. If you want to repeat the line three times, you can use any of these procedures:

  – Leave the cursor on the character that immediately follows the R3:

    ```
    R31700
    ```

  – Type one or more blanks following the R3:

    ```
    R3 700
    ```

  – Type one or more blanks following the R but before the number, leaving the cursor on the character that immediately follows the 3:

    ```
    R 3700
    ```

  – Type R3 and press the Erase EOF key to clear the rest of the line command field, or press the Erase EOF key and then type R3

- You can type these line commands on the TOP OF DATA line by typing over the asterisks that appear in its line command field:

  I, I*n*
  Insert one or *n* lines ahead of the data.

  A, A*n*
  Move or copy a line or lines one or *n* times ahead of the data.

  TE, TE*n*
  Type one or *n* text lines ahead of the data.

- You can type this line command on the BOTTOM OF DATA line by typing over the asterisks:

  B, B*n*
  Move or copy a line or lines one or *n* times following the data.

Line command summary

Table 14 on page 136 summarizes line commands.

Table 14. Summary of the line commands

Command	Description
"(—Column Shift Left" on page 138	Shifts columns left two positions or the specified number of positions.
")—Column Shift Right" on page 140	Shifts columns right two positions or the specified number of positions.
"<—Data Shift Left" on page 142	Shifts data left two positions or the specified number of positions.
">—Data Shift Right" on page 144	Shifts data right two positions or the specified number of positions.
"A, AK—Specify an After destination" on page 146	Identifies the line after which copied, moved, or model lines are to be inserted.
"A, AK—Specify an After destination" on page 146	Identifies the line after which copied, moved, or model lines are to be inserted, but indicates that another destination of the form A, B, or O is still required proceeding forward through the file before the data is moved or copied to the multiple destinations specified.
"B, BK—Specify a Before destination" on page 149	Identifies the line before which copied, moved, or model lines are to be inserted.
"B, BK—Specify a Before destination" on page 149	Identifies the line before which copied, moved, or model lines are to be inserted, but indicates that another destination of the form A, B, or O is still required proceeding forward through the file before the data is moved or copied to the multiple destinations specified.
"BOUNDS—Define Boundary Columns" on page 151	Displays the column boundary definition line.
"C—Copy Lines" on page 153	Copies a line from one location to another.
"C—Copy Lines" on page 153	Copies a block of lines from one location to another.

Table 14. Summary of the line commands (continued)

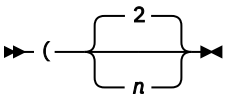
Command	Description
“COLS—Identify Columns” on page 155	Displays a position identification line.
“D—Delete Lines” on page 157	Deletes a line.
“D—Delete Lines” on page 157	Deletes a block of lines.
“F—Show the First Line” on page 158	Redisplays one or more lines at the beginning of a block of excluded lines.
“I—Insert Lines” on page 162	Inserts one or more blank data entry lines.
“L—Show the Last Line(s)” on page 164	Redisplays one or more lines at the end of a block of excluded lines.
“LC—Convert Characters to Lowercase” on page 165	Converts all uppercase alphabetic characters in a line to lowercase.
“LC—Convert Characters to Lowercase” on page 165	Converts all uppercase alphabetic characters in a block of lines to lowercase.
“M—Move Lines” on page 167	Moves a line from one location to another.
“M—Move Lines” on page 167	Moves a block of lines from one location to another.
“MASK—Define Masks” on page 169	Displays the contents of the mask when used with the I (insert), TE (text entry), and TS (text split) line commands.
“MD—Make Dataline” on page 171	Converts a `==MSG>`, `=NOTE=`, `=COLS>`, or `======` (information) line to data so that it can be saved as part of your data set.
“MD—Make Dataline” on page 171	Converts a block of `==MSG>`, `=NOTE=`, `=COLS>`, and `======` (information) lines to data so that they can be saved as part of your data set.
“O, OK—Overlay Lines” on page 173	Identifies a line over which data is to be moved or copied.
“O, OK—Overlay Lines” on page 173	Identifies the line over which data is to be moved or copied, but indicates that another destination of the form A, B, or O is still required proceeding forward through the file before the data is moved or copied to the multiple destinations specified.
“O, OK—Overlay Lines” on page 173	Identifies a block of lines over which data is to be moved or copied.
“O, OK—Overlay Lines” on page 173	Identifies a block of lines over which data is to be moved or copied, but indicates that another destination of the form A, B, or O is still required proceeding forward through the file before the data is moved or copied to the multiple destinations specified.
“R—Repeat Lines” on page 176	Repeats a line.
“R—Repeat Lines” on page 176	Repeats a block of lines.
“S—Show Lines” on page 178	Redisplays one or more lines with the leftmost indentation in a block of excluded lines.

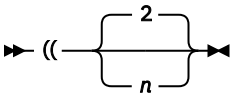
Table 14. Summary of the line commands (continued)	
Command	**Description**
“TABS—Control Tabs” on page 180	Displays the tab definition line.
“TE—Text Entry” on page 181	Inserts blank lines to allow power typing for text entry.
“TF—Text Flow” on page 184	Restructures paragraphs following deletions, insertions, splitting, and so forth.
“TS—Text Split” on page 186	Divides one or more lines so that data can be added.
“UC—Convert Characters to Uppercase” on page 188	Converts all lowercase alphabetic characters in a line to uppercase.
“UC—Convert Characters to Uppercase” on page 188	Converts all lowercase alphabetic characters in a block of lines to uppercase.
“X—Exclude Lines” on page 189	Excludes a line from a panel.
“X—Exclude Lines” on page 189	Excludes a block of lines from a panel.

(—Column Shift Left

The ((column shift left) line command moves characters on a line to the left without altering their relative spacing. Characters shifted past the current BOUNDS setting are deleted. See “Shifting data” on page 42 for more information.

Syntax





n

A number that tells the ISPF editor how many positions to shift. If you omit this operand, the default is 2.

Description

To column shift one line toward the left side of your display:

1. Type (in the line command field of the line to be shifted. Beside the command, type a number other than 2 if you want to shift the line other than 2 columns.
2. Press Enter.

To column shift a block of lines toward the left side of your display:

1. Type ((in the line command field of the first line to be shifted. Beside the command, type a number other than 2 if you want to shift the block of lines other than 2 columns.

2. Type ((in the line command field of the last line to be shifted. You can scroll (or use FIND or LOCATE) between typing the first ((and the second ((, if necessary.
3. Press Enter. The lines that contain the two ((commands and all of the lines between them are column shifted to the left.

The BOUNDS setting limits column shifting. If you shift columns beyond the current BOUNDS setting, the editor deletes the text beyond the BOUNDS without displaying a warning message.

Examples

To shift a group of lines to the left three column positions, specify the number of columns and the range in the line command field, as shown in Figure 54 on page 139.

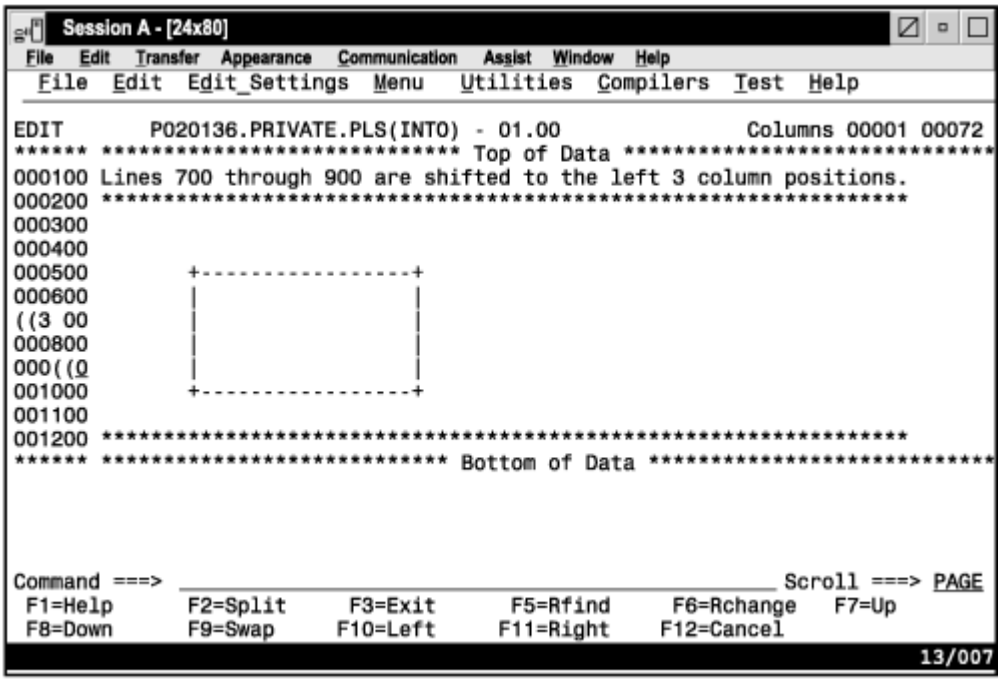


Figure 54. Before the ((Column Shift Left) line command

Press Enter and the editor shifts the specified lines three columns to the right. See Figure 55 on page 140.

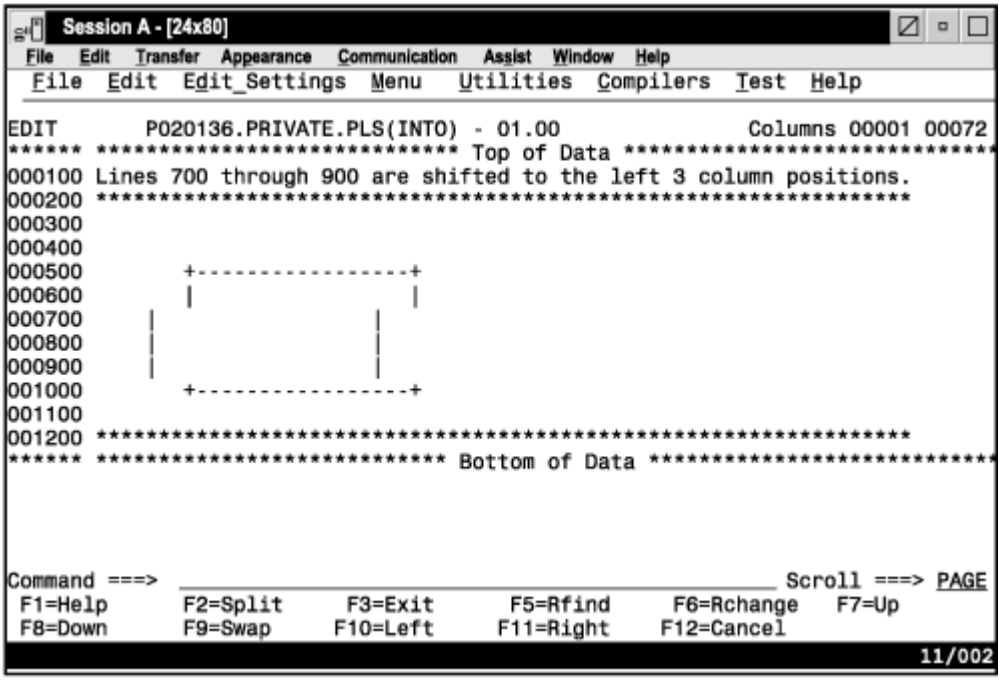
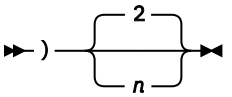


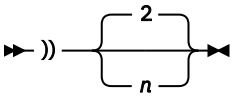
Figure 55. After the ((Column Shift Left) line command

)—Column Shift Right

The) (column shift right) line command moves characters on a line to the right without altering their relative spacing. Characters shifted past the current BOUNDS setting are deleted. See “Shifting data” on page 42 for more information.

Syntax





n

A number that tells the ISPF editor how many positions to shift. If you omit this operand, the default is 2.

Description

To column shift one line toward the right side of your display:

1. Type) in the line command field of the line to be shifted. Beside the command, type a number other than 2 if you want to shift the data other than 2 columns.
2. Press Enter.

To column shift a block of lines toward the right side of your display:

1. Type)) in the line command field of the first line to be shifted. Beside the command, type a number other than 2 if you want to shift the block of lines other than 2 columns.

2. Type)) in the line command field of the last line to be shifted. You can scroll (or use FIND or LOCATE) between typing the first)) and the second)), if necessary.
3. Press Enter. The lines that contain the two)) commands and all of the lines between them are column shifted to the right.

The BOUNDS setting limits column shifting. If you shift columns beyond the current BOUNDS setting, the editor deletes the text beyond the BOUNDS without displaying a warning message.

Examples

To shift a group of lines to the right 3 column positions, specify the number of columns and the range in the line command field, as shown in Figure 56 on page 141.

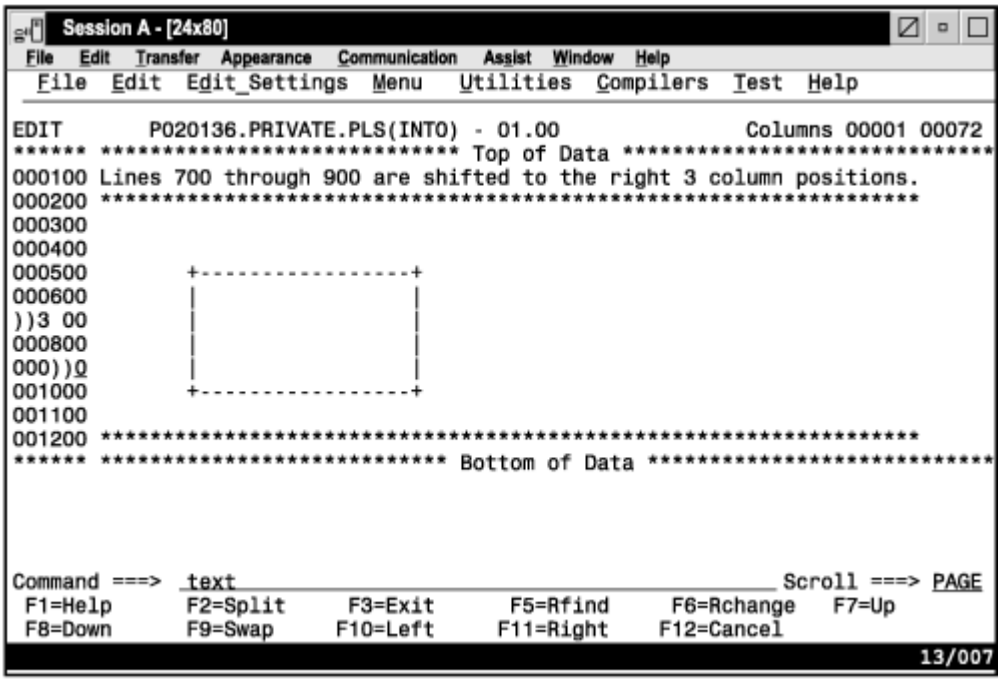


Figure 56. Before the) (Column Shift Right) line command

Figure 57 on page 142 shows that when you press Enter, the editor shifts the specified lines to the right 3 columns.

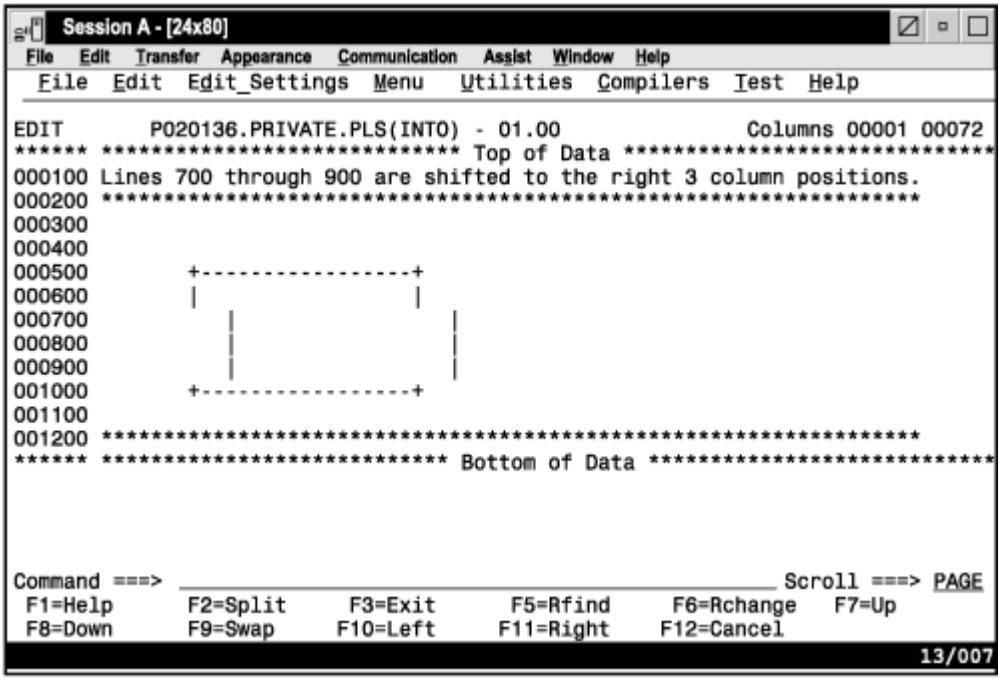
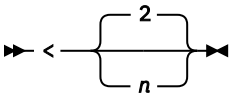


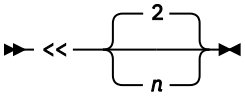
Figure 57. After the) (Column Shift Right) line command

<—Data Shift Left

The < (data shift left) line command moves the body of a program statement to the left without shifting the label or comments. This command attempts to prevent loss of data. See “Shifting data” on page 42 for more information.

Syntax





n

A number that tells the ISPF editor how many positions to shift. If you omit this operand, the default is 2.

Description

To data shift one line toward the left side of your display:

1. Type < in the line command field of the line to be shifted. Beside the command, type a number other than 2 if you want to shift the data other than 2 columns.
2. Press Enter.

To data shift a block of lines toward the left side of your display:

1. Type << in the line command field of the first line to be shifted. Beside the command, type a number other than 2 if you want to shift the block of lines other than 2 columns.

2. Type << in the line command field of the last line to be shifted. You can scroll (or use FIND or LOCATE) between typing the first << and the second <<, if necessary.
3. Press Enter. The lines that contain the two << commands and all of the lines between them are data shifted to the left.

The BOUNDS setting limits data shifting. If you shift data beyond the current BOUNDS setting, the text stops at the left bound and the shifted lines are marked with ==ERR> flags. If an error occurs in an excluded line, you can find the error with LOCATE, and remove the error flag by using RESET.

Examples

To use a data shift to shift the body of the program statements (on lines 7 through 10) 7 spaces to the left, specify the shift and the range in the line command field, as shown in Figure 58 on page 143.

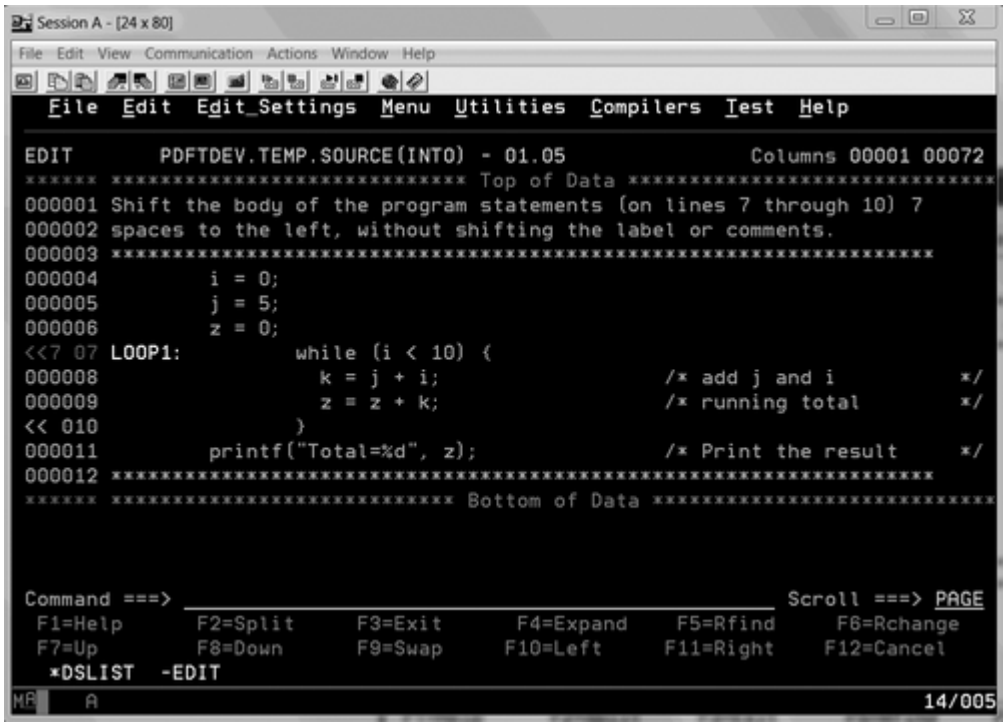


Figure 58. Before the < (Data Shift Left) line command

When you press Enter, the editor deletes 7 blanks on the specified lines, as shown in Figure 59 on page 144. Notice that the editor does not shift the label on line 7 or the comments on lines 8 and 9.

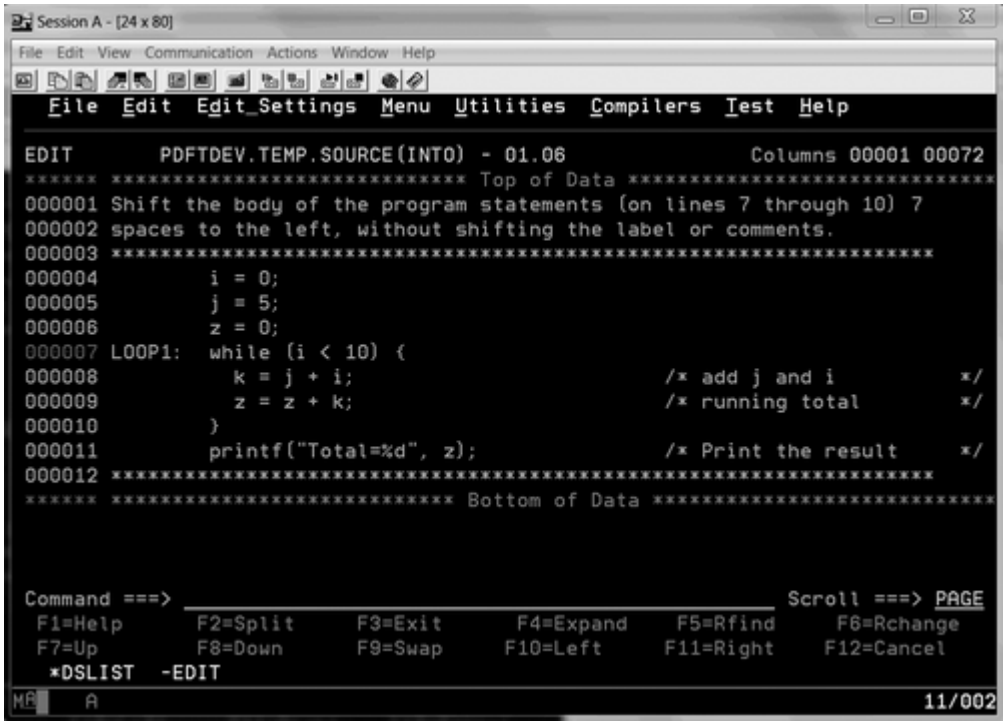
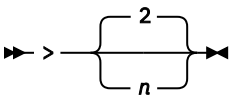


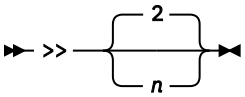
Figure 59. After the < (Data Shift Left) line command

>—Data Shift Right

The > (data shift right) line command moves the body of a program statement to the right without shifting the label or comments. This command attempts to prevent loss of data. See “Shifting data” on page 42 for more information.

Syntax





n

A number that tells the ISPF editor how many positions to shift. If you omit this operand, the default is 2.

Description

To data shift one line toward the right side of your display:

1. Type > in the line command field of the line to be shifted. Beside the command, type a number other than 2 if you want to shift the line other than 2 columns.
2. Press Enter.

To data shift a block of lines toward the right side of your display:

1. Type >> in the line command field of the first line to be shifted. Beside the command, type a number other than 2 if you want to shift the block of lines other than 2 columns.
2. Type >> in the line command field of the last line to be shifted. You can scroll (or use FIND or LOCATE) between typing the first >> and the second >>, if necessary.
3. Press Enter. The lines that contain the two >> commands and all of the lines between them are data shifted to the right.

The BOUNDS setting limits data shifting. If you shift data beyond the current BOUNDS setting, the text stops at the right bound and the shifted lines are marked with ==ERR> flags. If an error occurs in an excluded line, you can find the error with the LOCATE command, and remove the error flag by using RESET.

Examples

To use a data shift to shift the body of the program statements (on lines 7 through 10) 7 spaces to the right, specify the shift and the range in the line command field, as shown in Figure 60 on page 145.

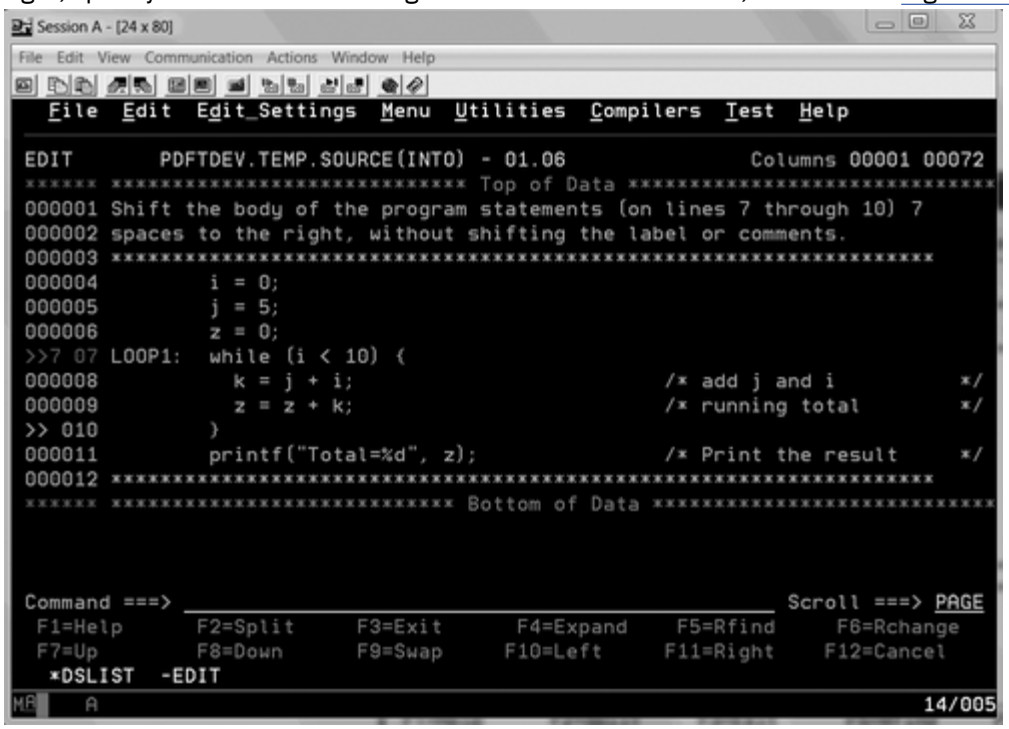


Figure 60. Before the > (Data Shift Right) line command

When you press Enter, the editor inserts 7 blanks on the specified lines. See Figure 61 on page 146. Notice that the editor does not shift the label on line 7 or the comments on lines 8 and 9.

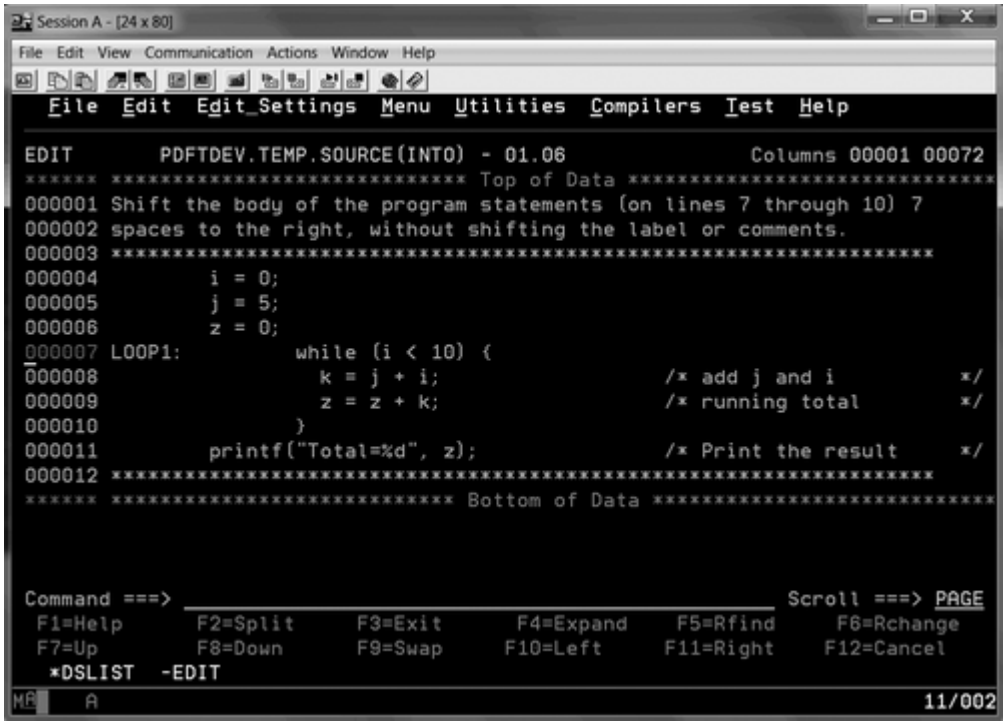


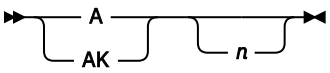
Figure 61. After the > (Data Shift Right) line command

A, AK—Specify an After destination

When data is to be moved or copied, the A (after) line command specifies the line after which the data is to be placed. When data is to be moved or copied to multiple destinations, the A (after) line command specifies the final destination line after which the data is to be placed.

When data is to be moved or copied to multiple destinations, the AK (after, multiple targets) line command specifies each multiple destination line (apart from the final destination line) after which the data is to be placed.

Syntax



n

A number that tells the ISPF editor to repeat the associated line command a specified number of times. If you do not type a number, or if the number you type is 1, the editor performs the command only once. The number does not affect associated primary commands.

Description

To specify that data is to be moved or copied after a specific line:

1. Type one of the commands that are listed in this table. Line commands are typed in the line command field. Primary commands are typed on the command line.

Table 15. Line and primary commans for A and AK

Line commands	Primary commands
“C—Copy Lines” on page 153	“COPY—Copy Data” on page 212
“M—Move Lines” on page 167	“MODEL—Copy a Model into the Current Data Set” on page 251
	“MOVE—Move Data” on page 254

2. To specify a *single* destination for the data that is to be moved or copied, type A in the line command field of the line that the moved or copied data is to follow. If you are specifying the destination for a line command, a number after the A line command specifies the number of times the other line command is performed. However, a number after the A command has no affect on a primary command.

   To specify *multiple* destinations for the data that is to be moved or copied:

   a. Type AK in the line command field of each line (apart from the final destination) that the moved or copied data is to follow.

   b. Type A in the line command field of the final line that the moved or copied data is to follow.

3. Press Enter.

4. Some of the commands in the preceding table can cause another panel to be displayed if more information is needed. If so, fill in the required information and press Enter to move, copy, or insert the data. See the information about the specified command if you need help.

   If no panel is displayed, the data is moved, copied, or inserted when you press Enter in step “3” on page 147.

You must always specify a destination except when you are using a primary command to move, copy, or insert data into a member or data set that is empty.

Two other line commands that are used to specify a destination are the B (before) command and the O (overlay) command. See “B, BK—Specify a Before destination” on page 149 and “O, OK—Overlay Lines” on page 173 for more information.

Examples

Figure 62 on page 148 shows how you can move data with the M and A line commands. Type M in the line command field of the line you want to move. Type A in the line command field of the line that you want the moved line to follow.

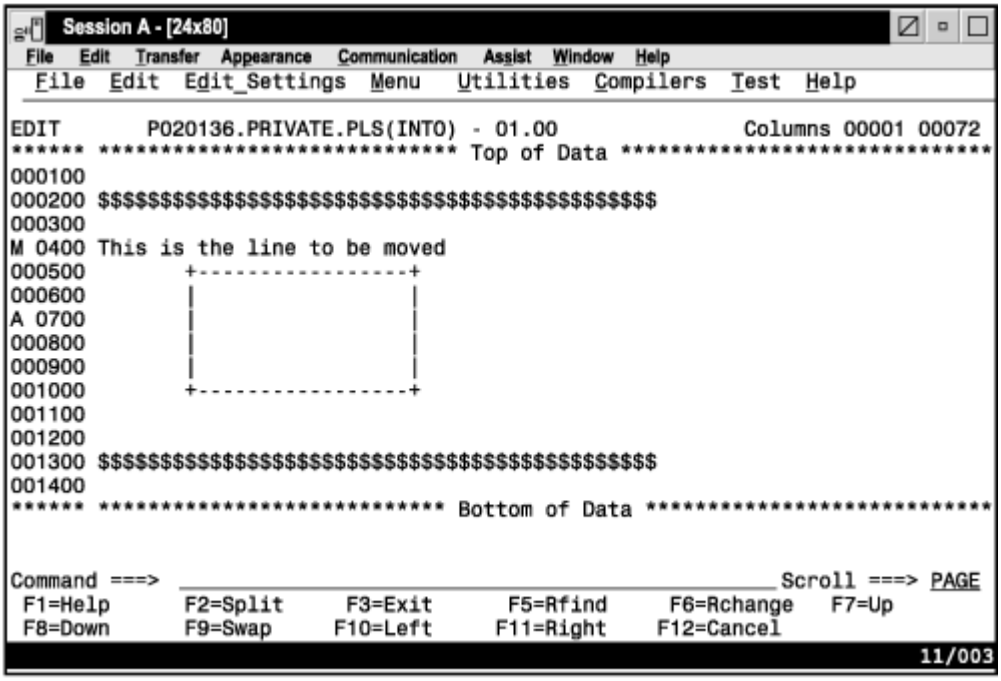


Figure 62. Before the A (After) line command

When you press Enter, the line where you typed the M command is moved after the line where you typed the A command. See Figure 63 on page 148.

Note:

1. If you press Enter before specifying where you want the data to go, the editor displays a `MOVE/COPY pending` message at the top of the panel. The line does not move until you specify a destination.
2. The AK line command indicates that another destination of the form of A, B, or O is still required proceeding forward through the file before the data is moved or copied to the multiple destinations specified.

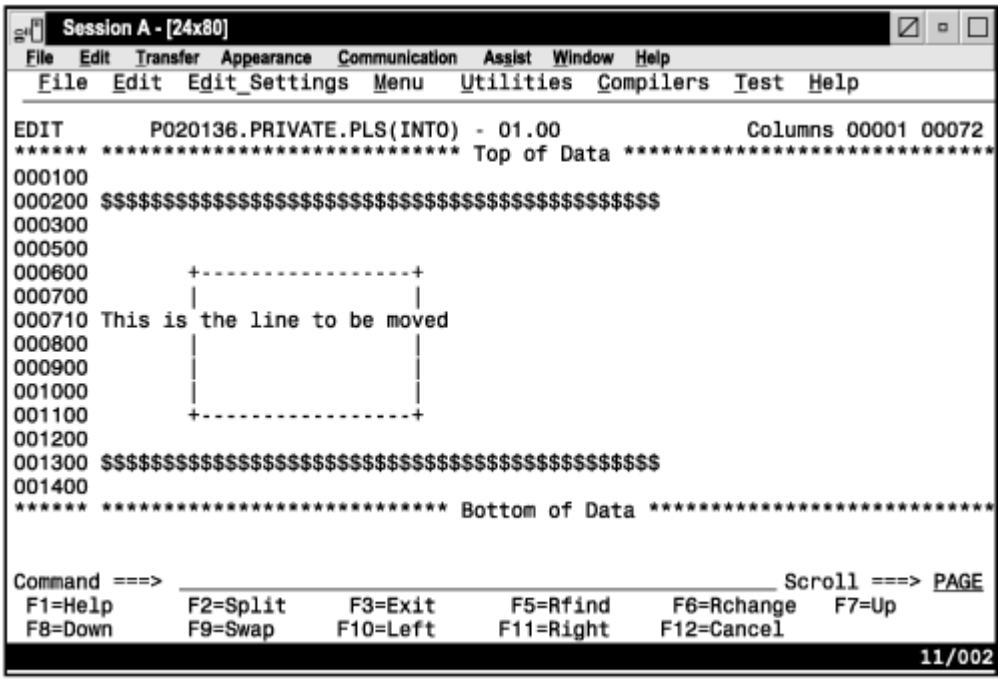


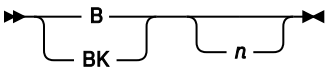
Figure 63. After the A (After) line command

B, BK—Specify a Before destination

When data is to be moved or copied, the B (before) line command specifies the line before which the data is to be placed. When data is to be moved or copied to multiple destinations, the B (before) line command specifies the final destination line before which the data is to be placed.

When data is to be moved or copied to multiple destinations, the BK (before, multiple targets) line command specifies each multiple destination line (apart from the final destination line) before which the data is to be placed.

Syntax



n

A number that tells the ISPF editor to repeat the associated line command a specified number of times. If you do not type a number, or if the number you type is 1, the command is not repeated. For associated primary commands, this number has no effect.

Description

To specify that data is to be moved, copied, or inserted before a specific line:

1. Type one of the commands that are listed in this table. Line commands are typed in the line command field. Primary commands are typed on the command line.

   Table 16. Line and primary commands for B

Line commands	Primary commands
“C—Copy Lines” on page 153	“COPY—Copy Data” on page 212
“M—Move Lines” on page 167	“MODEL—Copy a Model into the Current Data Set” on page 251
	“MOVE—Move Data” on page 254

2. To specify a *single* destination for the data that is to be moved or copied, type B in the line command field of the line that the moved or copied data is to precede. If you are specifying the destination for a line command, a number after the B line command specifies the number of times the other line command is performed. However, a number after the B command has no affect on a primary command.

   To specify *multiple* destinations for the data that is to be moved or copied:

   a. Type BK in the line command field of each line (apart from the final destination) that the moved or copied data is to precede.

   b. Type B in the line command field of the final line that the moved or copied data is to precede.

3. Press Enter.
4. Some of the commands in the preceding table can cause another panel to be displayed if more information is needed. If so, fill in the required information and press Enter to move, copy, or insert the data. See the information about the specified command if you need help.

If no panel is displayed, the data is moved, copied, or inserted when you press Enter in step “3” on page 149.

You must always specify a destination except when you are using a primary command to move, copy, or insert data into a member or data set that is empty.

Two other line commands that are used to specify a destination are the A (after) command and the O (overlay) command. See “A, AK—Specify an After destination” on page 146 and “O, OK—Overlay Lines” on page 173 for more information.

Examples

Figure 64 on page 150 shows how you can copy data with the C and B line commands. Type C in the line command field of the line you want to copy. Type B in the line command field of the line that the copied line precedes.

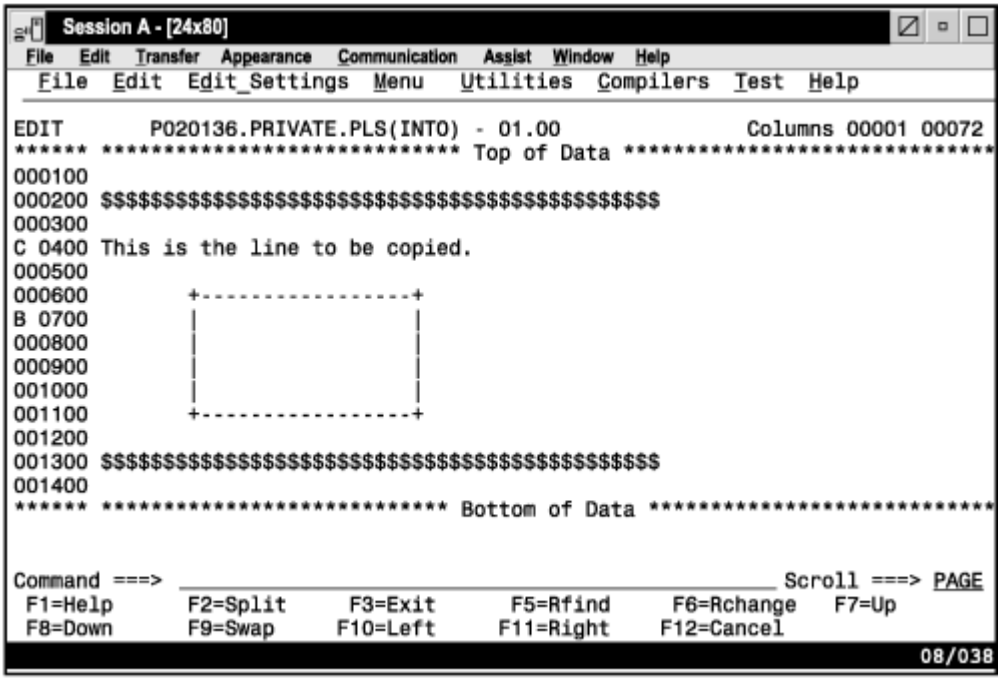


Figure 64. Before the B (Before) line command

When you press Enter, the line where you typed the C command is moved before the line where you typed the B command, as shown in Figure 65 on page 151.

Note:

1. If you press Enter before specifying where you want the data to go, the editor displays a `MOVE/COPY pending` message at the top of the panel. The line does not move until you specify a destination.
2. The BK line command indicates that another destination of the form A, B, or O is still required proceeding forward through the file before the data is moved or copied to the multiple destinations specified.

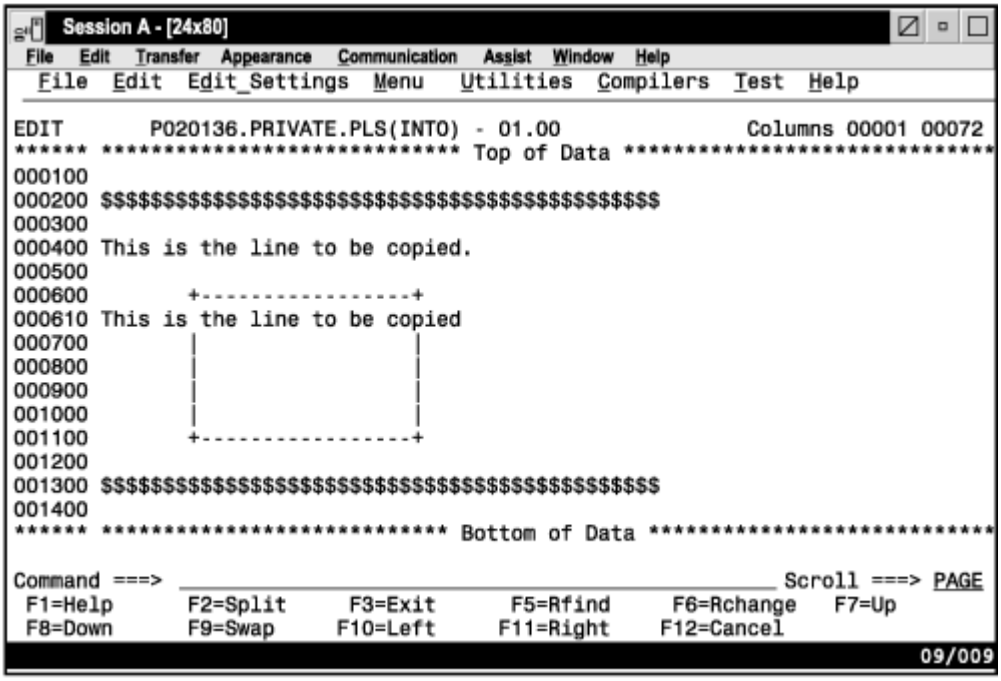
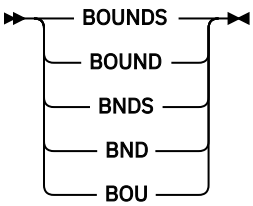


Figure 65. After the B (Before) line command

BOUNDS—Define Boundary Columns

The BOUNDS line command displays the boundary definition line.

Syntax



Description

The BOUNDS line command provides an alternative to setting the boundaries with the BOUNDS primary command or macro command; the effect on the member or data set is the same. However, if you use both the BOUNDS primary command and the BOUNDS line command in the same interaction, the line command overrides the primary command.

To display the boundary definition (=BNDS>) line:

1. Type BOUNDS in the line command field of any line that is not flagged.
2. Press Enter. The boundary definition line is inserted in the data set or member.

To change the BOUNDS settings:

1. Delete a < or > character. The < character shows the left BOUNDS setting and the > character shows the right BOUNDS setting.
2. Move the cursor to a different location on the =BNDS> line.

Note: You can use the COLS line command with the BOUNDS line command to help check and reposition the BOUNDS settings. The COLS line command displays the column identification line.

3. Retype the deleted character or characters.

   Note: The < character must be typed to the left of the > character.

4. Press Enter. The new BOUNDS settings are now in effect.

To revert to the default settings:

1. Display the boundary definition line.
2. Blank out its contents with the Erase EOF key or the Del (delete) key.
3. Press Enter.

To remove the boundary definition line from the panel, you can either type D in the line command field that contains the =BNDS> flag or type one of these commands on the command line:

- RESET (to reset all flagged lines), or
- RESET SPECIAL (to reset only the special lines)

The column numbers are always data column numbers (see "Referring to column positions" on page 106). Thus, for a variable format data set with number mode on, data column 1 is column 9 in the record.

See "Edit boundaries" on page 23 for more information, including tables that show commands affected by BOUNDS settings and default bounds settings for various types of data sets.

Examples

Figure 66 on page 152 shows the boundary definition line displayed with the column identification line. Type BOUNDS in the line command field.

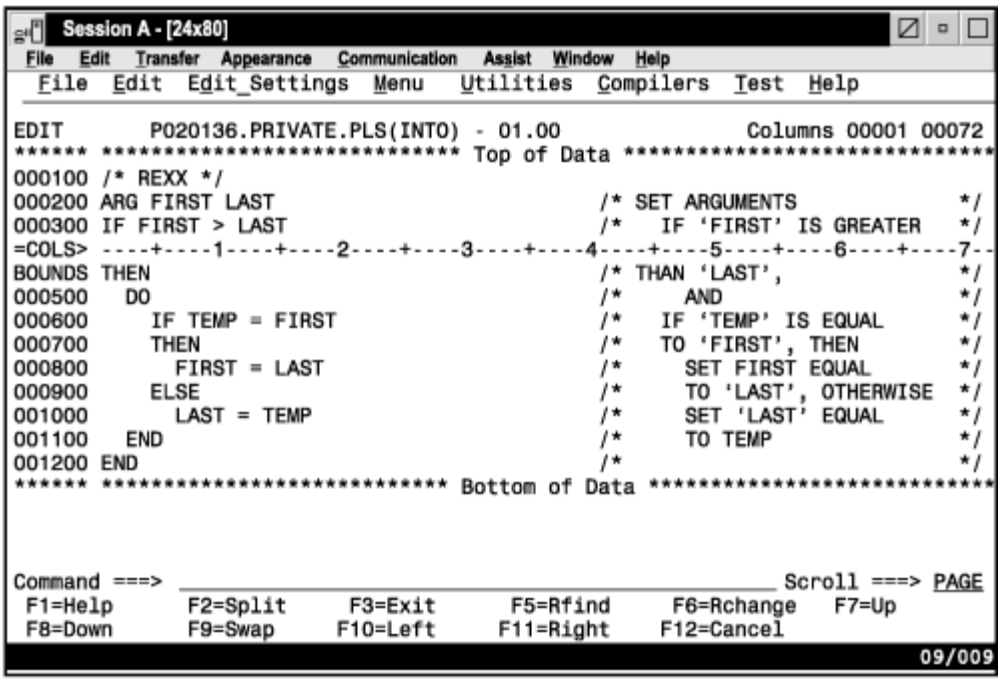


Figure 66. Before the BOUNDS line command

Figure 67 on page 153 shows that when you press Enter, the editor inserts the BOUNDS line and sets the left bound at column 43 and the right bound at column 69.

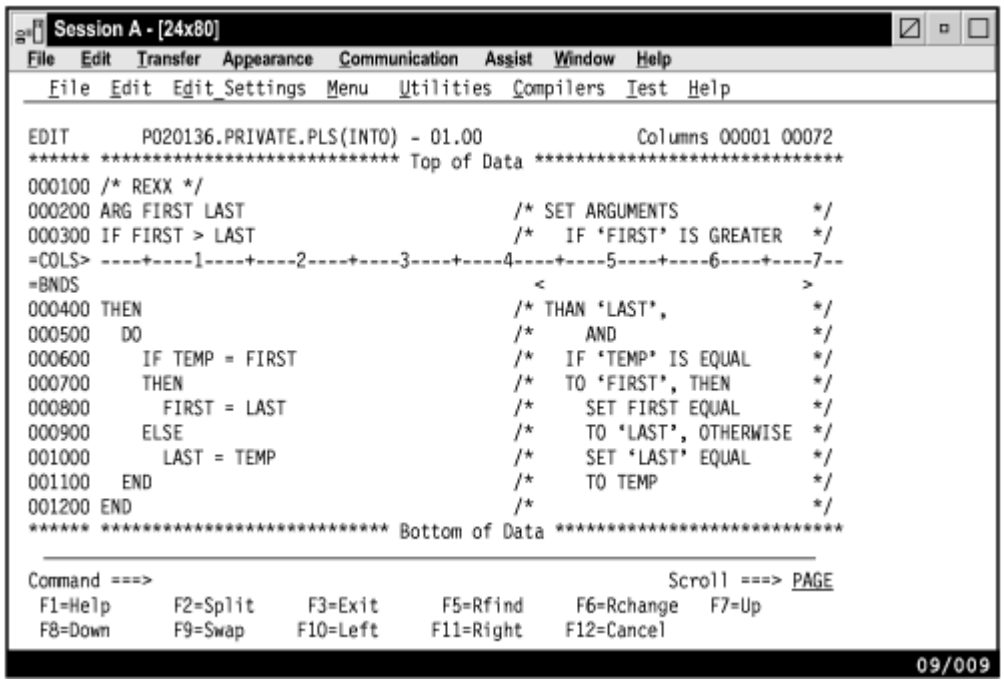
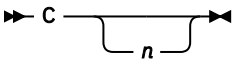


Figure 67. After the BOUNDS line command

C—Copy Lines

The C (copy) line command copies lines from one location to another.

Syntax



n

The number of lines to be copied. If you do not type a number, or if the number you type is 1, only the line on which you type C is copied.

Description

To copy one or more lines within the same data set or member:

1. Type C in the line command field of the line to be copied. If you also want to copy one or more lines that immediately follow this line, type a number greater than 1 after the C command.
2. Next, specify the destination of the line to be copied by using either the A (after), B (before), or O (overlay) line command.
3. Press Enter. The line or lines are copied to the new location.

To copy a block of lines within the same data set or member:

1. Type CC in the line command field of both the first and last lines to be copied. You can scroll (or use FIND or LOCATE) between typing the first CC and the second CC, if necessary.

2. Use the A (after), B (before), or OO (overlay) command to show where the copied lines are to be placed. Notice that when you use the block form of the C command (CC) to copy and overlay lines, you should also use the block form of the O command (OO).
3. Press Enter. The lines that contain the two CC commands and all of the lines between them are copied to the new location.

Note: Only blank characters in the lines specified with O or OO are overlaid with characters in the corresponding columns from the source lines. Characters that are not blank are not overlaid. The overlap affects only those characters within the current column boundaries.

To copy lines to another data set or member:

1. Type either CREATE or REPLACE on the command line.
2. Use one of the forms of the C command described previously.
3. Press Enter.
4. On the next panel that PDF displays, type the name of the data set or member that you want to create or replace.
5. Press Enter. The lines are copied to the data set or member that you specified.

Note: To copy lines into an existing data set or member without replacing that data set or member, edit the existing data set or member and use the COPY primary or macro command.

Examples

The example in Figure 68 on page 154 shows how to copy data by using the C and B line commands. Type C in the line command field of the line you want to copy. Type B in the line command field of the line that you want the copied line to precede.

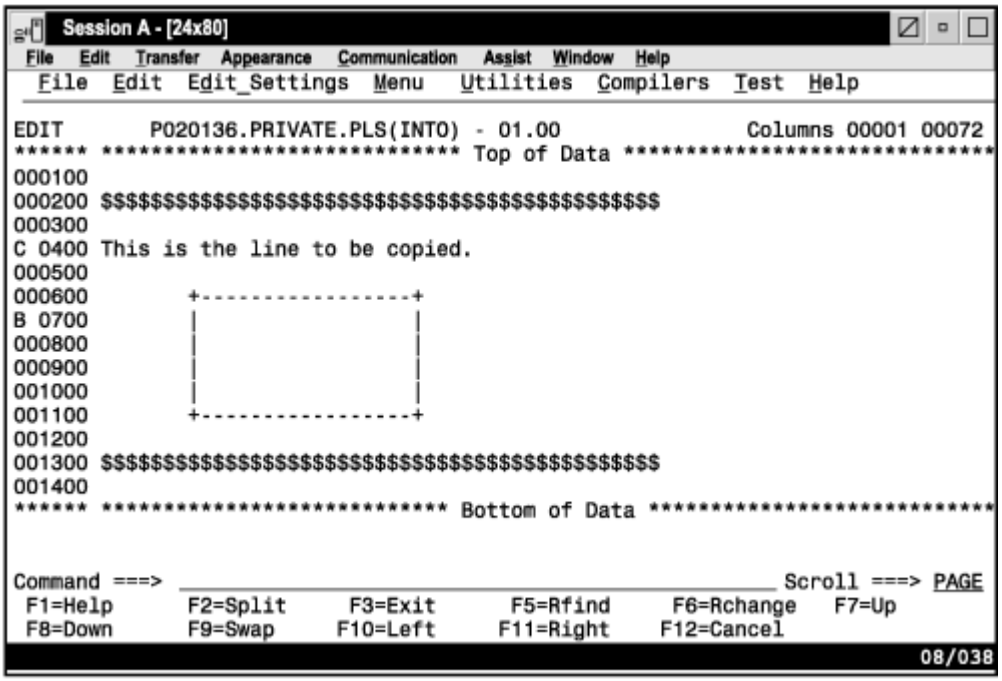


Figure 68. Before the C (Copy) line command

When you press Enter, the line where you typed the C command is copied preceding the line where you typed the B command, as shown in Figure 69 on page 155.

Note: If you press Enter before specifying where you want the data to go, the editor displays a `MOVE/COPY pending` message at the top of the panel. The line is not copied until you specify a destination.

While in MOVE/COPY pending mode, you can issue FIND and LOCATE primary commands, but you cannot use a CHANGE command to change data until after the copy completes.

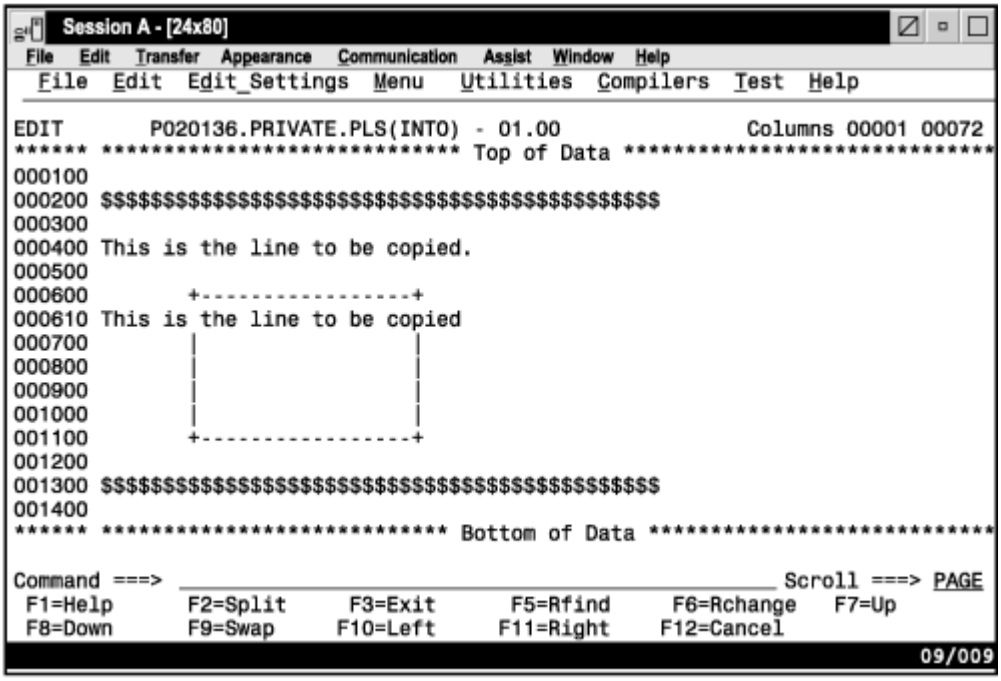
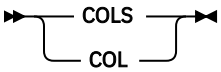


Figure 69. After the C (Copy) line command

COLS—Identify Columns

The COLS line command displays a column identification line.

Syntax



Description

To display the column identification (=COLS>) line:

1. Type COLS in the line command field of any line.
2. Press Enter.

   The column identification line is inserted in the data set or member after the line in which you entered COLS. The column identification line moves with the rest of the data when you scroll through the data set or member. To display a non-scrolling, non-editable column indicator line, use the COLS primary command. See “COLS—Display Fixed Columns Line” on page 208.

Note: You can use the COLS line command with the BOUNDS line command to help check and reposition the bounds settings.

To remove the column identification line from the panel, you can either type D in the line command field that contains the =COLS> flag, or type one of these commands on the command line:

- RESET (to reset all flagged lines), or

- RESET SPECIAL (to reset only the special lines)

Examples

The example in Figure 70 on page 156 shows the column identification line displayed with the boundary definition line. The COLS command is typed in the line command field.

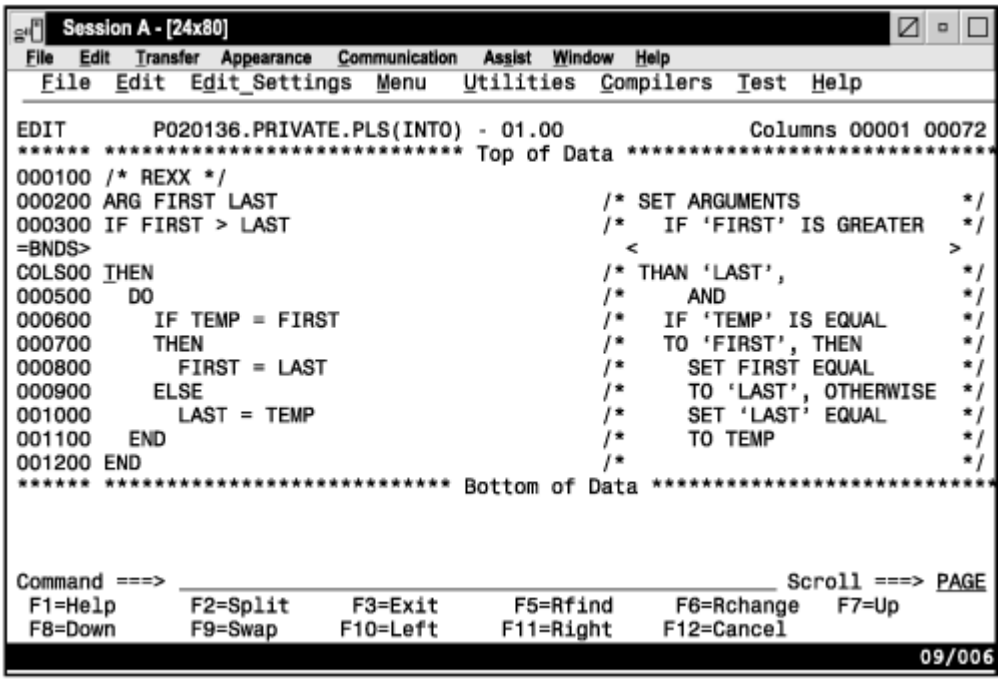


Figure 70. Before the COLS line command

When you press Enter, the editor inserts the COLS line, as shown in Figure 71 on page 156.

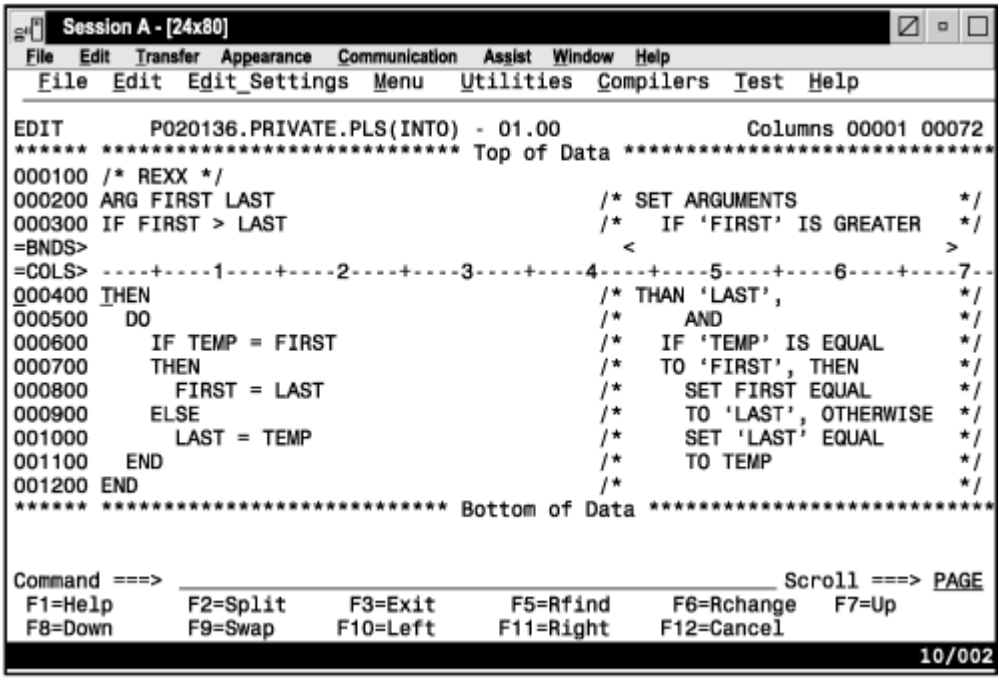
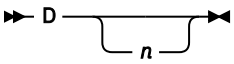


Figure 71. After the COLS line command

D—Delete Lines

The D (delete) line command deletes lines from your display.

Syntax



n

The number of lines to be deleted. If you do not type a number, or if the number you type is 1, only the line on which you type D is deleted.

Description

To delete one or more lines:

1. Type D in the line command field of the line to be deleted. If you also want to delete one or more lines that immediately follow this line, type a number greater than 1 after the D command.
2. Press Enter.

   The line or lines are deleted.

To delete a block of lines:

1. Type DD in the line command field of both the first and last lines to be deleted. You can scroll (or use FIND or LOCATE) between typing the first DD and the second DD, if necessary.
2. Press Enter.

   The lines that contain the two DD commands and all of the lines between them are deleted.

Examples

To delete two lines, type D2 in the line command field of the first line you want to delete. See Figure 72 on page 158.

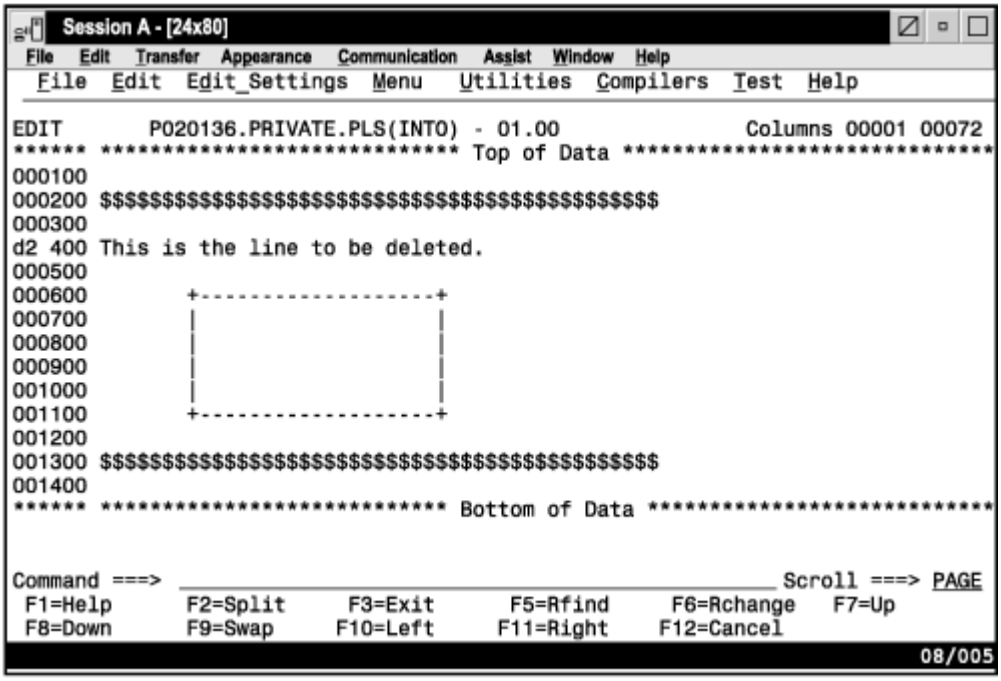


Figure 72. Before the D (Delete) line command

When you press Enter, the editor deletes the two lines specified. See Figure 73 on page 158.

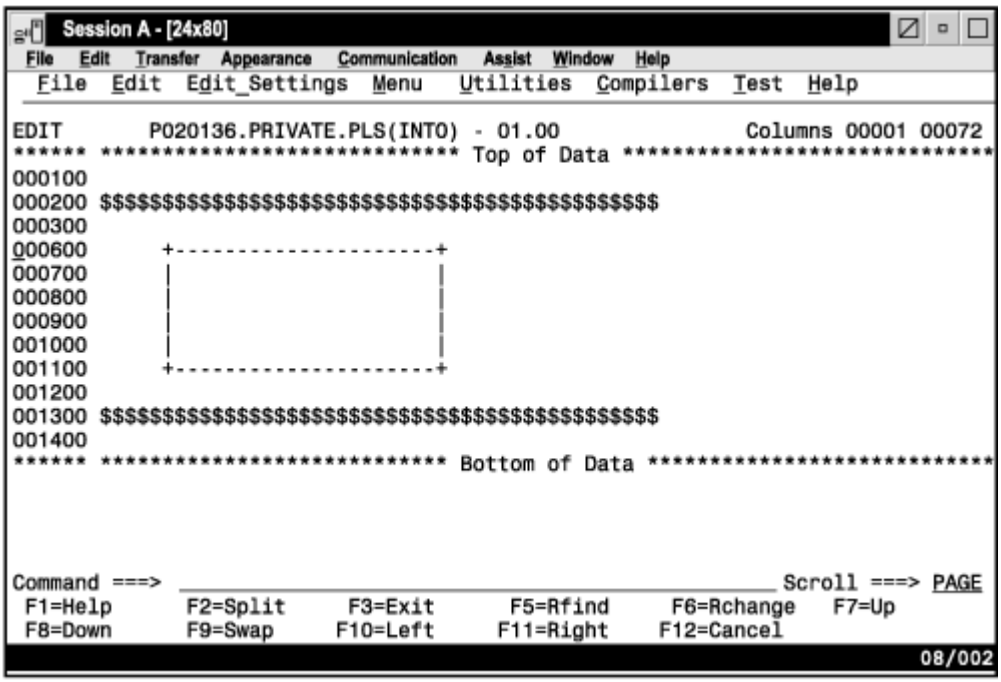
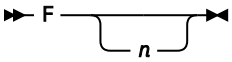


Figure 73. After the D (Delete) line command

F—Show the First Line

The F (show first line) line command redisplays one or more lines at the beginning of a block of excluded lines. See “Redisplaying excluded lines” on page 58 for more information about excluding lines.

Syntax



n

The number of lines to be redisplayed. If you do not type a number, or if the number you type is 1, only one line is redisplayed.

Description

To redisplay the first line or lines of a block of excluded lines:

1. Type F in the line command field next to the dashed line that shows where lines have been excluded. The message in the dashed line tells you how many lines are excluded. If you want to redisplay more than one line, type a number greater than 1 after the F command.
2. Press Enter.

   The first line or lines are redisplayed.

Examples

The example in Figure 74 on page 159 shows how to redisplay the excluded lines of a member. To redisplay the first three lines, type F3 in the line command field.

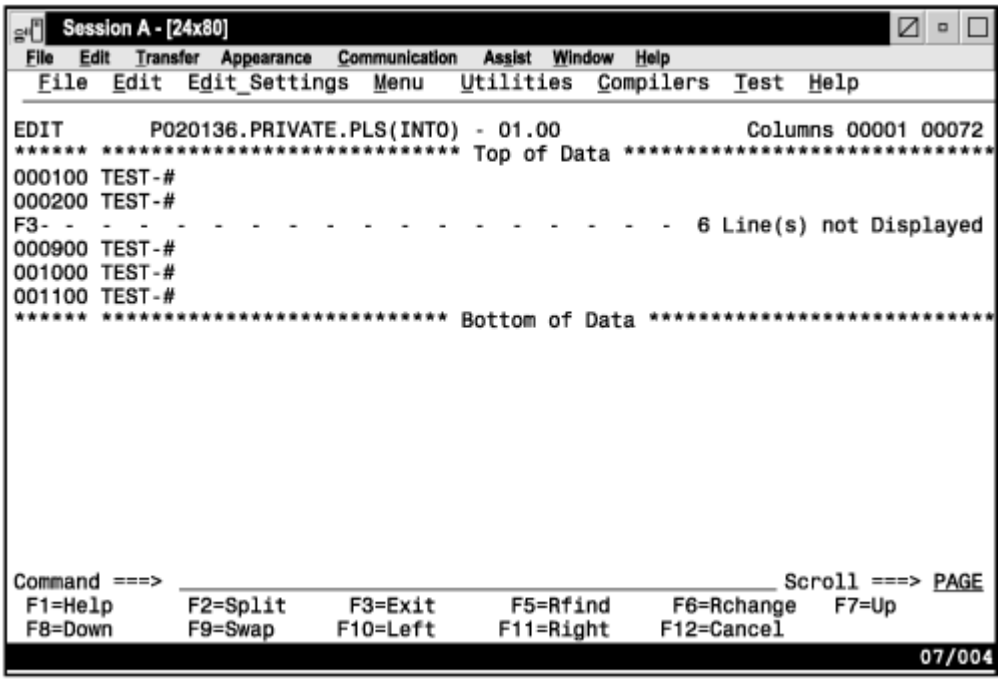


Figure 74. Before the F (Show First Line) line command

When you press Enter, the editor displays the first three lines, as shown in Figure 75 on page 160. Excluded lines do not need to be displayed again before saving the data. The excluded lines message line is never saved.

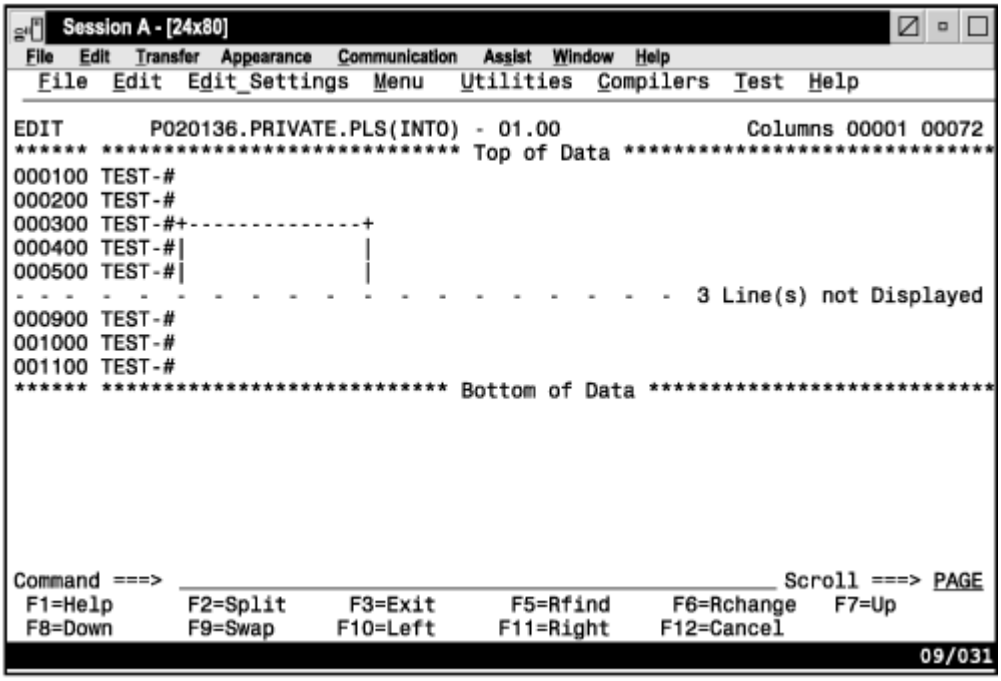
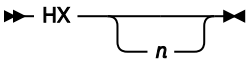


Figure 75. After the F (Show First Line) line command

HX—Show data in hexadecimal format

The HX (hexadecimal) line command displays characters in a data set or member in hexadecimal format.

Syntax



n

The number of lines to be displayed in hexadecimal format. If you do not type a number, or if the number you type is 1, only the line on which you type HX is displayed in hexadecimal format.

Description

To display characters on one or more lines in hexadecimal format:

1. Type HX in the line command field of the source code line that contains the characters you want to display in hexadecimal format. If you also want to display characters in hexadecimal format on one or more lines that immediately follow this line, type a number greater than 1 after the HX command.
2. Press Enter. The characters on the source code lines are displayed in hexadecimal format.

To display characters in a block of lines in hexadecimal format:

1. Type HXX in the line command field of both the first and last source code lines that you want to display in hexadecimal format. You can scroll (or use FIND or LOCATE) between typing the first HXX and the second HXX, if necessary.
2. Press Enter. The characters in the source code lines that contain the two HXX commands and in all of the source code lines between them are displayed in hexadecimal format.

When the file is not being displayed *totally* in hexadecimal format (that is, the HEX ON primary command is not in effect), records that have been marked by a HX or a HXX edit line command are displayed as a set of four lines, similar to the way records are displayed when the HEX ON primary command is used. The HX and HXX edit line commands act in a toggle manner to change the display of records. That is, if the record is already displayed in hexadecimal format due to an HX or HXX command, then issuing another HX or HXX command turns off the hexadecimal display for the record.

Note: The effect of any previous HX or HXX commands are canceled by the command, HEX OFF. HX is not available with FASTPATH panels such as ISREDDE.

Examples

Figure 76 on page 161 shows how to use the HX command without any operands. To display a line in hexadecimal format, type HX in the line command field of the line you want to display.

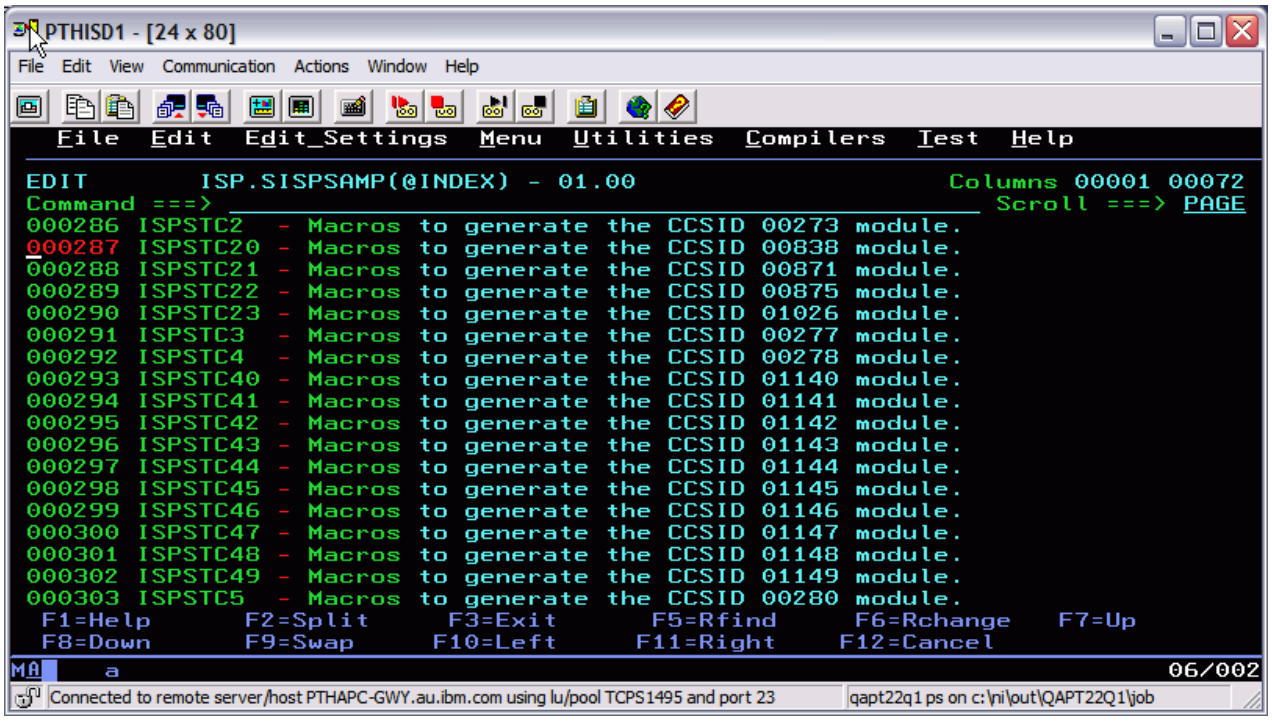


Figure 76. Before the HX (display in hexadecimal format) line command

When you press Enter, the editor converts the characters in the line to hexadecimal format. See Figure 77 on page 162.

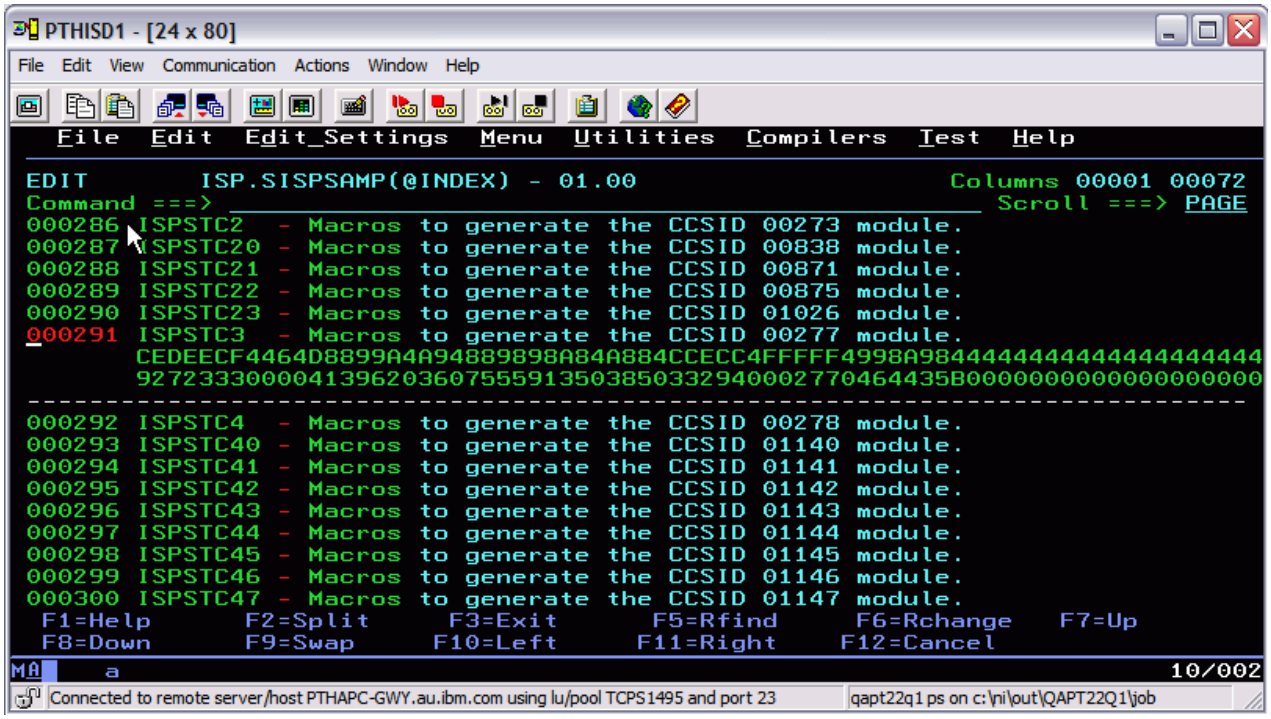
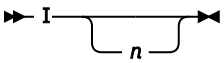


Figure 77. After the HX (display in hexadecimal format) line command

I—Insert Lines

The I (insert) line command inserts one or more lines in your data set or member. The inserted lines are blank unless you have defined a mask. See “MASK—Define Masks” on page 169 for more information about defining a mask.

Syntax



n

The number of blank lines to insert. If you do not type a number, or if the number you type is 1, only one line is inserted.

Description

To insert one or more lines in a data set or member:

1. Type I in the line command field of the line that the inserted line is to follow. If you want to insert more than one line, type a number greater than 1 after the I command.
2. Press Enter. The line or lines are inserted.

If you type any information, even a blank character in the inserted line, the line becomes part of the source data and is assigned a line number the next time you press Enter. However, if you do not type any information, the space for the new line is automatically deleted the next time you press Enter.

If you type information on the last, or only, inserted line and the cursor is still in the data portion of that line, the editor automatically inserts another line when you press Enter or a scroll function key, but only if the new inserted line remains on the panel. If the new line is at the bottom of the panel, the editor automatically scrolls down so that the new line is displayed at the bottom of the screen.

Examples

Figure 78 on page 163 shows how to insert lines in a member. To insert three lines, type I3 in the line command field.

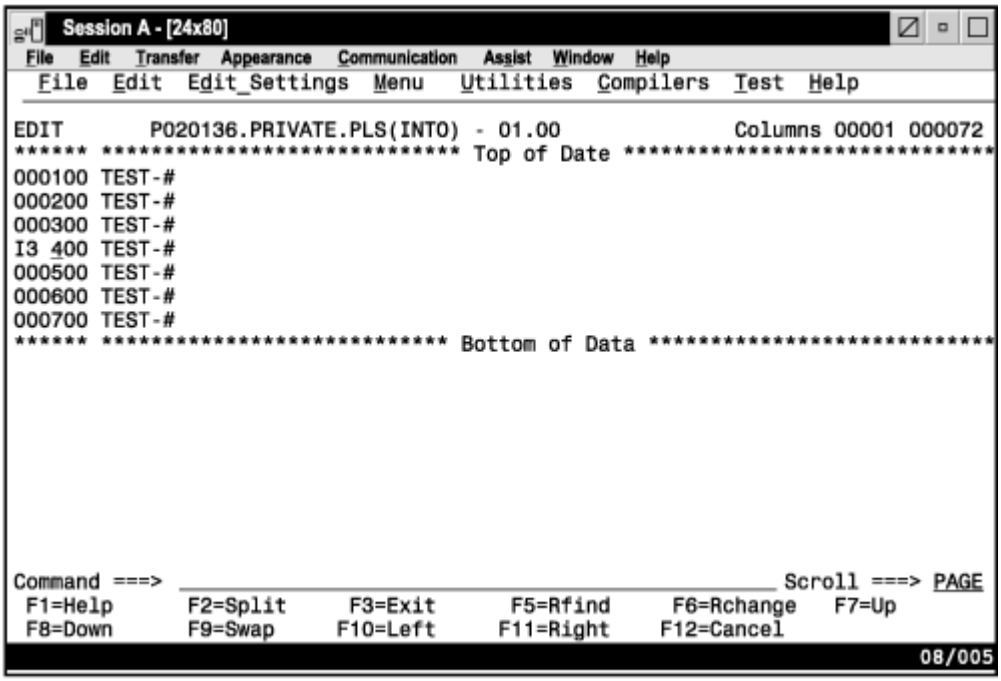


Figure 78. Before the I (Insert) line command

When you press Enter, the editor inserts three lines. See Figure 79 on page 163.

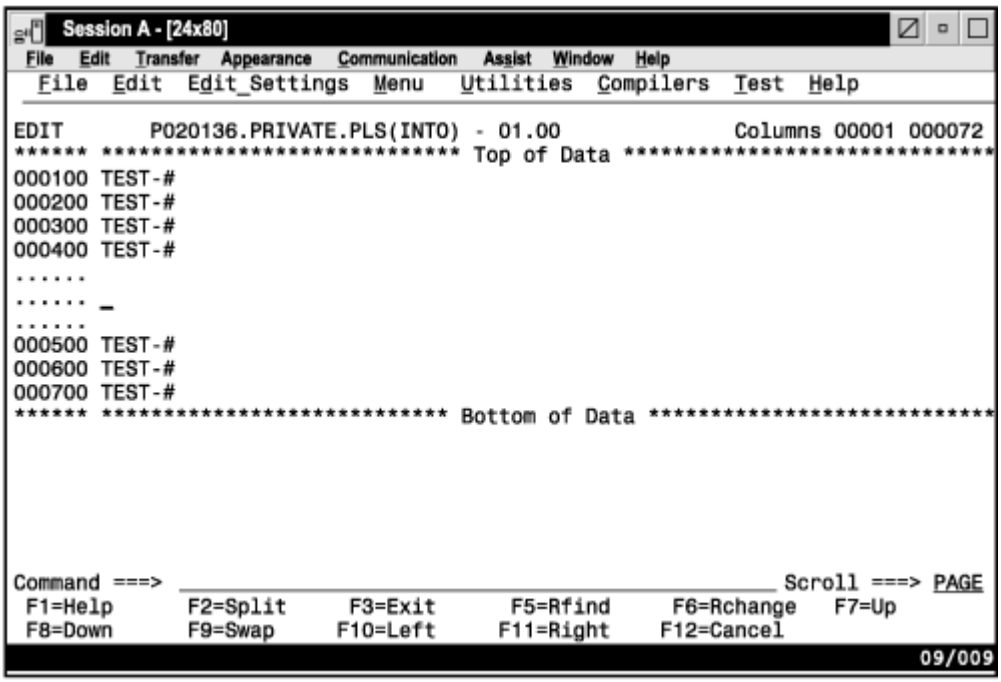
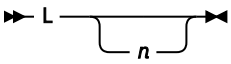


Figure 79. After the I (Insert) line command

L—Show the Last Line(s)

The L (show last line) line command redisplays one or more lines at the end of a block of excluded lines. See “Redisplaying excluded lines” on page 58 for more information about excluding lines.

Syntax



n

The number of lines to be redisplayed. If you do not type a number, or if the number you type is 1, only one line is redisplayed.

Description

To redisplay the last line or lines of a block of excluded lines:

1. Type L in the line command field next to the dashed line that shows where lines have been excluded. The message in the dashed line tells you how many lines are excluded. If you want to redisplay more than one line, type a number greater than 1 after the L command.
2. Press Enter. The last line or lines are redisplayed.

Examples

Figure 80 on page 164 shows how to redisplay the last three excluded lines. To redisplay the last three lines, type L3 in the line command field of the excluded lines.

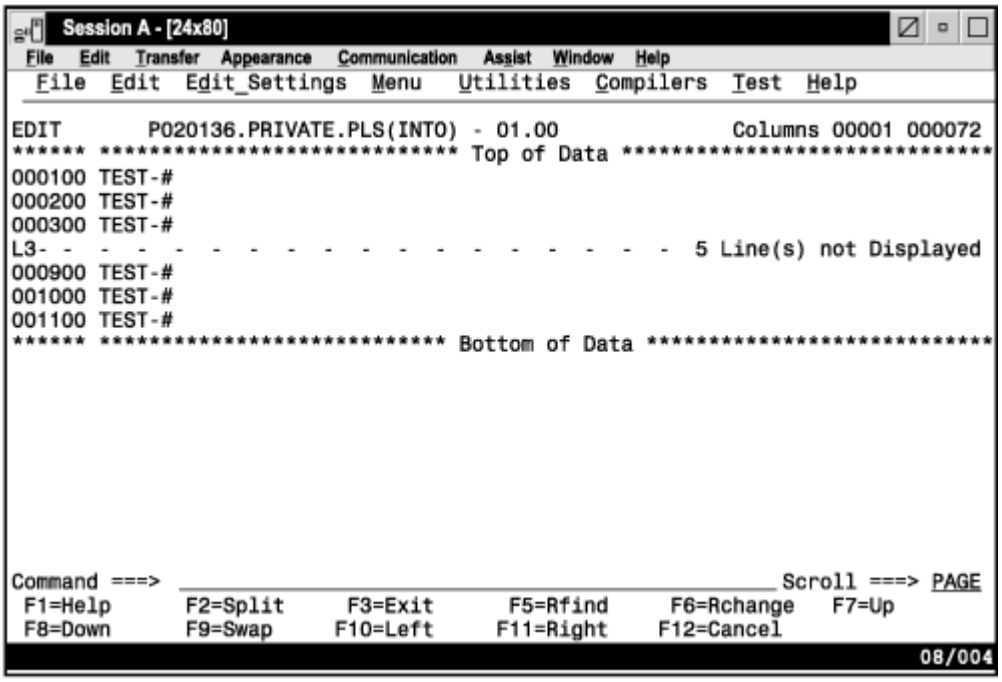


Figure 80. Before the L (Show Last Line) line command

When you press Enter, the editor redisplays the last three lines. See Figure 81 on page 165.

Note: Excluded lines do not need to be displayed again before saving the data. The excluded lines message line is never saved.

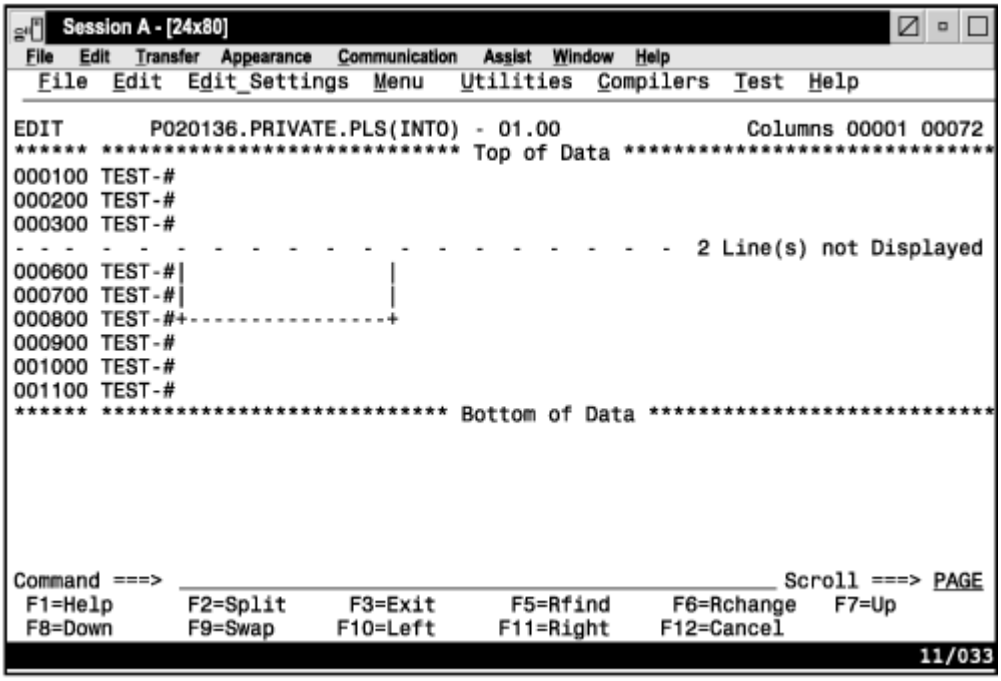
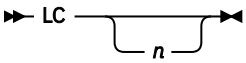


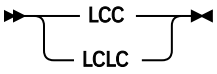
Figure 81. After the L (Show Last Line) line command

LC—Convert Characters to Lowercase

The LC (lowercase) line command converts characters in a data set or member from uppercase to lowercase. However, it does not affect the caps mode of the data that you are editing.

Syntax





n

The number of lines to be converted to lowercase. If you do not type a number, or if the number you type is 1, only the line on which you type LC is converted to lowercase.

Description

To convert characters on one or more lines to lowercase:

1. Type LC in the line command field of the source code line that contains the characters you want to convert. If you also want to convert characters on one or more lines that immediately follow this line, type a number greater than 1 after the LC command.
2. Press Enter. The characters on the source code lines are converted to lowercase.

To convert characters in a block of lines to lowercase:

1. Type LCC in the line command field of both the first and last source code lines that contain characters that are to be converted. You can scroll (or use FIND or LOCATE) between typing the first LCC and the second LCC, if necessary.
2. Press Enter. The characters in the source code lines that contain the two LCC commands and in all of the source code lines between them are converted to lowercase.

See the UC (uppercase) line command and the CAPS primary and macro commands, which are related, for information about converting characters from uppercase to lowercase and vice versa.

Examples

Figure 82 on page 166 shows how to use the LC command without any operands. To convert a line, type LC in the line command field of the line you want to convert.

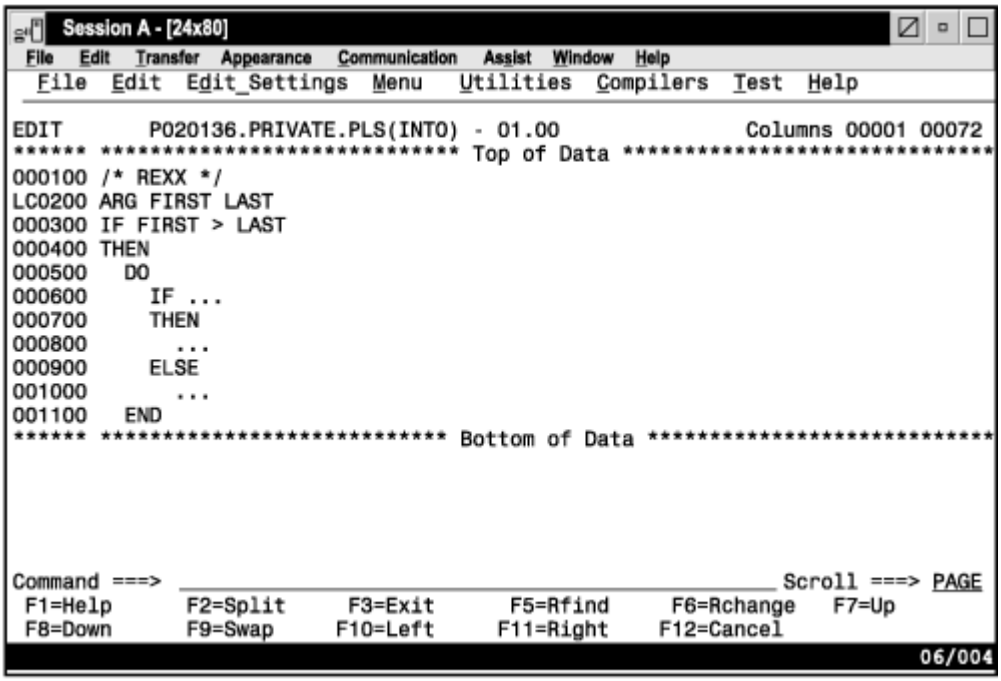


Figure 82. Before the LC (Lowercase) line command

When you press Enter, the editor converts the characters in the line to lowercase. See Figure 83 on page 167.

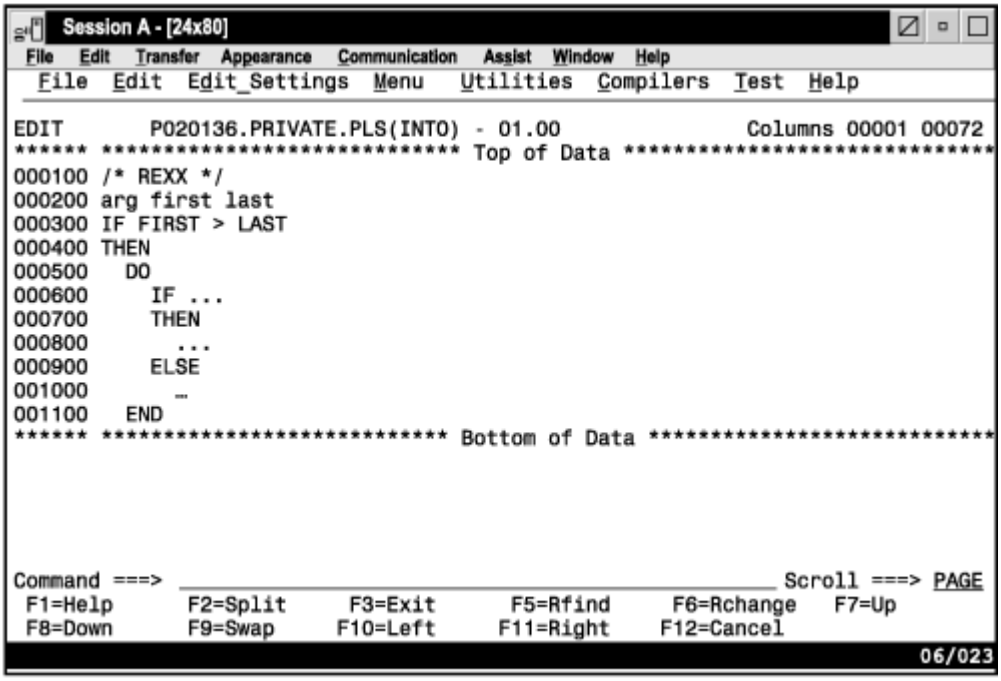
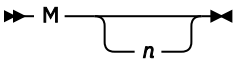


Figure 83. After the LC (Lowercase) line command

M—Move Lines

The M (move) line command moves lines from one location to another.

Syntax



n

The number of lines to be moved. If you do not type a number, or if the number you type is 1, only the line on which you type M is moved.

Description

To move one or more lines within the same data set or member:

1. Type M in the line command field of the line to be moved. If you want to move one or more lines that immediately follow this line, type a number greater than 1 after the M command.
2. Next, specify the destination of the line to be moved by using either the A (after), B (before), or O (overlay) line command. See the descriptions of those commands if you need more information about them.
3. Press Enter. The line or lines are moved to the new location.

To move a block of lines within the same data set or member:

1. Type MM in the line command field of both the first and last lines to be moved. You can scroll (or use FIND or LOCATE) between typing the first MM and the second MM, if necessary.

2. Use the A (after), B (before), or OO (overlay) command to show where the moved lines are to be placed. Notice that when you use the block form of the M command (MM) to move and overlay lines, you should also use the block form of the O command (OO).
3. Press Enter. The lines that contain the two MM commands and all of the lines between them are moved to the new location.

Note: Only blank characters in the lines specified with O or OO are overlaid with characters in the corresponding columns from the source lines. Characters that are not blank are not overlaid. The overlap affects only those characters within the current column boundaries.

To move lines to another data set or member:

1. Type either CREATE or REPLACE on the command line.
2. Use one of the forms of the M command described previously.
3. Press Enter.
4. On the next panel, type the name of the data set or member that you want to create or replace.
5. Press Enter. The lines are moved to the data set or member that you specified.

Note: To move lines into an existing data set or member without replacing that data set or member, use the MOVE primary or macro command.

Examples

Figure 84 on page 168 shows how you can move data by using the M with the A (After) line command. To move a line, type M in the line command field of the line you want to move. Type a A in the line command field of the line you want the moved line to follow.

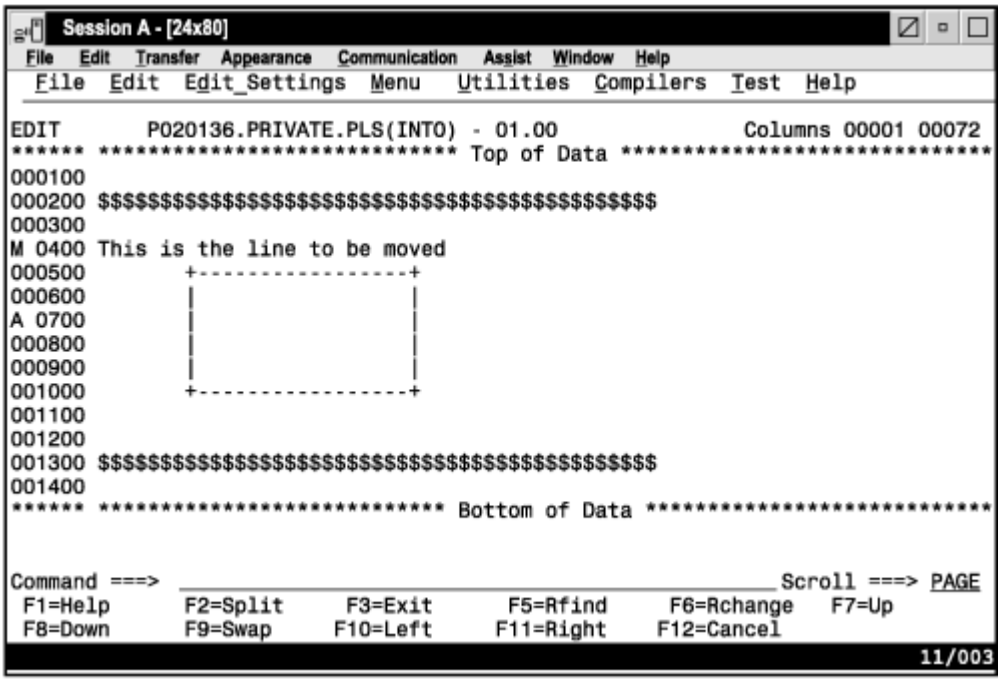


Figure 84. Before the M (Move) line command

When you press Enter, the editor moves the line where you typed the M command to a position immediately after the line where you typed the A command, as shown in Figure 85 on page 169. If you press Enter before specifying a destination, the editor displays a `MOVE/COPY pending` message at the top of the panel. The line is not moved until you specify a destination.

Note: While in MOVE/COPY pending mode, you can issue FIND and LOCATE primary commands, but you cannot use a CHANGE command to change data until after the copy completes.

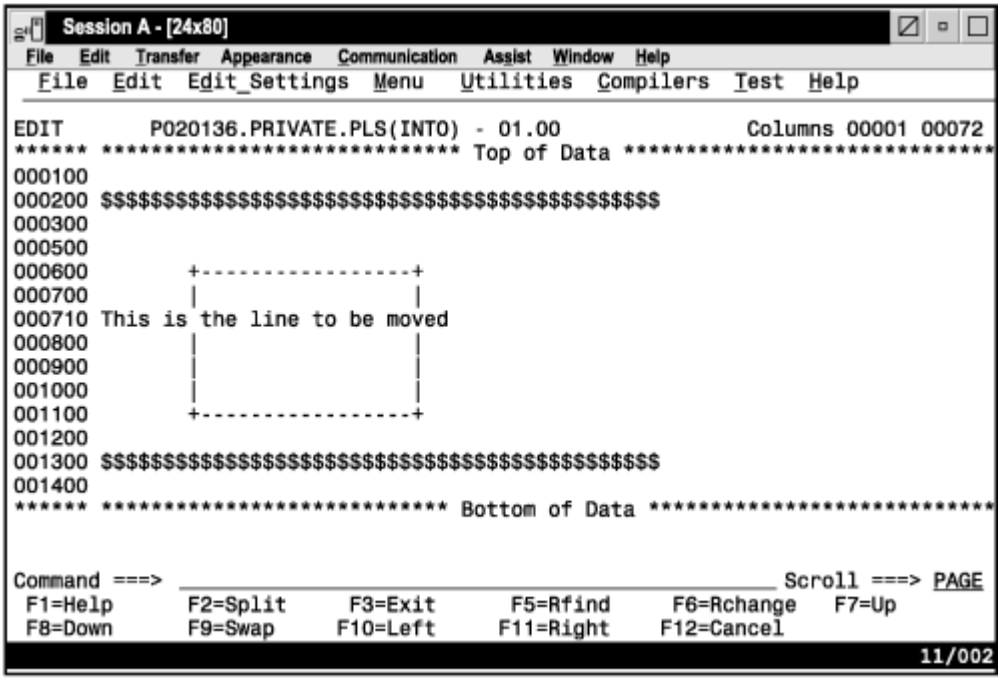


Figure 85. After the M (MOVE) line command

MASK—Define Masks

The MASK line command displays the =MASK> line. On this line, you can type characters that you want to insert into an unformatted data set or member. These characters, which are called the *mask*, are inserted whenever you use the I (insert), TE (text entry), or TS (text split) line commands, or when you edit an empty data set.

Syntax

Description

To display the =MASK> line:

1. Type MASK in the line command field of any line.
2. Press Enter. The =MASK> line is displayed.

Initially, the mask contains all blanks. To define a mask:

1. Add characters to or delete characters from the =MASK> line while it is displayed.
2. Press Enter. The mask is now defined.

Once a mask is defined, the contents of the =MASK> line are displayed whenever a new line is inserted. This occurs when you use the I (insert), TE (text entry), and TS (text split) line commands, and when you edit an empty data set. You can change the mask definition whenever you need to by repeating the preceding steps.

To remove the =MASK> line from the panel, perform one of these actions:

- Type D in the line command field that contains the =MASK> flag and press Enter.

- Type RESET on the command line and press Enter.
- End the edit session by:
  - Pressing F3 (if it is defined as the END command), or
  - Typing END on the command line and pressing Enter

The mask line is never saved as part of the data. However, the mask remains in effect, even if it is not displayed, until you change it. The contents of the mask are retained in the current edit profile, and are automatically used the next time you edit the same kind of data.

The MASK command is ignored in *formatted edit mode*. You enter formatted edit mode when you type the name of a previously defined format in the **Format Name** field on the Edit Entry panel when beginning an edit session. If you have defined a mask before entering formatted edit mode, the mask is not retained in the current edit profile.

Examples

In Figure 86 on page 170, the mask is displayed and the characters /* and */ are typed on the mask line.

```
Session A - [24x80]
File  Edit  Transfer  Appearance  Communication  Assist  Window  Help
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help

EDIT       P020136.PRIVATE.PLS(INTO) - 01.00              Columns 00001 00072
****** ***************************** Top of Data ******************************
000100 /* REXX */
=COLS> ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
000200 ARG FIRST LAST                          /* SET ARGUMENTS            */
000300 IF FIRST > LAST                         /*   IF 'FIRST' IS GREATER  */
000400 THEN                                    /* THAN 'LAST',             */
MASK   _ DO                                    /*      AND                 */
000600     IF TEMP = FIRST                     /*   IF 'TEMP' IS EQUAL     */
000700     THEN                                /*   TO 'FIRST', THEN       */
000800       FIRST = LAST                      /*     SET FIRST EQUAL      */
000900     ELSE                                /*     TO 'LAST', OTHERWISE */
001000       LAST = TEMP                       /*     SET 'LAST' EQUAL     */
001100   END                                   /*     TO TEMP              */
001200 END                                     /*                          */
****** ***************************** Bottom of Data ***************************



Command ===> _____________________________________________ Scroll ===> PAGE
 F1=Help      F2=Split     F3=Exit      F5=Rfind     F6=Rchange   F7=Up
 F8=Down      F9=Swap     F10=Left     F11=Right    F12=Cancel
                                                              10/009
```

Figure 86. Before the MASK line command

When you insert five lines, the new lines contain the contents of the mask. See Figure 87 on page 171.

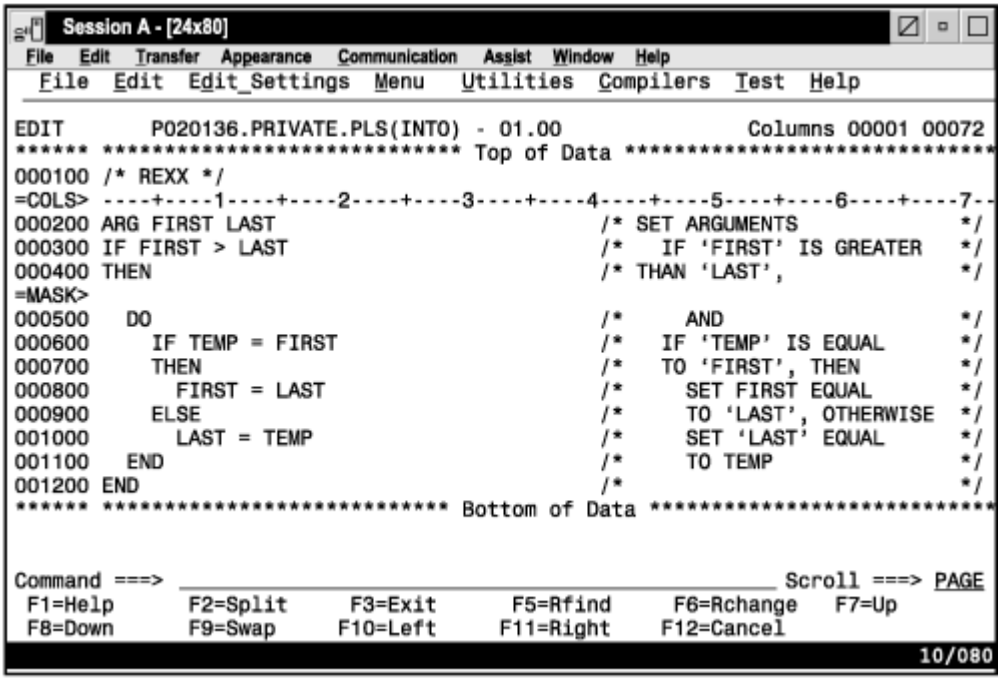
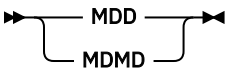


Figure 87. After the MASK line command

MD—Make Dataline

The MD (make dataline) line command converts one or more ==MSG>, =NOTE=, =COLS>, or ====== (information) lines to data so they can be saved as part of your data set.

Syntax



n

The number of lines to be converted to data. If you do not type a number, or if the number you type is 1, only the line on which you type MD is converted.

Description

If you enter the MD line command on:

- Any line except a ==MSG>, =NOTE=, =COLS>, or ====== line, it is ignored.
- The TOP OF DATA and BOTTOM OF DATA lines, it is not allowed.
- An excluded line, any converted lines remain excluded and are converted.
- A line that contains a label, the label remains after the line is converted.

Note: The MD line command only works on the editable =COLS> lines produced by the COLS line command. It does not work with the non-editable =COLS> indicator line produced by the COLS primary command.

For best results, you should set your edit profile to NUMBER OFF and make sure that the record length of your data set or member is at least 80 before entering the MD line command. Otherwise, data on the right may be truncated.

To convert one or more lines to data:

1. Type MD in the line command field next to the line that is to be converted. If you also want to convert one or more lines that immediately follow this line, type a number greater than 1 after the MD command.
2. Press Enter. The lines are converted to data.

To convert a block of lines to data:

1. Type MDD in the line command field of both the first and last lines to be converted. You can scroll (or use the FIND or LOCATE command) between typing the first MDD and the second MDD, if necessary.
2. Press Enter. The lines that contain the two MDD commands and all eligible lines between them are converted to data.

Examples

Figure 88 on page 172 shows how you can convert a block of temporary lines to data by using the block form of the MD line command. Type MDD over the =NOTE= line flags in the line command field of the first and last lines of the block of lines that you want to convert to data.

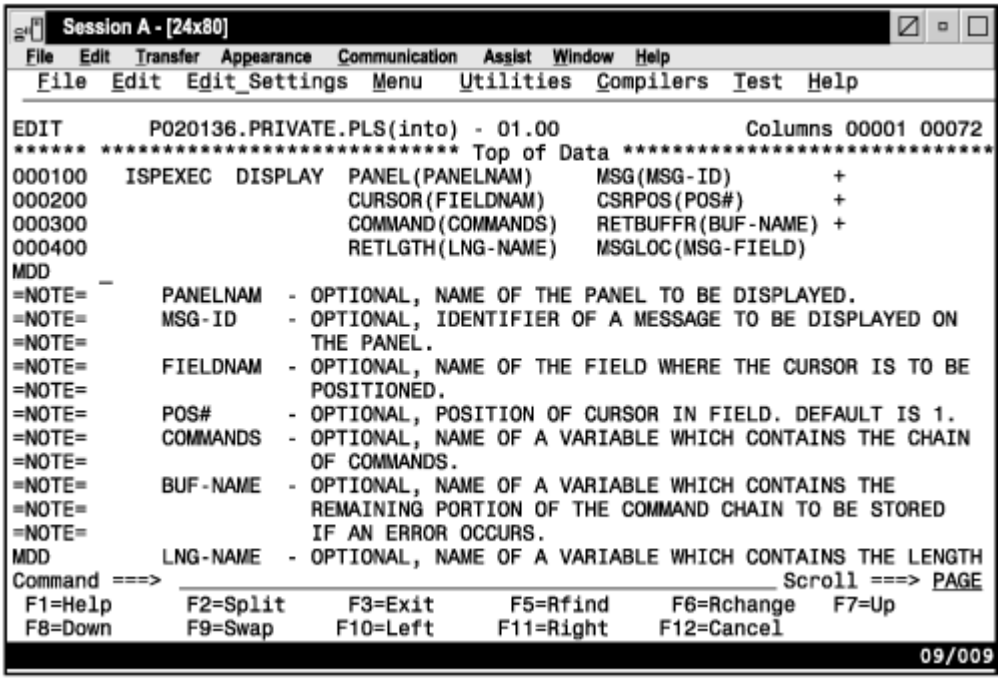


Figure 88. Before the MD (Make Dataline) line command

When you press Enter, the lines on which the MDD commands are typed and all of the lines between them are converted to data. See Figure 89 on page 173.

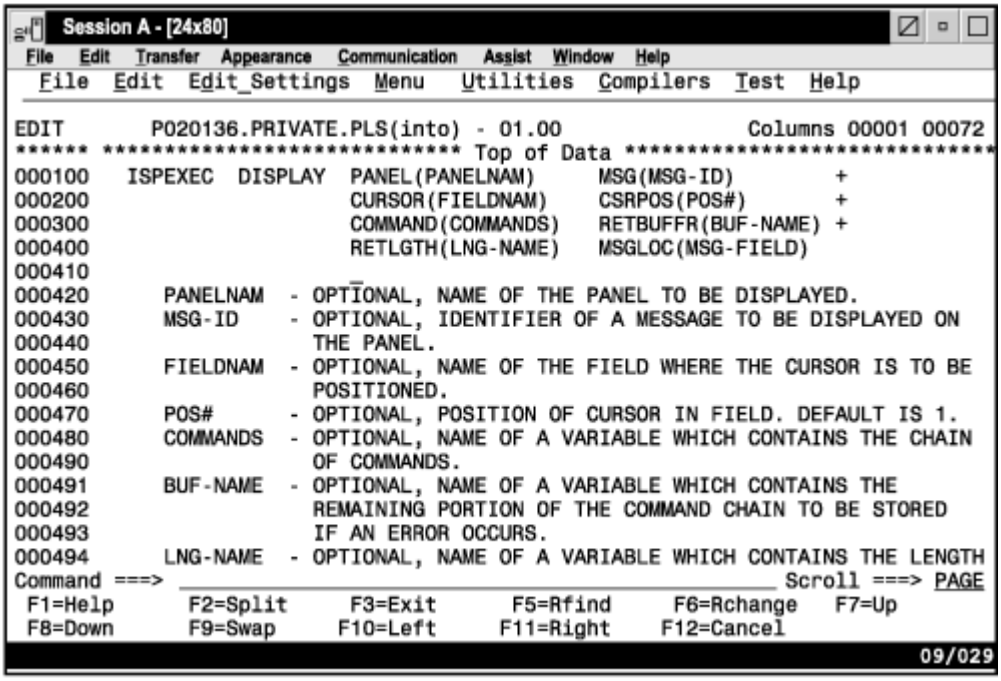


Figure 89. After the MD (Make Dataline) line command

O, OK—Overlay Lines

When data is to be copied or moved by the C (copy) or M (move) line commands and overlaid on one or more existing lines of data, the O (overlay) line command specifies the destination for the data.

If there are multiple destinations for the data, the OK (overlay, intermediate target) line command specifies each intermediate destination for the data. You specify the final destination for the data with either the O (overlay), A (after), or B (before) line commands. The final destination must be after the intermediate destinations in the file. For more information about the A and B line commands, see:

"A, AK—Specify an After destination" on page 146
"B, BK—Specify a Before destination" on page 149

The data that is copied or moved overlays blanks in the destination lines of data. This allows you to rearrange a single-column list of items into multiple column, or tabular, format.

When data is to be moved or copied and then overlaid on a *single* destination:

- Where the destination is a *single line*:
  - The O (overlay) line command specifies the destination for the data.

    You can type a number after the O line command to specify the number of times that the M or C line command is to be performed. For example, typing the command O3 against a line causes the data to be moved or copied and then overlaid on that line and also the next two lines.
- Where the destination is a *block of lines*:
  - The OO (overlay, multiple-line target) line command specifies the first and last line of the destination for the data.

When data is to be moved or copied and then overlaid on *multiple* destinations:

- Where each destination is a *single line*:

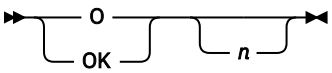
  - The OK (overlay, intermediate target) line command specifies each intermediate destination (but not the final destination) for the data.

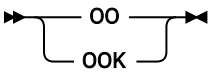
    You can type a number after the OK line command to specify the number of times that the M or C line command is to be performed. For example, typing the command OK3 against a line causes the data to be moved or copied and then overlaid on that line and also the next two lines.
  - The O (overlay) line command specifies the final destination for the data.

    You can type a number after the O line command as previously described.
- Where each destination is a *block of lines*:
  - The OOK (overlay, intermediate multiple-line target) line command specifies the first and last line of each intermediate destination (but not the final destination) for the data.
  - The OO (overlay, multiple-line target) line command specifies the first and last line of the final destination for the data.

Note: The OK and OOK line commands indicate that another destination of the form A, B, or O is still required proceeding forward through the file before the data is moved or copied to the multiple destinations specified.

Syntax





n

The number of lines to be overlaid. If you do not type a number, or if the number you type is 1, only one line is overlaid.

Description

To overlay one or more *single* lines:

1. Type either M or C in the line command field of the line that is to be moved or copied.
2. To specify a *single* destination for the data that is to be moved or copied, type O in the line command field of the line that the moved or copied line is to overlay.

   To specify *multiple* destinations for the data that is to be moved or copied:

   a. Type OK in the line command field of each intermediate destination line (but not the final destination line) that the moved or copied data is to overlay.

      To overlay data in the lines following an intermediate destination line, type a number after the OK line command to specify the number of times that the M or C line command is to be performed.

   b. Type O in the line command field of the final destination line that the moved or copied data is to overlay. The final destination line must come after all the intermediate destination lines.

      To overlay data in the lines following the final destination line, type a number after the O line command to specify the number of times that the M or C line command is to be performed.
3. Press Enter. The data being moved or copied overlays the specified line or lines.

To overlay one or more *blocks of lines*:

1. Type either MM or CC in the line command field of the first and last lines of a block of lines that is to be moved or copied. You can scroll (or use FIND or LOCATE) between typing the first command and the second command, if necessary.

2. To specify a *single* destination for the data that is to be moved or copied, type 00 in the line command field of the first and last lines that the block of lines being moved or copied is to overlay. Again, you can scroll (or use FIND or LOCATE) between typing the first 00 and the second 00, if necessary.

   To specify *multiple* destinations for the data that is to be moved or copied:

   a. Type 00K in the line command field of the first and last lines of each intermediate destination (but not the final destination) that the block of lines being moved or copied is to overlay.
   b. Type 00 in the line command field of the first and last lines of the final destination that the moved or copied data is to overlay. The lines of the final destination must come after all those of the intermediate lines.
3. Press Enter. The lines that contain the two CC or MM commands and all of the lines between them overlay the lines that contain:
   - Each pair of OOK commands and all of the lines between them.
   - The two OO commands and all of the lines between them.

Only blank characters in the lines specified with O or OO (or OK or OOK) are overlaid with characters in the corresponding columns from the source lines. Characters that are not blank are not overlaid. The overlap affects only those characters within the current column boundaries.

The number of source and receiving lines need not be the same. If there are more receiving lines, the source lines are repeated until the receiving lines are gone. If there are more source lines than receiving lines, the extra source lines are ignored. The overlay operation involves only data lines. Special lines such as MASK, TABS, BNDS, and COLS are ignored as either source or receiving lines.

Note: There is no special support for DBCS data handling. You are responsible for DBCS data integrity when overlaying lines.

Examples

Figure 90 on page 176 illustrates the O (overlay) line command. Suppose you were editing a list in a single left-adjusted column and wanted to place portions of the list side-by-side. First, using the) (column shift right) command, shift a portion of the list the appropriate amount to the right to overlay in a multiple column format. Next, type MM in the line command field to mark the beginning and end of the block of lines you want to move, then type 00 in the line command field to mark the destination of the lines you want to overlay.

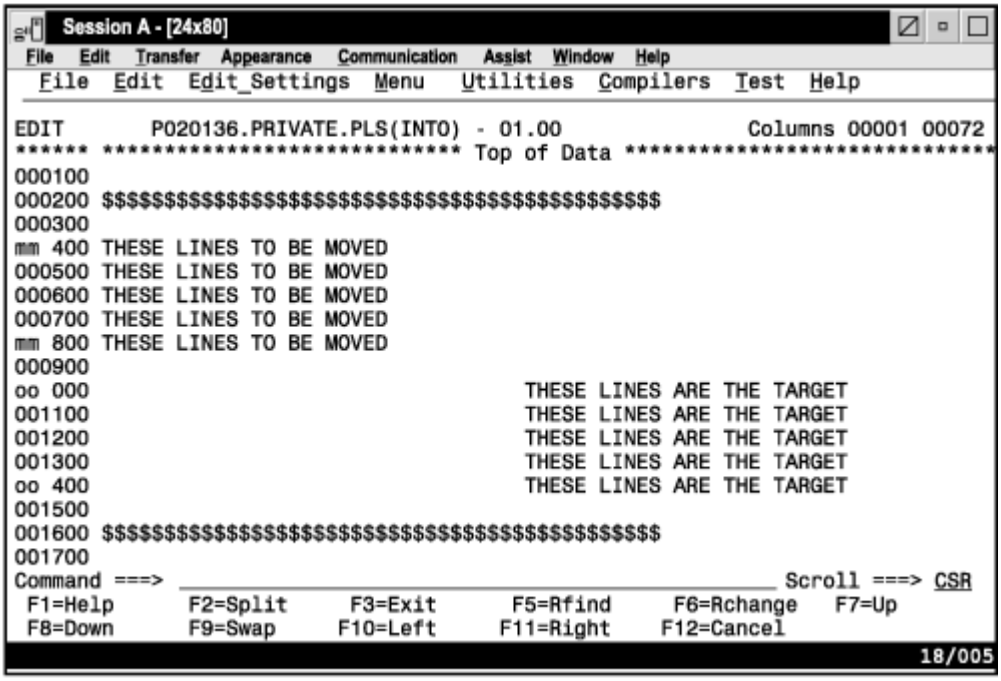


Figure 90. Before the O (Overlay) line command

When you press Enter, the editor overlays the lines you marked to move on the destination block. See Figure 91 on page 176.

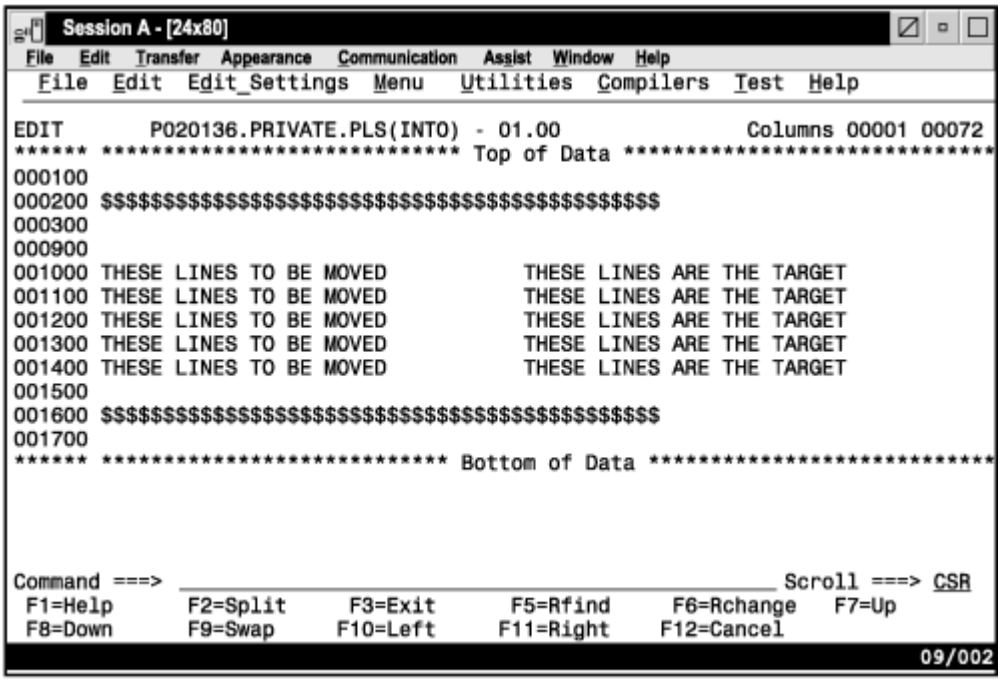
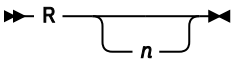


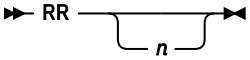
Figure 91. After the O (Overlay) line command

R—Repeat Lines

The R (repeat) line command repeats one or more lines in your data set or member immediately after the line on which the R command is entered.

Syntax





n

The number of lines to be repeated. If you do not type a number, or the number you type is 1, only the line on which you type R is repeated.

Description

To repeat one or more lines:

1. Type R in the line command field of the line that is to be repeated. If you want to repeat the line more than once, type a number that is greater than 1 immediately after the R command.
2. Press Enter. The editor inserts a duplicate copy or copies of the line immediately after the line that contains the R command.

To repeat a block of lines:

1. Type RR in the line command field of both the first and last lines to be repeated. You can scroll (or use FIND or LOCATE) between typing the first RR and the second RR, if necessary.
2. Press Enter. The lines that contain the two RR commands and all of the lines between them are repeated immediately after the line that contains the second RR command.

Examples

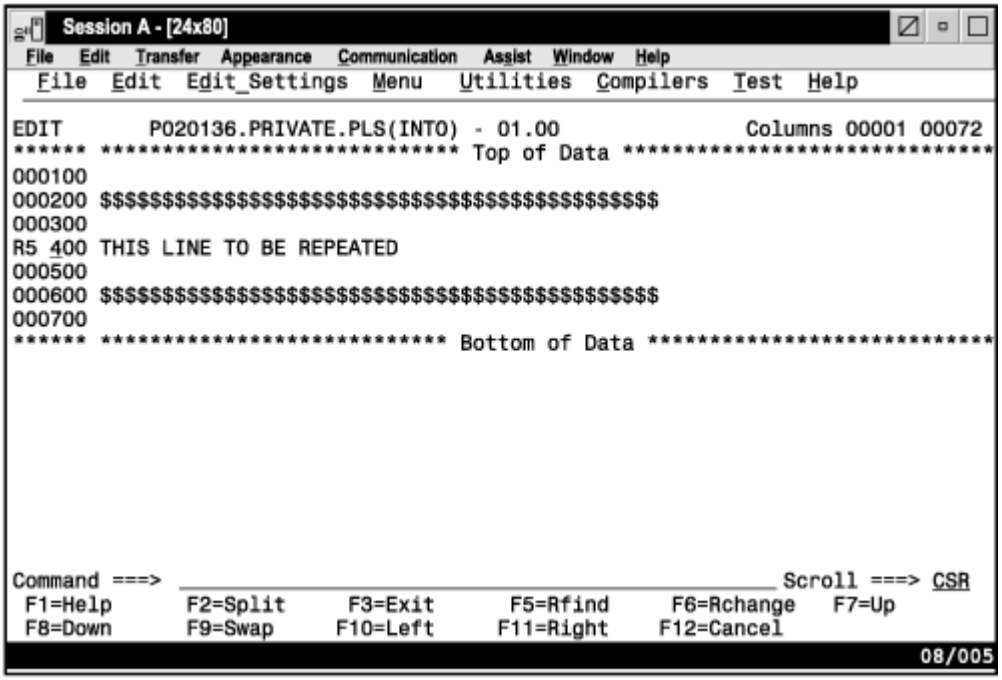


Figure 92. Before the R (repeat) line command

When you press Enter, the editor repeats line 000400 five times. See Figure 93 on page 178.

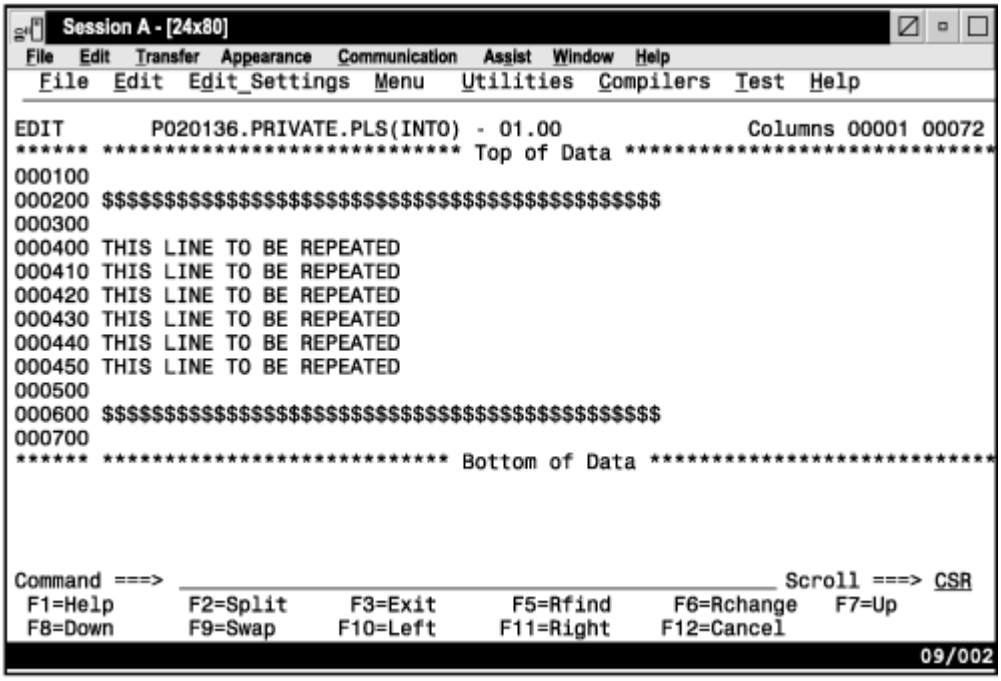
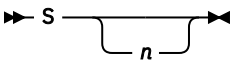


Figure 93. After the R (Repeat) line command

S—Show Lines

The S (show line) line command causes one or more lines in a block of excluded lines to be redisplayed. The redisplayed lines have the leftmost indentation levels; they contain the fewest leading blanks. See “Redisplaying excluded lines” on page 58 for more information about redisplaying excluding lines.

Syntax



n

The number of lines to be redisplayed. If there are more than 2 excluded lines, and you do not type a number or if the number you type is 1, only one line is redisplayed.

Note: If you enter an S line command to display all but one line of an excluded block, then that line is also displayed. This could result in more lines being displayed than the number you requested. For example, if five lines are excluded in a block, an S4 command causes all five lines to be displayed.

Description

To redisplay a line or lines of a block of excluded lines:

1. Type S in the line command field next to the dashed line that shows where a line or lines has been excluded. The message in the dashed line tells you how many lines are excluded.

   If you want to redisplay more than one line, type a number greater than 1 after the S command. If you type S3, for example, the three lines with the leftmost indentation level are displayed again. If more than three lines exist at this indentation level, only the first three are displayed.
2. Press Enter. The line or lines with the fewest leading blanks are redisplayed.

Examples

Figure 94 on page 179 shows how to redisplay a member's excluded lines. To redisplay four lines, type S4 in the line command field.

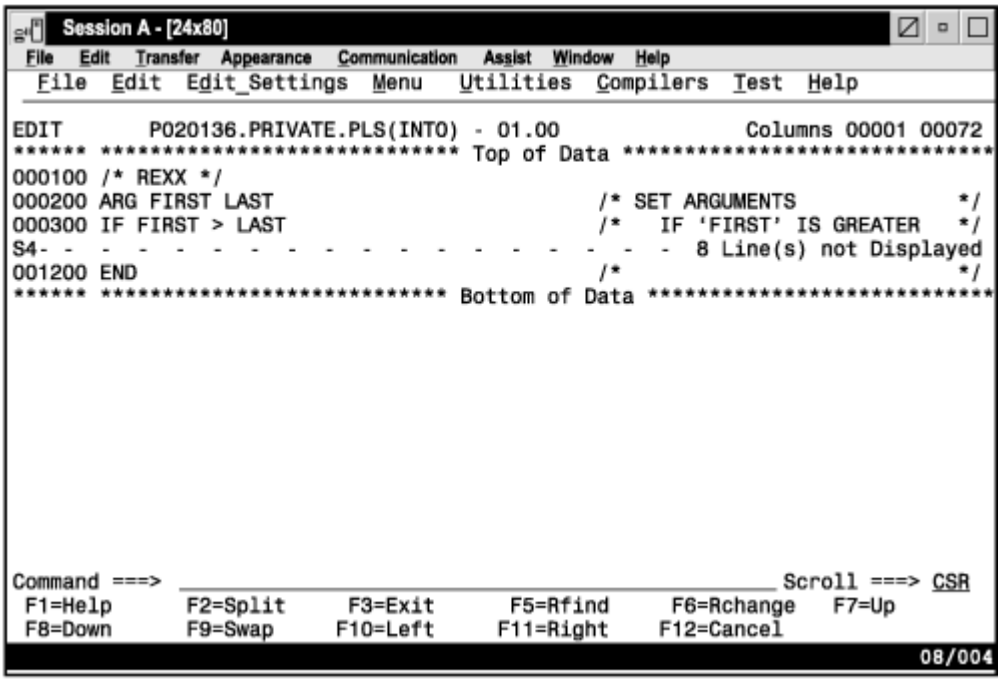


Figure 94. Before the S (Show) line command

When you press Enter, the four lines are redisplayed. See Figure 95 on page 179.

Note: Excluded lines do not need to be displayed again before saving the data. The excluded lines message line is never saved.

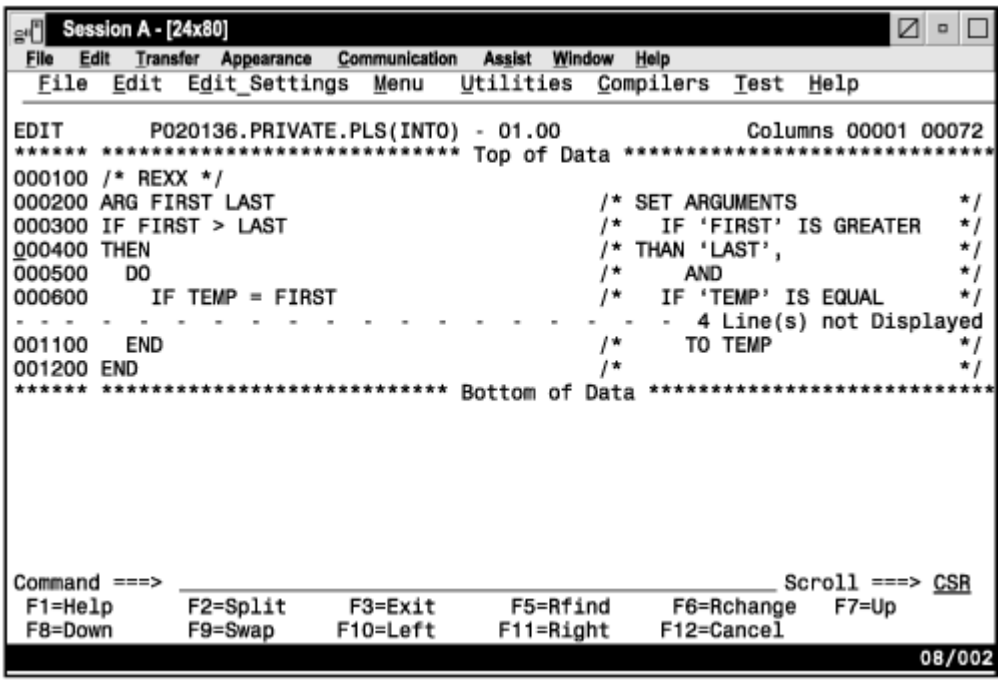


Figure 95. After the S (Show) line command

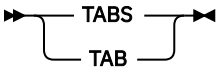
TABS—Control Tabs

The TABS line command:

- Displays the =TABS> (tab-definition) line
- Defines tab positions for software, hardware, and logical tabs

Use PROFILE to check the setting of tabs mode and the logical tab character. See “Using tabs” on page 63 if you need more information about using tabs.

Syntax



Description

When you type TABS in the line command field, =TABS> is displayed along with any previously defined tab positions. To remove the =TABS> line, use the D (delete) line command or the RESET primary command, or end the edit session. The =TABS> line is never saved as part of the data.

The tab definitions remain in effect, even if they are not displayed, until you change them. Tab definitions are retained in the current edit profile, and are automatically used the next time you edit the same kind of data.

Using software and hardware tabs

Edit a data set, type TABS ALL on the command line, and press Enter:

```
Command ===> TABS ALL
```

Now, type COLS in the line command field and press Enter again. A partial =COLS> line with positions 9 through 45 is shown in this example:

```
=COLS> -1----+----2----+----3----+----4----+
```

Next use the TABS line command to define software and hardware tabs. Type TABS in the line command field beneath the =COLS> line and press Enter.

When the =TABS> line appears, type hyphens in columns 15, 25, and 35, and asterisks in columns 20, 30, and 40, using the =COLS> line to find these columns:

```
=COLS> -1----+----2----+----3----+----4----+
=TABS>       -    *    -    *    -    *
```

With the preceding =TABS> line, you can move the cursor to a software tab position (hyphen) by pressing Enter, even if another character already occupies that position. To move the cursor to a hardware tab position (one space to the right of an asterisk), press either the Tab Forward or Tab Backward key. See Figure 96 on page 181.

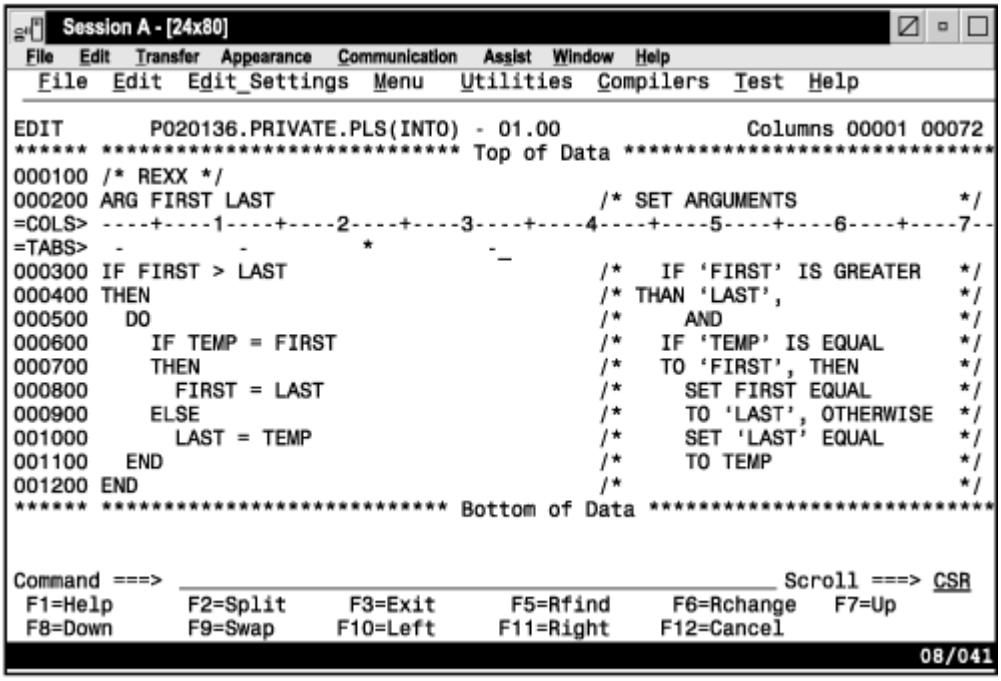


Figure 96. TAB line command example

Using software tab fields

You can define a *software tab field* by typing underscores or hyphens in two or more consecutive columns. This moves the cursor to the first nonblank character in the field. If the field contains all blanks, the cursor moves to the beginning of the field.

Using the example in the preceding section, create a software tab field by typing hyphens in columns 10 through 14. Then type some data inside the field and at each of the other tab positions, but below the =TABS> line:

```
=COLS> -1----+----2----+----3----+----4----+
=TABS>  ------    *    -    *    -    *
          123          456       789_
```

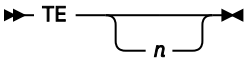
Notice in the preceding example that the cursor is positioned to the right of data string 789. With the cursor in this position, press Enter. The cursor moves under the 1 in the 123 data string, not to column 10, which is the beginning of the field.

TE—Text Entry

The TE (text entry) line command provides one very long line wrapped around many lines of the display to allow power typing for text entry. The editor does the formatting for you.

The TE line command is different from the I (insert) line command. The I command inserts a specified number of separate, blank lines as well as the mask, if there is one, as you typed it. With the TE command, the input data is formatted, only mask line characters outside the current boundaries are added to the formatted lines.

Syntax



n
The number of blank lines to be added. If you do not type a number, the display is filled with blanks from the line following the TE to the bottom of the screen.

Description

Before you enter text entry mode:

- If you are going to be typing text in paragraph form, make sure caps mode is off. Otherwise, when you press Enter, your text changes to uppercase.
- You may want to turn off number mode to prevent sequence numbers from writing over any of your text.
- Make sure the bounds setting is where you want it so that the text will flow correctly when you end text entry mode.

To enter text entry mode:

1. Type TE in the line command field. If you want to specify several blank lines, type a number greater than 1 immediately after the TE command. If the number that you type is greater than the number of lines remaining on the display, the vertical bar that shows where you will run out of room is not displayed and the keyboard does not lock at the last character position on the display. You can scroll down to bring the additional blank text entry space into view.
2. Press Enter. The editor inserts a single continuous blank area for the specified number of lines or to the bottom of the display.

To begin a new paragraph:

1. Use the return (Enter), cursor movement, or Tab keys to advance the cursor enough spaces to leave one blank line on the display.

If there are insufficient blank spaces on the display, the keyboard locks when you try to type beyond the last character position. A vertical bar (|) is displayed above the cursor at the locked position.

To generate more blank spaces:

1. Press the Reset key to unlock the keyboard.
2. Press Enter.

To end text entry mode:

1. Press Enter. The data is flowed together into a paragraph and any embedded blanks are preserved. The left and right sides of the paragraph are determined by the current bounds.

See “Word processing” on page 61 and “Entering text (power typing)” on page 63 if you need more information.

Examples

Figure 97 on page 183 shows how the TE (text entry) command allows you to use power typing and word wrap to input text. The edit profile is set to `NUMBER OFF` and `CAPS OFF`. Also, the left bound is set to 1 and the right bound is set to 72. A new data set member called CHAP10 has been started and the TE command is typed in the line command field.

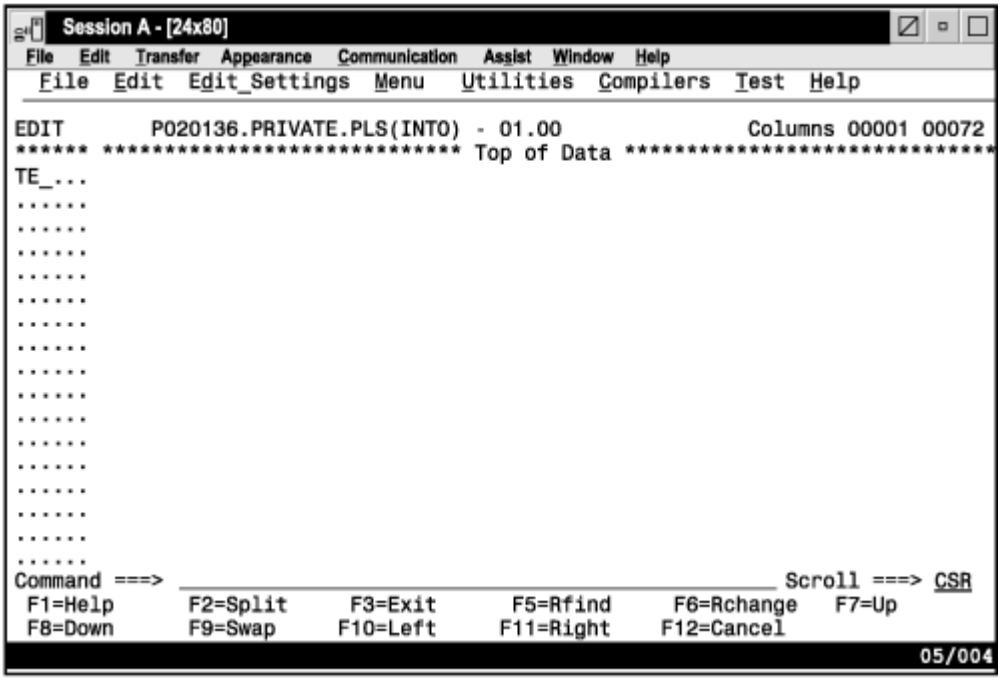


Figure 97. Before the TE (Text Entry) line command

When you press Enter, the editor begins text entry mode. The cursor shows where text input begins and the vertical bar in the lower-right corner of the panel shows how much room you have to work with. See Figure 98 on page 183.

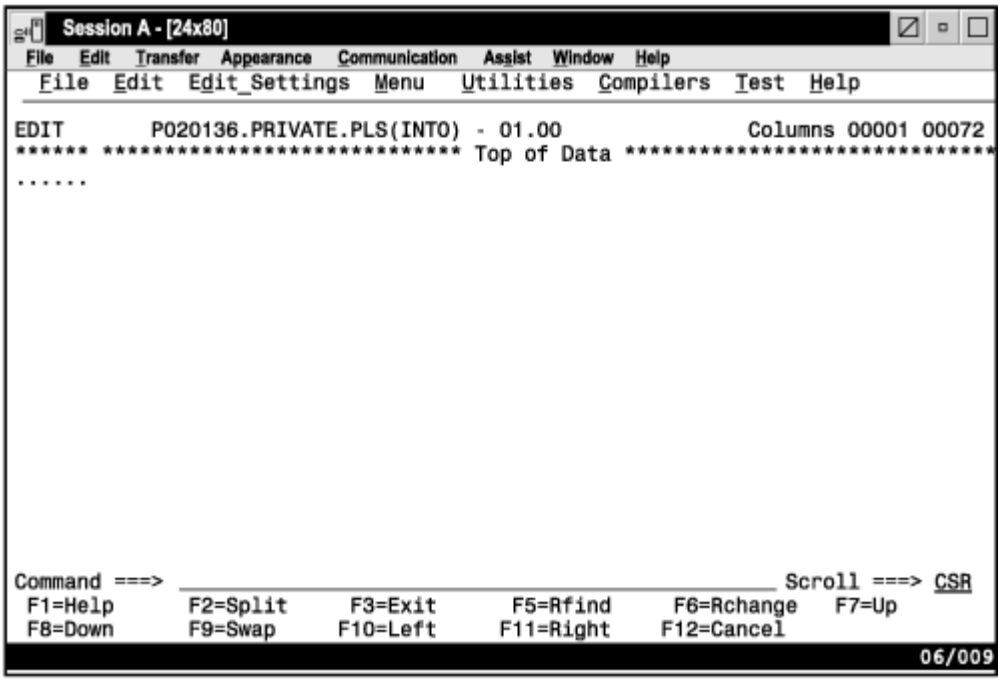


Figure 98. After the TE (Text Entry) line command

When you enter text, some of the words are split between lines, with part of the word at the right end of a line and the remainder of the word at the beginning of the next line. See Figure 99 on page 184.

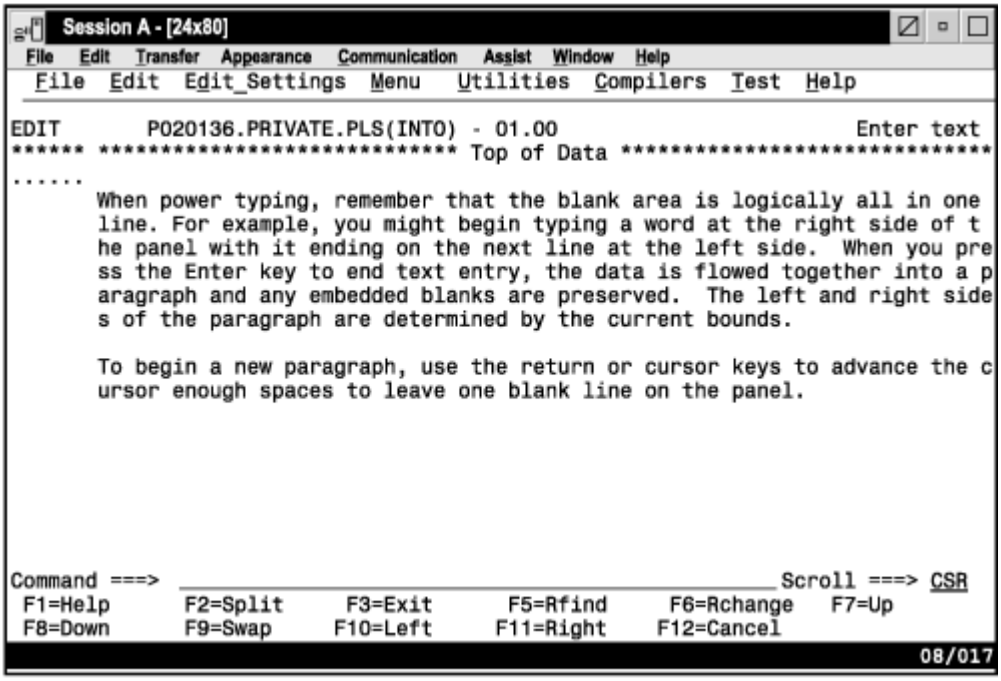


Figure 99. Sample text during text entry mode

When you press Enter, the editor exits text entry mode. As shown in Figure 100 on page 184, the text flows between the bounds settings and the line numbers are displayed in the line command field.

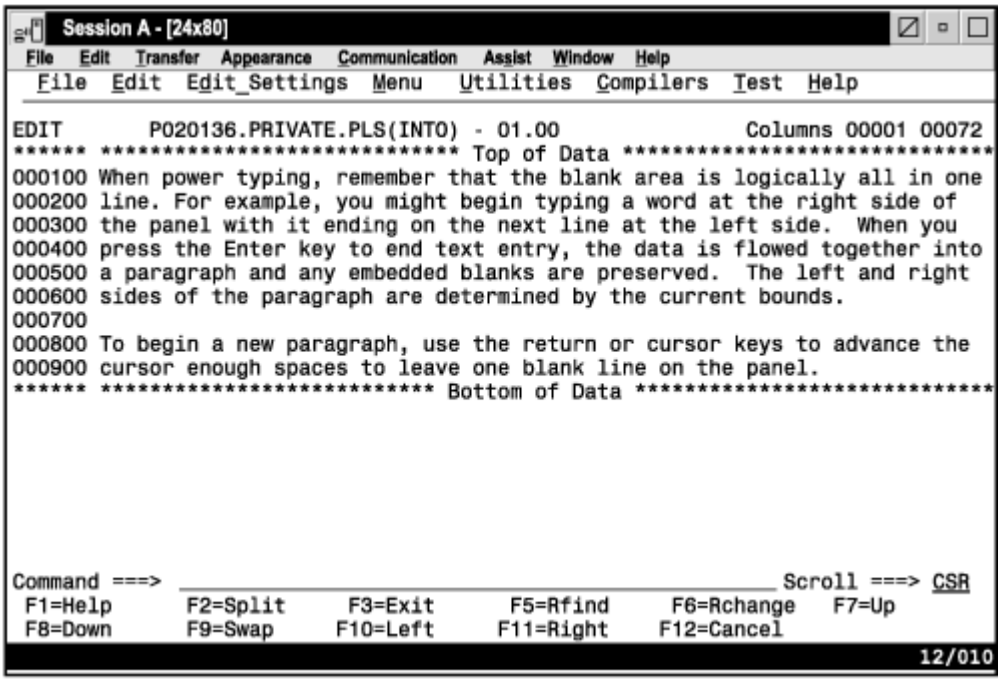


Figure 100. Sample text after text entry mode

TF—Text Flow

The TF (text flow) line command restructures paragraphs. This is sometimes necessary after deletions, insertions, or splitting.

Syntax

n

The column number to which the text should be flowed. The default is the panel width when default boundaries are in effect. If you are using nondefault bounds, the right boundary is used. This is different from the TFLOW macro command, which always defaults to the right boundary.

If a number greater than the right boundary is specified, the right boundary is used.

Description

To flow text:

1. Type TF in the line command field of the line at which you want the text to begin flowing. If you want to specify the rightmost column position for the restructured text, type a number greater than 1 immediately after the TF command.
2. Press Enter. The text is flowed from the beginning of that line to the end of the paragraph.

See “Word processing” on page 61 and “Formatting paragraphs” on page 61 for more information.

Examples

Figure 101 on page 185 demonstrates text restructuring. The bounds are set at columns 1 and 72. A TF50 command is typed on line 000041.

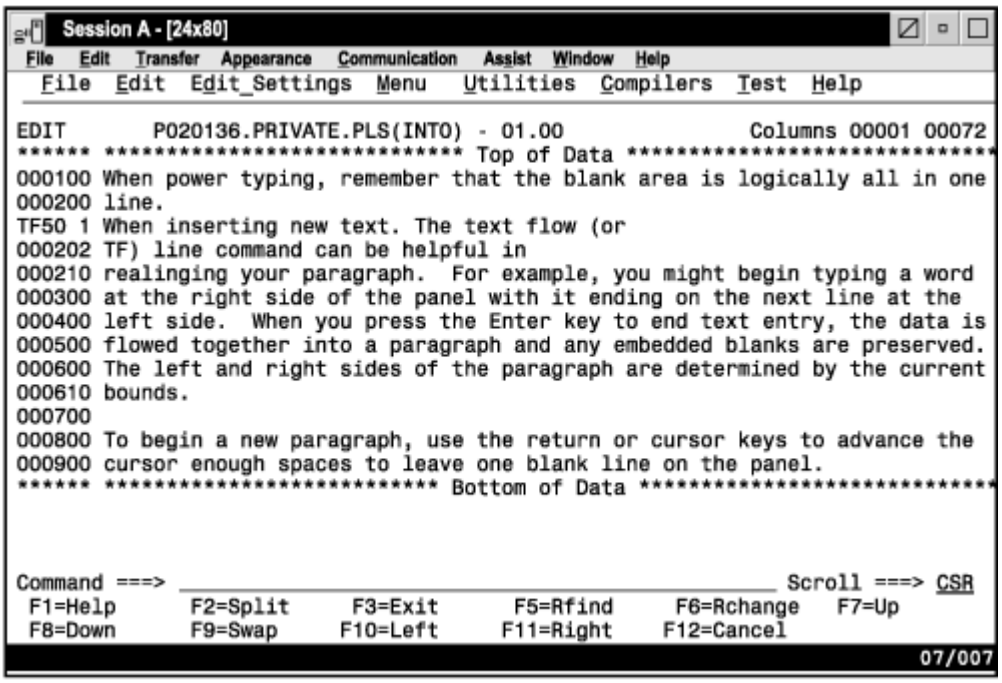


Figure 101. Before the TF (Text Flow) line command

When you press Enter, the editor takes all text in that paragraph between columns 1 and 72 and reformats it between columns 1 and 50. See Figure 102 on page 186.

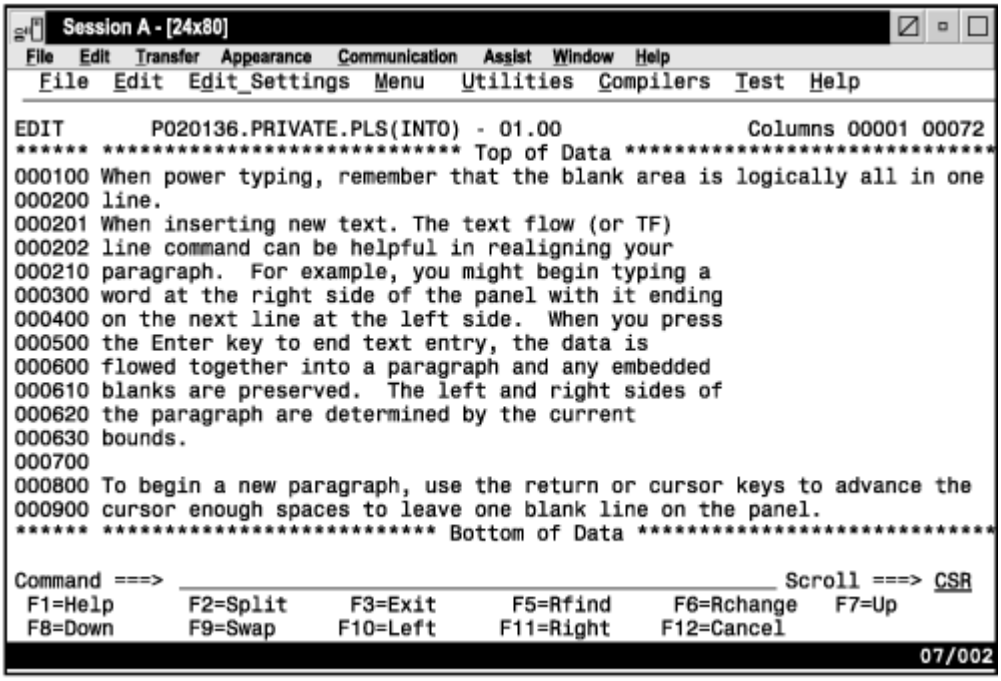
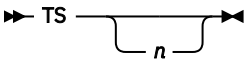


Figure 102. After the TF (Text Flow) line command

TS—Text Split

The TS (text split) line command moves part or all of a line of text to the following line. This makes it easier for you to add new material to existing text.

Syntax



n

The number of blank lines to be inserted between the split lines. If you do not type a number, or if the number that you type is 1, the editor inserts only one blank line.

Description

To split a line:

1. Type TS in the line command field of the line you would like to split. If you want to insert more than one blank line between the split lines, type a number greater than 1 immediately after the TS command.
2. Move the cursor to the desired split point.
3. Press Enter.

To rejoin lines, use the TF (text flow) line command. See “TF—Text Flow” on page 184 for more information.

For more information about splitting lines and other word processing commands, see “Word processing” on page 61 and “Splitting lines” on page 62.

Examples

Figure 103 on page 187 shows how to split text and to insert blank lines. To split the text and insert three lines, type TS3 in the line command field of the line you want to split and place the cursor where you want the line split.

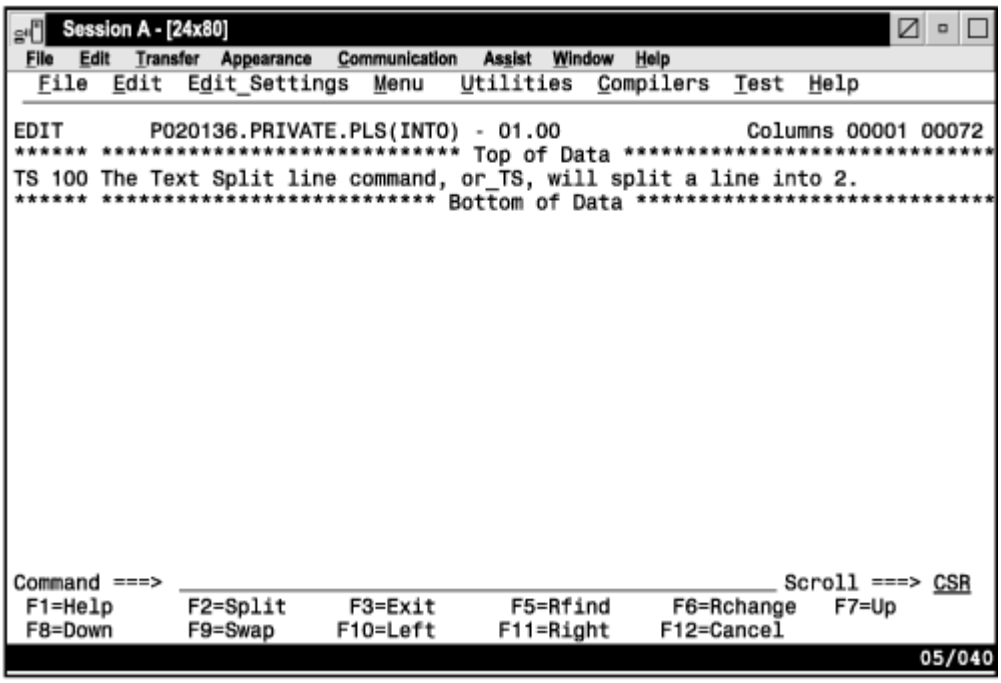


Figure 103. Before TS (Text Split) line command

When you press Enter, the line is split at the cursor position and the editor inserts the number of blank lines specified, as shown in Figure 104 on page 187.

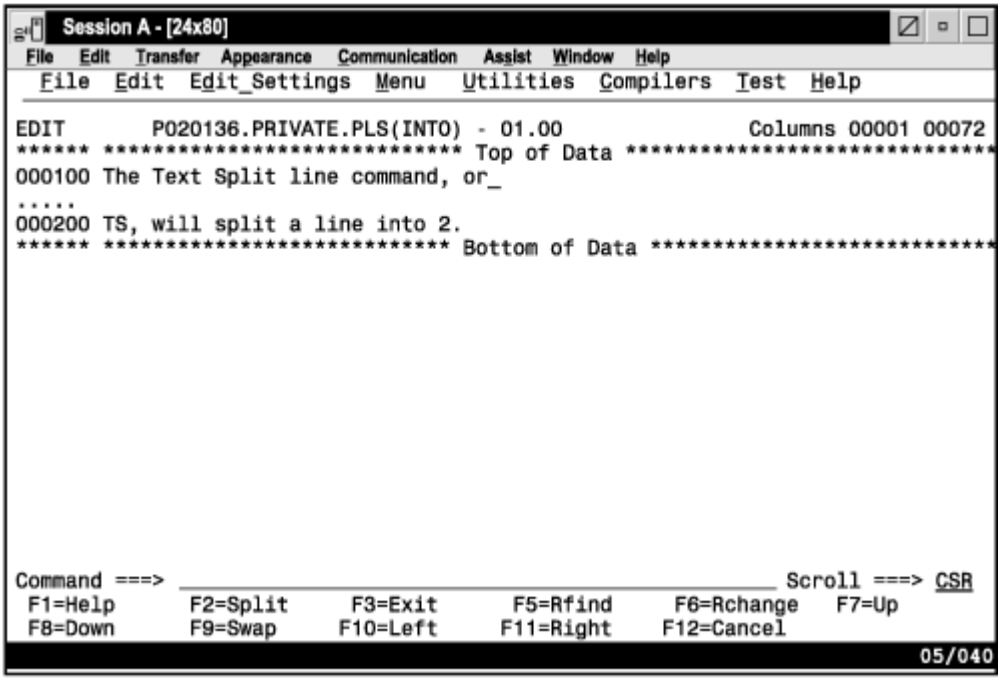
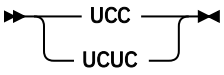


Figure 104. After TS (Text Split) line command

UC—Convert Characters to Uppercase

The UC (uppercase) line command converts characters in a data set or member from lowercase to uppercase. However, it does not affect the caps mode of the data that you are editing.

Syntax



n

The number of lines to be converted to uppercase. If you do not type a number, or if the number you type is 1, only the line on which you type UC is converted to uppercase.

Description

To convert characters on one or more lines to uppercase:

1. Type UC in the line command field of the source code line that contains the characters that you want to convert. To convert characters on lines following this one, type a number greater than 1 after the UC command.
2. Press Enter. The characters on the source code line or lines are converted to uppercase.

To convert characters in a block of lines to uppercase:

1. Type UCC in the line command field of both the first and last source code lines that contain characters that are to be converted. You can scroll (or use FIND or LOCATE) between typing the first UCC and the second UCC, if necessary.
2. Press Enter. The characters in the source code lines that contain the two UCC commands and in all of the source code lines between them are converted to uppercase.

See the "LC—Convert Characters to Lowercase" on page 165 line command and the CAPS primary command ("CAPS—Control Automatic Character Conversion" on page 204) and macro command ("CAPS—Set or Query Caps Mode" on page 309) for information about converting characters from uppercase to lowercase and vice versa.

Examples

Figure 105 on page 189 shows how to convert lines of text to uppercase. To convert lines of text to uppercase, place the UC command and the number of lines you want to convert in the line command field where you want the conversion to start.

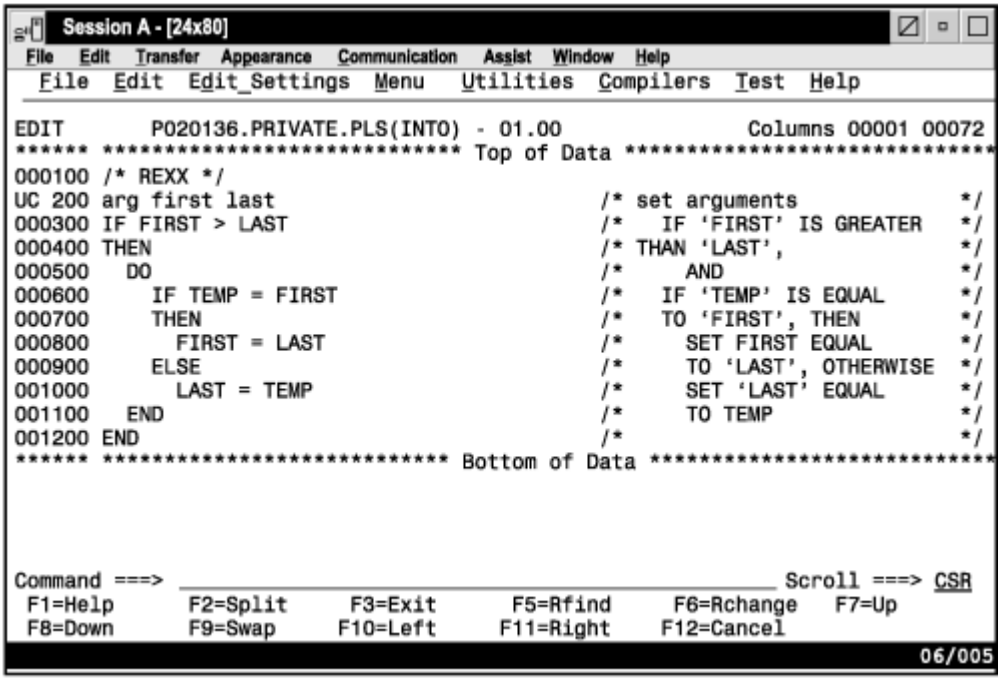


Figure 105. Before the UC (Uppercase) line command

When you press Enter, the editor converts the lines specified to uppercase. See Figure 106 on page 189.

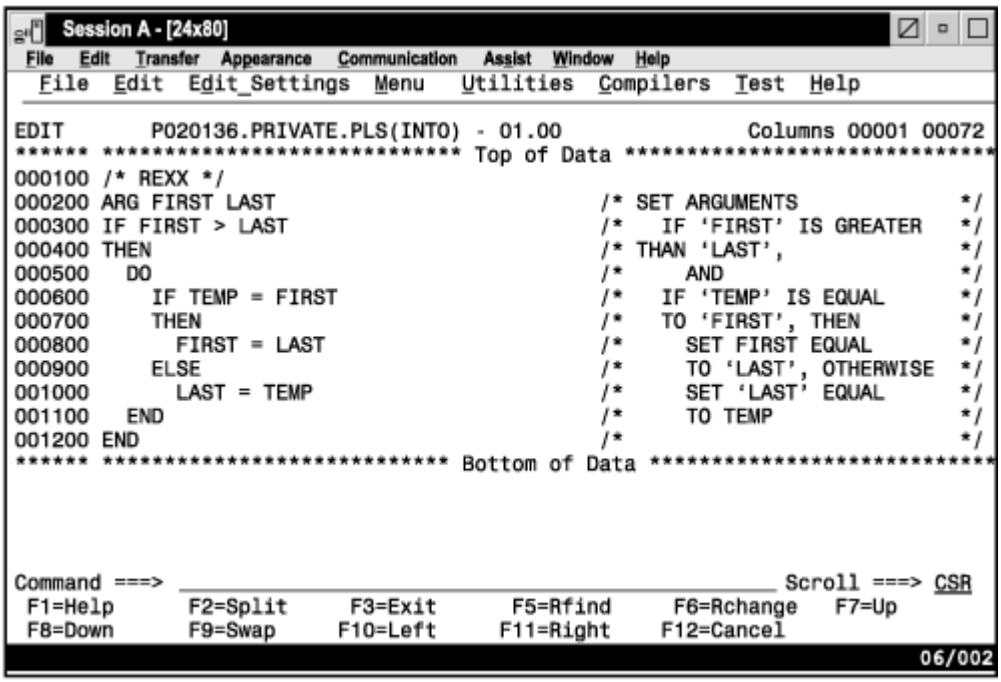


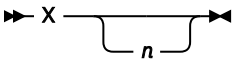
Figure 106. After the UC (Uppercase) line command

X—Exclude Lines

The X (exclude) line command replaces one or more lines on the panel with a dotted line. The dotted line contains a message that specifies how many lines have been excluded.

The excluded lines are not erased. They are simply hidden from view and can still be affected by edit line, primary, and macro commands.

Syntax



n
The number of lines to be excluded. If you do not type a number, or if the number that you type is 1, PDF excludes only the line on which you type the X command.

Description

To exclude one or more lines:

1. Type X in the line command field of the line that you want to exclude. If you want to exclude one or more lines that immediately follow this line, type a number greater than 1 immediately after the X command.
2. Press Enter. The lines are excluded from the panel.

To exclude a block of lines:

1. Type XX in the line command field of both the first and last lines that you want to exclude. You can scroll (or use FIND or LOCATE) between typing the first XX and the second XX, if necessary.
2. Press Enter. The lines that contain the two XX commands and all of the lines between them are excluded.

See “Excluding lines” on page 58 for more information on using this command.

Examples

Figure 107 on page 191 shows how lines are excluded from a member. To exclude six lines, type X6 in the line command field.

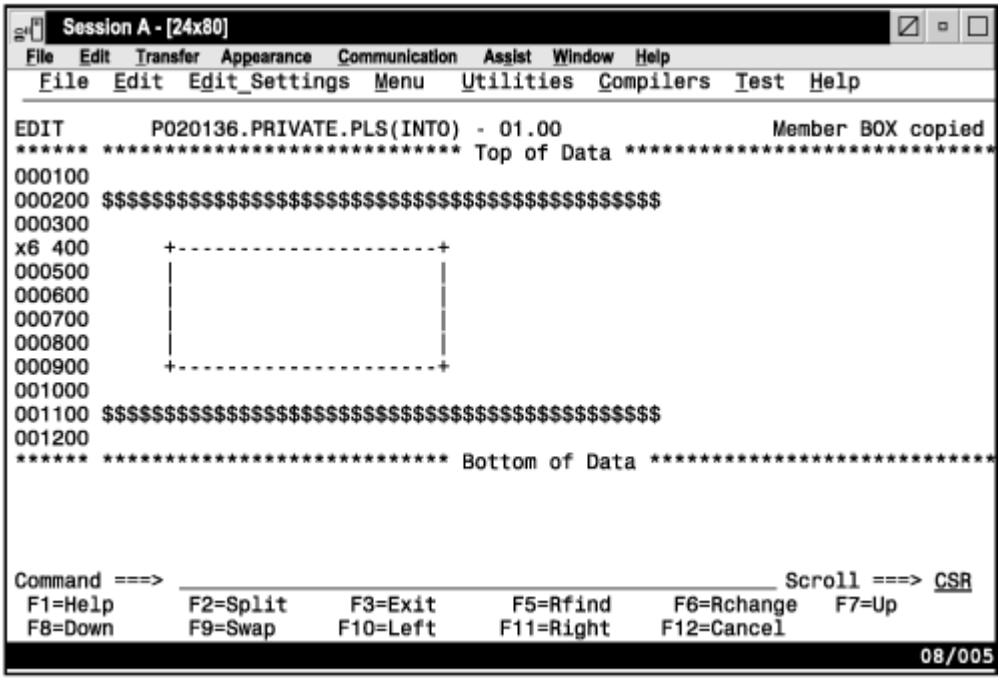


Figure 107. Before the X (Exclude) line command

When you press Enter, the editor excludes the specified lines. See Figure 108 on page 191.

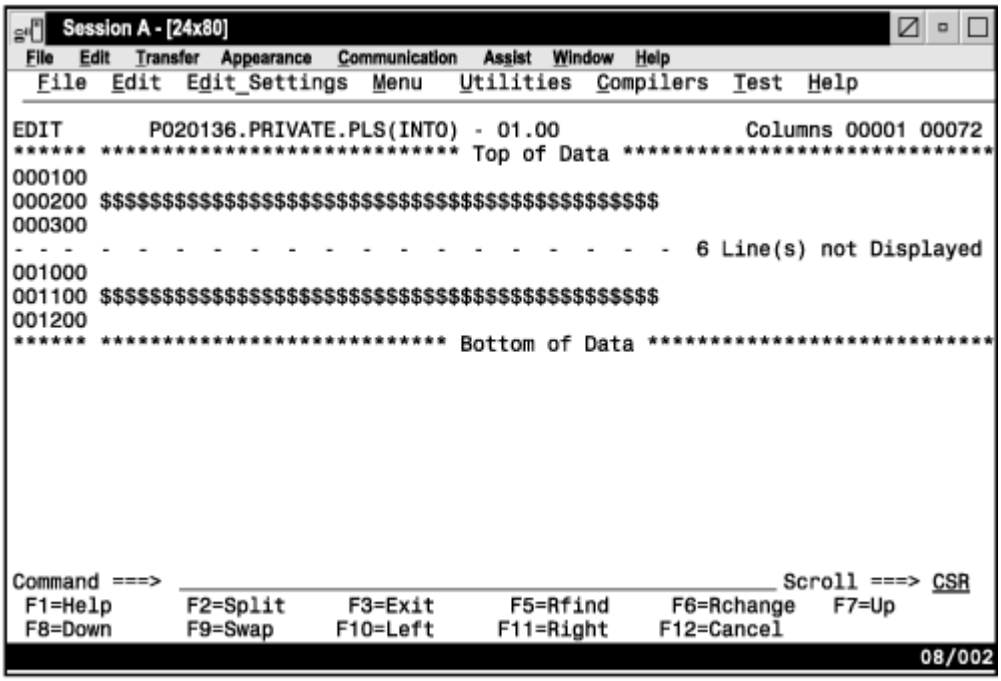


Figure 108. After the X (Exclude) line command

To redisplay excluded lines, use the F (show first line), L (show last line), or S (show lines) line command.

Chapter 10. Edit primary commands

Primary commands affect the entire data set being edited, whereas line commands usually affect only a single line or block of lines. To enter a primary command, either:

- Type the command on the command line and press Enter, or
- Press the function key to which the command is assigned

Most primary commands can be abbreviated. In fact, many can be typed as a single letter, such as L for LOCATE or F for FIND. In this topic, the syntax diagram for each command shows the allowable abbreviations, if any. For a complete list of command abbreviations, see Appendix A, " Abbreviations for Commands and Other Values," on page 429.

Each command description consists of:

Syntax
A syntax diagram for coding the command, including a description of any required or optional operands.

Description
A summary of the function and operation of the command. This definition also refers to other commands that can be used with this command.

Example
Sample usage of the command.

Edit primary command summary

This table summarizes the edit primary commands. See the complete description of the commands on the referenced page.

Table 17. Summary of the primary commands

Command	Description
"AUTOLIST—Create a Source Listing Automatically" on page 196	Controls the automatic printing of data to the ISPF list data set.
"AUTONUM—Number Lines Automatically" on page 198	Controls the automatic renumbering of data when it is saved.
"AUTOSAVE—Save Data Automatically" on page 199	If the data is changed, automatically saves it when you issue an END command.
"BOUNDS—Control the Edit Boundaries" on page 201	Sets the left and right boundaries.
"BROWSE—Browse from within an Edit Session" on page 202	Browses a data set or member without leaving your current edit session.
"BUILTIN—Process a Built-In Command" on page 203	Processes a built-in command even if a macro with the same name has been defined.
"CANCEL—Cancel Edit Changes" on page 203	Ends the edit session without saving any of the changes.
"CAPS—Control Automatic Character Conversion" on page 204	Sets caps mode.

Table 17. Summary of the primary commands (continued)

Command	Description
"CHANGE—Change a Data String" on page 205	Changes a data string into another string.
"COMPARE—Edit Compare" on page 209	Compares library member or data set with the data being edited.
"COPY—Copy Data" on page 212	Copies a library member or data set into the data being edited.
"CREATE—Create Data" on page 217	Writes the data you are editing into a library member or data set only if it does not already exist.
"CUT—Cut and Save Lines" on page 220	Saves lines to a clipboard for later retrieval by PASTE command.
"DEFINE—Define a Name" on page 222	• Assigns an alias to a macro or built-in command. • Disables the use of a macro or built-in command. • Identifies a macro that replaces a built-in command of the same name. • Identifies programs that are edit macros.
"DELETE—Delete Lines" on page 224	Deletes lines from the data you are editing.
"EDIT—Edit from within an Edit Session" on page 226	Edits a data set or member without leaving your current edit session (recursive edit).
"EDITSET—Display the Editor Settings Dialog" on page 228	Causes the Edit Settings panel to be displayed.
"END—End the Edit Session" on page 231	Ends the current edit session.
"EXCLUDE—Exclude Lines from the Display" on page 232	Excludes lines from the panel.
"FIND—Find a Data String" on page 234	Finds a data string.
"FLIP—Reverse Exclude Status of Lines" on page 236	Reverses the exclude status of a specified range of lines in a file or all the lines in the file.
"HEX—Display Hexadecimal Characters" on page 238	Specifies whether the hexadecimal form of the data should be displayed.
"HIDE—Hide Excluded Lines Message" on page 241	Removes the "*n* Line(s) not Displayed" messages from the display where lines have been excluded by the EXCLUDE command.
"HILITE—Enhanced Edit Coloring" on page 242	Highlights in user-specified colors many language-specific constructs, program logic features, the phrase containing the cursor, and any strings that match the previous FIND operation or those that would be found by an RFIND or RCHANGE request. Can also be used to set default colors for the data area in non-program files and for any characters typed since the previous Enter or function key entry.
"IMACRO—Specify an Initial Macro" on page 246	Saves the name of an initial macro in the edit profile.

Table 17. Summary of the primary commands (continued)

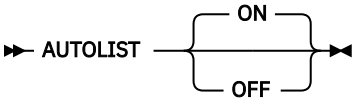
Command	Description
"LEVEL—Specify the Modification Level Number" on page 247	Sets the modification level number to be kept as part of the PDF library statistics.
"LOCATE—Locate a Line" on page 249	Locates a line.
"MODEL—Copy a Model into the Current Data Set" on page 251	Copies a model into the data you are editing or defines the current model class.
"MOVE—Move Data" on page 254	Moves a library member or data set into the data you are editing.
"NONUMBER—Turn Off Number Mode" on page 258	Turns off number mode.
"NOTES—Display Model Notes" on page 258	Specifies whether the MODEL command is to display notes.
"NULLS—Control Null Spaces" on page 259	Controls null spaces.
"NUMBER—Generate Sequence Numbers" on page 260	Generates sequence numbers.
"PACK—Compress Data" on page 262	Specifies whether data is to be stored normally or compressed.
"PASTE—Move or Copy Lines from Clipboard" on page 263	Moves or copies lines from a clipboard into an edit session.
"PRESERVE—Enable Saving of Trailing Blanks" on page 264	Specifies whether trailing blanks should be saved when data is stored.
"PROFILE—Control and Display Your Profile" on page 264	Controls and displays your profile.
"RCHANGE—Repeat a Change" on page 267	Repeats the most recently processed CHANGE command.
"RECOVERY—Control Edit Recovery" on page 268	Controls edit recovery.
"RENUM—Renumber Data Set Lines" on page 269	Renumbers data set lines.
"REPLACE—Replace Data" on page 271	Writes the data you are editing into a library member even if it already exists.
"RESET—Reset the Data Display" on page 275	Resets the data display.
"RFIND—Repeat Find" on page 277	Locates the data string defined by the most recently processed SEEK, FIND, or CHANGE command, or excludes a line that contains the data string from the previous EXCLUDE command.
"RMACRO—Specify a Recovery Macro" on page 278	Saves the name of a recovery macro in the edit profile.
"SAVE—Save the Current Data" on page 278	Saves the current data without ending the edit session.
"SETUNDO—Set the UNDO Mode" on page 280	Sets the UNDO mode.
"SORT—Sort Data" on page 282	Puts data in a specified order.
"STATS—Generate Library Statistics" on page 285	Specifies whether PDF library statistics are to be created when this member is saved.

Table 17. Summary of the primary commands (continued)	
Command	**Description**
"SUBMIT—Submit Data for Batch Processing" on page 285	Submits the data you are editing for batch processing.
"TABS—Define Tabs" on page 286	Defines tab positions for software, hardware, and logical tabs.
"UNDO—Reverse Last Edit Interaction" on page 288	Removes the data modifications of a previous interaction.
"UNNUMBER—Remove Sequence Numbers" on page 290	Removes sequence numbers.
"VERSION—Control the Version Number" on page 292	Sets the version number to be kept as part of the PDF library statistics.
"VIEW—View from within an Edit Session" on page 294	View a data set or member without leaving your current edit session.

AUTOLIST—Create a Source Listing Automatically

The AUTOLIST primary command sets autolist mode, which controls the automatic printing of data to the ISPF list data set.

Syntax



ON
Generates a source listing in the ISPF list data set for eventual printing when you end an edit session in which you changed and saved data.

OFF
No source listing is generated.

Description

Autolist mode is saved in the edit profile. To check the current setting of autolist mode:

1. On the command line, type:

```
PROFILE 3
```

2. Press Enter. The third line of the edit profile shows the autolist mode setting.

To turn on autolist mode:

1. On the command line, type:

```
AUTOLIST ON
```

2. Press Enter.

To turn off autolist mode:

1. On the command line, type:

```
AUTOLIST OFF
```

2. Press Enter.

Examples

This example shows how to use the AUTOLIST command to save a copy of a source code listing in the ISPF list data set and to print the list data set.

1. As you edit a data set, you decide to store a listing of the source code in the ISPF list data set so that you can print it later. Enter the PROFILE 3 command to display the first 3 lines of the edit profile. This shows you whether autolist mode is on or off.

```
PROFILE 3
```

2. You can see from the edit profile that autolist mode is off:

```
 =PROF> ....PLI (VARIABLE - 72)....RECOVERY ON....NUMBER OFF.....................
 =PROF> ....CAPS OFF....HEX OFF....NULLS OFF....TABS OFF.........................
 =PROF> ....AUTOSAVE ON....AUTONUM OFF....AUTOLIST OFF....STATS ON...............
```

3. Enter the AUTOLIST ON command to turn on autolist mode:

```
AUTOLIST ON
```

   The edit profile changes accordingly:

```
 =PROF> ....PLI (VARIABLE - 72)....RECOVERY ON....NUMBER OFF.....................
 =PROF> ....CAPS OFF....HEX OFF....NULLS OFF....TABS OFF.........................
 =PROF> ....AUTOSAVE ON....AUTONUM OFF....AUTOLIST ON....STATS ON...............
```

4. After editing the data set, save your changes by entering the END command. The changes are saved because, as you can see in the preceding partial edit profile, autosave mode is on.

```
END
```

   ISPF creates a list data set with the contents of the data set member that you were editing. The name of the list data set is:

```
prefix.user-id.SPFn.LIST
```

   Note: See *z/OS ISPF User's Guide Vol I* for information about list data sets.

5. Before leaving ISPF, use the jump function to go to option 0.2 and check the log/list defaults:

```
=0.2
```

   The Log and List Defaults panel shows the current default settings for the handling of log and list data sets.

6. Because you want to print the list data set, make sure that the PD option is entered in the Process Option field under the List Data Set Default Options heading:

```
Process option   ===> PD
```

   Note: Also, make sure that the appropriate JCL information is entered at the bottom of the Log and List Defaults panel so that the print job is submitted.

7. You can now end the session, knowing that the list data set will be printed:

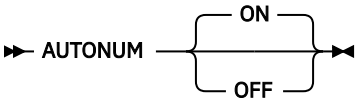
```
=X
```

8. When the session ends, TSO displays a message that says the print job has been submitted.

AUTONUM—Number Lines Automatically

The AUTONUM primary command sets autonum mode, which controls the automatic renumbering of data when it is saved.

Syntax



ON
Turns on automatic renumbering. When number mode is also on, the data is automatically renumbered when it is saved.

OFF
Turns off automatic renumbering. Data is not renumbered.

Description

When number mode is on (see (xref refid="number"), the first line of a data set or member is normally line number 000100, the second number is 000200, and so forth. However, as lines are inserted and deleted, the increment between line numbers can change.

For example, you might think that when a line is inserted between 000100 and 000200, line 000200 would be given the number 000300 and the new line would become 000200. Instead, the existing lines retain their numbers and the new line is given line number 000110.

Therefore, if the original line number increments are important to you, the AUTONUM command renumbers your lines automatically so that the original increments are maintained.

Autonum mode is saved in the edit profile. To check the current settings of number mode and autonum mode:

1. On the command line, type:

```
PROFILE 3
```

2. Press Enter. The first line of the edit profile shows the number mode setting and the third line shows the autonum mode setting.

To turn on autonum mode:

1. On the command line, type:

```
AUTONUM ON
```

2. Press Enter.

To turn off autonum mode:

1. On the command line, type:

```
AUTONUM OFF
```

2. Press Enter.

Examples

This example shows a practical application of AUTONUM command usage. You have been editing a data set with number mode on.

Note: If you are editing a data set or member with number mode off and then decide to turn number mode on, make sure that columns 1 through 6 of your data set are blank. Otherwise, the sequence numbers created by the NUMBER command can overlay any of your data in columns 1 through 6. Use either the COLUMN SHIFT or DATA SHIFT line command to indent the data.

You now want to end the edit session. However, since you had to insert and delete many lines, your line numbering is no longer uniform. Therefore, you decide to use autonum mode so that the next time you edit this data set the line numbers will be correct.

1. First, check the edit profile to see whether autonum mode is already on by entering the PROFILE 3 command to display the first 3 lines of the edit profile.

   ```
   PROFILE 3
   ```

2. You can see from the edit profile that autonum mode is off:

   ```
   =PROF> ....PLI (VARIABLE - 72)....RECOVERY ON....NUMBER OFF.....................
   =PROF> ....CAPS OFF....HEX OFF....NULLS OFF....TABS OFF.........................
   =PROF> ....AUTOSAVE ON....AUTONUM OFF....AUTOLIST OFF....STATS ON...............
   ```

3. Enter the AUTONUM ON command to turn on autonum mode:

   ```
   AUTONUM ON
   ```

   The edit profile changes accordingly:

   ```
   =PROF> ....PLI (VARIABLE - 72)....RECOVERY ON....NUMBER OFF.....................
   =PROF> ....CAPS OFF....HEX OFF....NULLS OFF....TABS OFF.........................
   =PROF> ....AUTOSAVE ON....AUTONUM ON....AUTOLIST ON....STATS ON.................
   ```

4. After editing the data set, save your changes by entering the END command. The changes will be saved because, as you can see in the preceding partial edit profile, autosave mode is on.

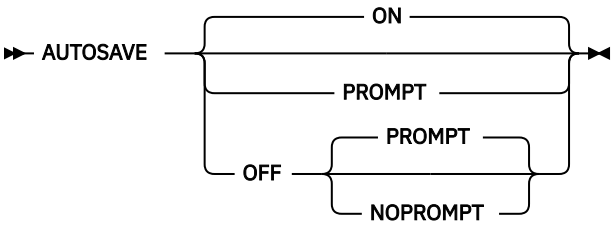
   ```
   END
   ```

   ISPF saves the data set that you were editing, along with any changes. The next time you edit the data set, the line numbers will have the proper increments.

AUTOSAVE—Save Data Automatically

The AUTOSAVE primary command sets autosave mode, which controls whether changed data is saved when you enter END.

Syntax



ON
Turns autosave mode on. When you enter END, any changed data is saved.

OFF PROMPT
Turns autosave mode off with the PROMPT operand. You are notified that changes have been made and that either the SAVE command (followed by END) or CANCEL must be used. When you use AUTOSAVE PROMPT by itself, it implies the OFF command.

OFF NOPROMPT
Turns autosave mode off with the NOPROMPT operand. You are not notified and the data is not saved when you issue an END command. END becomes an equivalent to CANCEL. Use the NOPROMPT operand with caution.

Description

Data is considered changed if you have operated on it in any way that could cause a change. Shifting a blank line or changing a word to the same word does not actually alter the data, but the editor considers this data changed. When you enter SAVE, the editor resets the change status.

Autosave mode, along with the PROMPT operand, is saved in the edit profile. To check the current setting of autosave mode:

1. On the command line, type:

   ```
   PROFILE 3
   ```

2. Press Enter. The third line of the edit profile shows the autosave mode setting.

To turn on autosave mode:

1. On the command line, type:

   ```
   AUTOSAVE
   ```

   Note: This is the equivalent of entering AUTOSAVE ON.
2. Press Enter. The next time you enter END, any changes that you made to the data set or member that you were editing are saved.

To turn off autosave mode:

1. On the command line, type:

   ```
   AUTOSAVE OFF
   ```

   Note: This is the equivalent of entering AUTOSAVE OFF PROMPT.
2. Press Enter. The next time you enter END when a data set or member has been changed, the editor prompts you to specify whether you want changes to the data set or member saved (SAVE) or not saved (CANCEL). However, if no changes have been made to the data set or member, the edit session ends without a prompt.

To turn off autosave mode and specify that you do not want to be prompted when data has changed:

1. On the command line, type:

   ```
   AUTOSAVE OFF NOPROMPT
   ```

2. Press Enter. The next time you enter END when a data set or member has been changed, the edit session ends without saving your changes, just as if you had entered CANCEL. You are not prompted to save the changes.

For more information on saving data, see the CANCEL and END primary commands, and the DATA_CHANGED, CANCEL, and END macro commands.

Examples

This example shows a practical application of AUTOSAVE usage.

1. You have been editing a data set member and now want to end the edit session. Enter END:

   ```
   END
   ```

2. The member that you were editing remains with this message in the upper-right corner:

   ```
   DATA CHANGED-SAVE/CANCEL
   ```

   This message implies that autosave mode in the edit profile is set to AUTOSAVE OFF PROMPT. You are prompted to enter either SAVE to save your changes, or CANCEL to end the edit session without saving your changes.

   You also have the option to change autosave mode in the edit profile to AUTOSAVE ON. By doing so, the next time you enter END, your changes will be saved and the edit session will end.
3. You decide to turn on autosave mode:

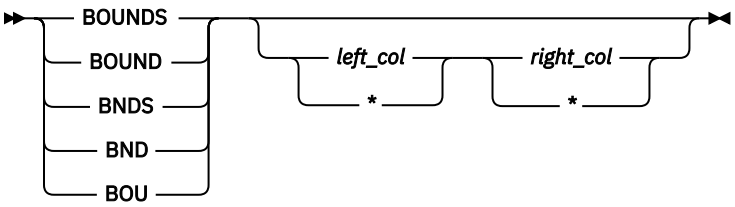
   ```
   AUTOSAVE ON
   ```
4. Then you enter END again to save your changes and end the edit session.

   ```
   END
   ```

BOUNDS—Control the Edit Boundaries

The BOUNDS primary command sets the left and right boundaries and saves them in the edit profile.

Syntax



left_col
The left boundary column to be set.

right_col
The right boundary column to be set.

***
The current value of the boundary.

To reset the boundaries to the default columns:

1. On the command line, type:

   ```
   BOUNDS
   ```
2. Press Enter. The boundaries are reset to the default columns.

See “Edit boundaries” on page 23 for more information, including tables that show commands affected by bounds settings and default bounds settings for various types of data sets.

The column numbers are always data column numbers (see “Referring to column positions” on page 106). Thus, for a variable format data set with number mode on, data column 1 is column 9 in the record.

You cannot specify the same column for both boundaries.

Description

The BOUNDS primary command provides an alternative to setting the boundaries with the BOUNDS line command or macro command; the effect on the member or data set is the same. However, if you use both

the BOUNDS primary command and the BOUNDS line command in the same interaction, the line command overrides the primary command.

Examples

To set the left boundary to 1 and the right boundary to 72, type:

```
BOUNDS 1 72
```

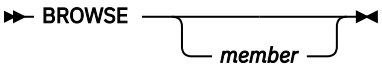
To set the left boundary to 10 and leave the right as is, type:

```
BOUNDS 10 *
```

BROWSE—Browse from within an Edit Session

The BROWSE primary command allows you to browse a sequential data set, partitioned data set member, or z/OS UNIX file during your current edit session.

Syntax



member
A member of the ISPF library or other partitioned data set you are currently editing. You may enter a member pattern to generate a member list.

Description

To browse a data set, member, or z/OS UNIX file during your current edit session:

1. On the command line, type:

   ```
   BROWSE
   ```

   or

   ```
   BROWSE member
   ```

   Here, *member* represents the name of a member of the partitioned data set you are editing. The member operand is optional.
2. Press Enter. If you specified a member name, the current library concatenation sequence finds the member. The member displays for browsing. If you do not specify a member name, the Browse Command Entry panel, which is similar to the regular Browse Entry panel, appears. You can enter the name of any sequential data set, partitioned data set, or z/OS UNIX file to which you have access. When you press Enter, the data set, member, or z/OS UNIX file displays for browsing. The editor suspends your initial edit session until the browse session is complete.
3. To exit from the browse session, enter the END command. The current session resumes.

Examples

To browse member YYY of the current library concatenation:

1. On the command line, type:

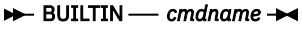
   ```
   BROWSE YYY
   ```

2. Press Enter.

BUILTIN—Process a Built-In Command

You can use the BUILTIN primary command with edit macros and the DEFINE command to process a built-in edit primary command, even if a macro has been defined with the same name.

Syntax



cmdname
The built-in command to be processed.

Description

To process a built-in primary command instead of a command with the same name that has been defined as an alias:

1. On the command line, type:

   ```
   BUILTIN cmdname
   ```

   where *cmdname* is the name of a primary command.
2. Press Enter. The edit primary command is processed.

Examples

This example shows a practical application of BUILTIN command usage.

1. You have a macro named MACEND that you have created. You want to run your MACEND macro instead of ISPF's built-in END command. Enter this command:

   ```
   DEFINE END ALIAS MACEND
   ```

   Note: If the END command is issued in your MACEND macro without being preceded by the BUILTIN macro command, the MACEND macro would be run again, resulting in a loop.
2. To run your MACEND macro, enter:

   ```
   END
   ```
3. To end the edit session without redefining END, use BUILTIN, as follows:

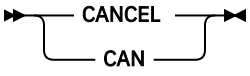
   ```
   BUILTIN END
   ```

   This command issues ISPF's built-in END command instead of your MACEND macro.

CANCEL—Cancel Edit Changes

The CANCEL primary command ends your edit session without saving any of the changes you have made.

Syntax



Description

CANCEL is especially useful if you have changed the wrong data, or if the changes themselves are incorrect. To cancel changes to a data set:

1. On the command line, type:

   ```
   CANCEL
   ```

2. Press Enter. The edit session ends without saving your changes.

Note: If you issue SAVE and later issue CANCEL, the changes you made before issuing SAVE are not canceled.

See the DATA_CHANGED, AUTOSAVE, and END commands for more information about saving data.

CANCEL does not cause automatic recording in the ISPF list data set, regardless of the setting of the autolist mode.

Examples

After editing the data, you decide that you want the data set the way it was before editing. Enter this command:

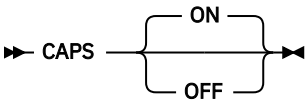
```
CANCEL
```

The edit session ends with the data set in its original state.

CAPS—Control Automatic Character Conversion

The CAPS primary command sets the caps mode, which controls whether alphabetic data that you type at the terminal is automatically converted to uppercase during the edit session.

Syntax



ON
Turns caps mode on.

OFF
Turns caps mode off.

Description

The editor sets the caps mode according to the data in the file retrieved for editing. If caps mode has been on and the data contains lowercase letters, the mode switches and the editor displays a message indicating the change. Likewise, if caps mode is off and the editor contains all uppercase letters, the mode switches and the editor displays a message.

Caps mode is saved in the edit profile. To override the automatic setting of caps mode, you can include the CAPS command in an initial macro.

Caps mode is usually on during program development work. When caps mode is on, any alphabetic data that you type, plus any other alphabetic data that already exists on that line, is converted to uppercase when you press Enter or a function key.

To set caps mode on:

1. On the command line, type:

   ```
   CAPS
   ```

2. Press Enter. Caps mode is set to on in the edit profile.

Caps mode is usually off when you edit text documentation. When caps mode is set to off, any alphabetic data that you type remains just as you typed it. If you typed it in uppercase, it stays in uppercase; if you typed it in lowercase, it stays in lowercase. Alphabetic data already typed on a line is not affected. To set caps mode off:

1. On the command line, type:

   ```
   CAPS OFF
   ```

2. Press Enter. Caps mode is set to off in the edit profile.

The CAPS command does not apply to DBCS fields in formatted data or to DBCS fields in mixed fields. If you specify CAPS, the DBCS fields remain unchanged.

See the LC (lowercase) and UC (uppercase) line commands and the CAPS macro command for more information about changing case.

Examples

This example shows a practical application of CAPS command usage.

1. You are editing a data set that contains all uppercase letters, with caps mode off. The data you are typing contains both uppercase and lowercase letters, but you want all of the letters to be uppercase. On the command line, type:

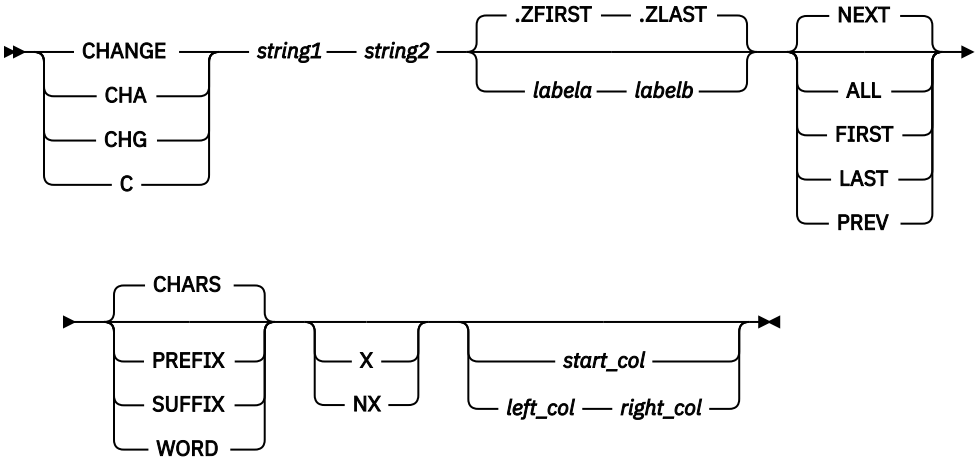
   ```
   CAPS
   ```

2. Press Enter.
3. Move the cursor back to the line on which you were typing.
4. Finish typing the line or type over one or more of the existing letters.
5. Press Enter. All of the letters on the line are converted to uppercase.

CHANGE—Change a Data String

The CHANGE primary command changes one string into another.

Syntax



string1
The search string you want to change. See "Finding, seeking, changing, and excluding data" on page 44.

string2
The string you want to replace *string1*. See "Finding, seeking, changing, and excluding data" on page 44.

labela*, *labelb
Labels identifying the start and end of the group of lines the CHANGE command is to search.

For more information about using labels to identify a group of lines, see "Labels and line ranges" on page 59.

NEXT
Starts at the first position after the current cursor location and searches ahead to find the next occurrence of *string1*.

ALL
Starts at the top of the data and searches ahead to find all occurrences of *string1*.

FIRST
Starts at the top of the data and searches ahead to find the first occurrence of *string1*.

LAST
Starts at the bottom of the data and searches backward to find the last occurrence of *string1*.

PREV
Starts at the current cursor location and searches backward to find the previous occurrence of *string1*.

CHARS
Locates *string1* anywhere the characters match.

PREFIX
Locates *string1* at the beginning of a word.

SUFFIX
Locates *string1* at the end of a word.

WORD
Locates *string1* when it is delimited on both sides by blanks or other non-alphanumeric characters.

X
Scans only lines that are excluded from the display.

NX
Scans only lines that are not excluded from the display.

start_col
The first column to be included in the range of columns to be searched. When you specify only one column, the editor finds the string only if the string starts in the specified column.

left_col
The first column to be included in the range of columns to be searched.

right_col
The last column to be included in the range of columns to be searched.

Note:

1. For more information about restricting the search to only a portion of each line, see “Limiting the search to specified columns” on page 54.
2. The CHANGE command allows you to control the starting point and the direction of the search by positioning the cursor and using either the NEXT or PREV operand. For more information, see “Starting point and direction of the search” on page 52.

Description

You can use the CHANGE command with the FIND and EXCLUDE commands to find a search string, change it, and then exclude the line that contains the string from the panel.

To change the next occurrence of "ME" to "YOU" without specifying any other qualifications:

1. On the command line, type:

```
CHANGE ME YOU
```

2. Press Enter. This command changes only the next occurrence of the letters "ME" to "YOU". Since no other qualifications were specified, the letters "ME" can be:
   - Uppercase or a mixture of uppercase and lowercase
   - At the beginning of a word (prefix), the end of a word (suffix), or the entire word (word)
   - In an excluded line or a non-excluded line
   - Anywhere within the current boundaries

To change the next occurrence of "ME" to "YOU", but only if the letters are uppercase:

1. On the command line, type:

```
CHANGE C'ME' YOU
```

2. Press Enter. This type of change is called a character string change (note the C that precedes the search string) because it changes the next occurrence of the letters ME to YOU only if the letters are found in uppercase. However, since no other qualifications were specified, the change occurs no matter where the letters are found, as outlined in the preceding list.

For more information, including other types of search strings, see “Finding, seeking, changing, and excluding data” on page 44.

Examples

The example shown changes the first plus ("+") in the data set to a minus ("-"). However, the plus must occur on or between lines labeled .E and .S and it must be the first character of a word:

```
CHANGE '+' '-' .E .S FIRST PREFIX
```

The example shown changes the last plus in the data set to a minus. However, the plus must occur on or between lines labeled .E and .S; it must be the last character of a word; and it must be found on an excluded line:

```
CHANGE '+' '-' .E .S LAST SUFFIX X
```

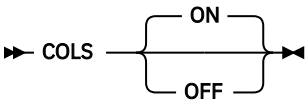
The example shown changes the plus that immediately precedes the cursor position to a minus. However, the cursor must not be positioned ahead of the lines labeled .E and .S. Also, the plus must occur on or between the labeled lines; it must be a standalone character (not part of any other word); it must be on a non-excluded line; and it must exist within columns 1 and 5:

```
CHANGE '+' '-' .E .S PREV WORD NX 1 5
```

COLS—Display Fixed Columns Line

The COLS primary command displays a non-scrolling columns indicator line at the top of the data area.

Syntax



ON
Display columns line.

OFF
Remove columns line from the display.

Description

The COLS command displays a columns indicator line at the top of the data area in Edit and View mode. This works in the same manner as the columns line under Browse.

The columns line differs from that displayed by the COLS line command in that the line command field is protected. This means that it cannot be copied, moved, or deleted by overtyping with line commands. The line does not scroll with the data, and therefore the number of data lines displayed is reduced by one.

Entering COLS with no parameter toggles the display to the opposite. For example, if the columns line is currently displayed, entering COLS removes it.

Examples

To display the columns indicator line, enter this command:

```
COLS ON
```

Figure 109 on page 209 shows an example of an edit screen displaying the columns indicator line.

```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
EDIT       LEEBURR.TEST($$ZZZZ) - 01.10                 Columns 00001 00072
=COLS> ----+----1----+----2----+----3----+----4----+----5----+----6----+----7--
****** ***************************** Top of Data ******************************
000001 //LEEBURRC JOB CLASS=A,MSGCLASS=X
000002 //STEPPLX  EXEC PGM=AKEEPLX,REGION=2048K,PARM='SOURCE(SEG)'
000003 //SYSPRINT DD  SYSOUT=A
000004 //SYSUT1   DD  UNIT=SYSDA,SPACE=(TRK,(30,10))
000005 //SYSUT2   DD  UNIT=SYSDA,DSN=&&ASM,DISP=(NEW,PASS),
000006 //             SPACE=(TRK,(30,10))
000007 //SYSUT3   DD  UNIT=SYSDA,SPACE=(TRK,(30,10))
000008 //SYSUT4   DD  UNIT=SYSDA,SPACE=(TRK,(30,10))
000009 //SYSLIB   DD  DISP=SHR,DSN=PDFTOS2C.LEEBURR.SOURCE
000010 //         DD  DISP=SHR,DSN=PDFTOS2C.APARTEST.SOURCE
⋮

Command ===> _____________________________________________ Scroll ===> CSR
 F1=Help      F2=Split     F3=Exit      F5=Rfind     F6=Rchange   F7=Up
 F8=Down      F9=Swap     F10=Left     F11=Right    F12=Cancel
```

Figure 109. Member with COLS indicator line

COMPARE—Edit Compare

The COMPARE command compares the file you are editing with an external sequential data set, member of a partitioned data set, or z/OS UNIX file. Lines that exist only in the file being edited are marked, and lines that exist only in the file being compared are inserted as information lines in the file being edited. The command operates as a primary command or an edit macro command.

If you compare the file you are editing with a member of a PDSE version 2 data set that is configured for member generations, the current generation of the member is used for the comparison.

You can use the Delete and Make Data line commands to merge changes between files that are being compared.

The COMPARE function supports all line lengths, but some SuperC options are ignored for line lengths greater than 256 characters long.

When you are editing a cataloged data set, explicit data set names refer to cataloged data sets. However, if you are editing an uncataloged data set and specify only a member name, COMPARE searches for the member in the current uncataloged data set. For example, if you are editing an uncataloged data set called "userid.TEMP", then the command

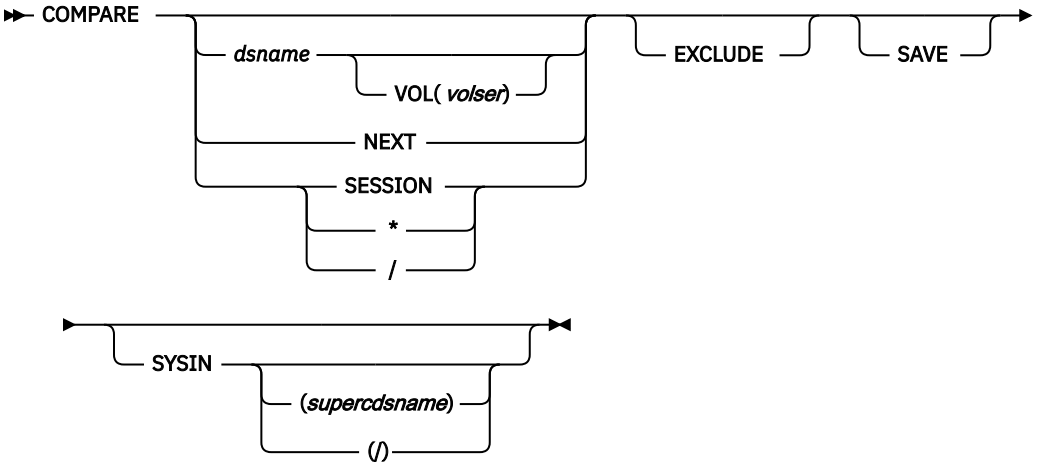
```
COMPARE TEMP
```

first looks for member TEMP in the current, uncataloged data set, then looks for a cataloged data set named TEMP (TSO prefix rules apply). If it finds data set TEMP, and the data set being edited is a PDS member, then the same named member is searched for in data set TEMP.

Use of COMPARE when editing concatenations that contain uncataloged data sets is not supported and can lead to unpredictable results.

If you have made changes to the data before issuing the COMPARE command, the COMPARE command uses the current contents of the edit session during the comparison. Because COMPARE does not require the data to be saved on disk, you can use the COMPARE command from EDIF, VIIF, or EDIREC sessions. However, COMPARE NEXT and COMPARE SESSION are *not* supported in EDIF, VIIF, or EDIREC sessions.

Syntax



no operand
The "Edit Compare Settings and/or Command Parameters" panel is displayed.

This panel enables you to customize the comparison by selecting the relevant SuperC options to use. The comparison is always a LINE compare with the options UPDLDEL, NOLISTL, LINECMP, and CKPACKL specified.

You can also specify Compare Command Parameters. The Name field is used to specify the dsname, NEXT, or * (session) parameters. The Volume field is used to enter the volume serial for an uncataloged data set. The Exclude field is used to specify the EXCLUDE parameter. The SYSIN field is used to specify the SYSIN parameter. The Save field is used to specify the SAVE parameter. The Set SYSIN data set field is used to display a panel where the SYSIN data set name can be specified. See below for a description of these parameters.

The SEQ, NOSEQ, or COBOL keywords are automatically specified depending on the NUMBER state in the edit profile. Mixed data can be enabled, and is always assumed to be specified when you are in an edit session with MIXED specified in the profile. Each field in the Edit Compare Settings and/or Command Parameters panel has field level help.

Note: When *don't process* (DP) options are used, the resulting display shows DP lines in the current file as unlabeled and does not show DP lines from the comparison file. This can be misleading. Because comparisons which ignore parts of the file might show data in one file and not in the other, use caution when using DP options. When you use options that ignore programming language comments, the *don't process reformatted lines* option is recommended.

dsname
The name of a member, data set, or z/OS UNIX file to which the current file is compared. This variable can be specified as a fully qualified data set name (in quotation marks), a partially qualified data set name, a member name, or path name. (Also, see “Specifying z/OS UNIX pathnames with edit primary and macro commands” on page 16.)

If you specify only a member name, it can be preceded by a left parenthesis symbol. The right parenthesis is allowed but not required. The current edit session must be of a member of a partitioned data set. The current edit concatenation is searched for the member to compare.

If you specify only a data set name and the current file is a member of a PDS, then the specified data set is searched for a member of the same name as the member being edited.

VOL(*volser*)
Used when comparing against an uncataloged data set. Specifies the volser of the volume containing the uncataloged data set.

NEXT
Specifies to do a comparison between the currently edited member and the next member of the same name found at a higher level of the hierarchy (or next level of the edit concatenation) than the current

member. For example, if the current member is found in the third level of the concatenation, and a like-named member exists at the fourth level, then the third and fourth level members are compared. After data is saved in the lowest level, compares are done from that level upward. If you specify *dsname*, the NEXT keyword cannot be used.

SESSION
Specifies that you want to compare the changes you have made during the edit session with the copy of the data saved on disk. Use COMPARE SESSION (or COMPARE *) to see the changes you have made to the edit data since the beginning of the edit session or since the last SAVE command.

*****
Same as SESSION.

EXCLUDE
Specifies that all matching lines in the compared data sets are excluded from the display *except* for a specified number of lines above and below the differences. The differences themselves are also shown in the display. The specified number of lines that are shown is set on the Edit Compare Settings and/or Command Parameters panel. If you do not specify a new number for this edit session, then whatever was the last number set is still valid. To change this number, issue the COMPARE command with no operand and change the EXCLUDE field on the Edit Compare Settings and/or Command Parameters panel. Valid numbers are 0 through 12, inclusive.

You can also use the `COMPARE EXCLUDE` command at any time to exclude all lines in a file except lines with line labels and information lines, and the lines above and below those lines. When you specify EXCLUDE without a data set name or NEXT, no comparison is done. Instead the labels and information lines that already exist in the file are used to exclude functions.

/
Can be used when you need to enter a long path name for the z/OS UNIX file to be compared against. This causes the display of a popup window containing a scrollable field for the input of a path name.

SAVE
Specifies that SuperC (which performs the actual compare function) create a listing. The listing is saved in a data set with one of these names:

- *tsopref*.ISPFEDIT.COMPARE.LIST (where *tsopref* is your TSO prefix).
- *tsopref.userid*.ISPFEDIT.COMPARE.LIST (where *userid* is your TSO user ID and it does not match your TSO prefix).
- *userid*.ISPFEDIT.COMPARE.LIST (where no TSO prefix is defined in your TSO user profile).

Note: If the ISPF configuration table keyword USE_ADDITIONAL_QUAL_FOR_PDF_DATA_SETS is set to YES, an additional qualifier defined with the ISPF_TEMPORARY_DATA_SET_QUALIFIER keyword is included before the ISPFEDIT qualifier.

The save function is intended for debugging purposes, but it also provides a way to create a SuperC listing. The listing produced is a Change listing (option CHNGL). No notification is given regarding successful creation of the listing, and errors allocating the listing do not cause the comparison to end.

Note: Because of the way the SuperC comparison is done, the file currently being edited is shown in the SuperC listing as the *old* file, and the file to which the current file is being compared is listed as the *new* file. Therefore, insertions refer to lines that are *not* in the current file, and deletions refer to lines that are only in the current file.

SYSIN
Specifies not to free the ddname SYSIN before calling SuperC to compare files. This enables you to pass SuperC Process Statements to alter the comparison. No validation is done on the type of SYSIN allocation or the contents of the data set.

supercdsname
The name of a data set containing SuperC process statements.

/
: Displays the Edit Compare SYSIN specification panel where you can specify the name of a data set containing SuperC Process statements that are used for the compare. The SYSIN data set is freed at the end of the compare.

Examples

To display the Edit Compare Settings and/or Command Parameters panel:

1. On the command line, type:

```
COMPARE
```

2. Press Enter.

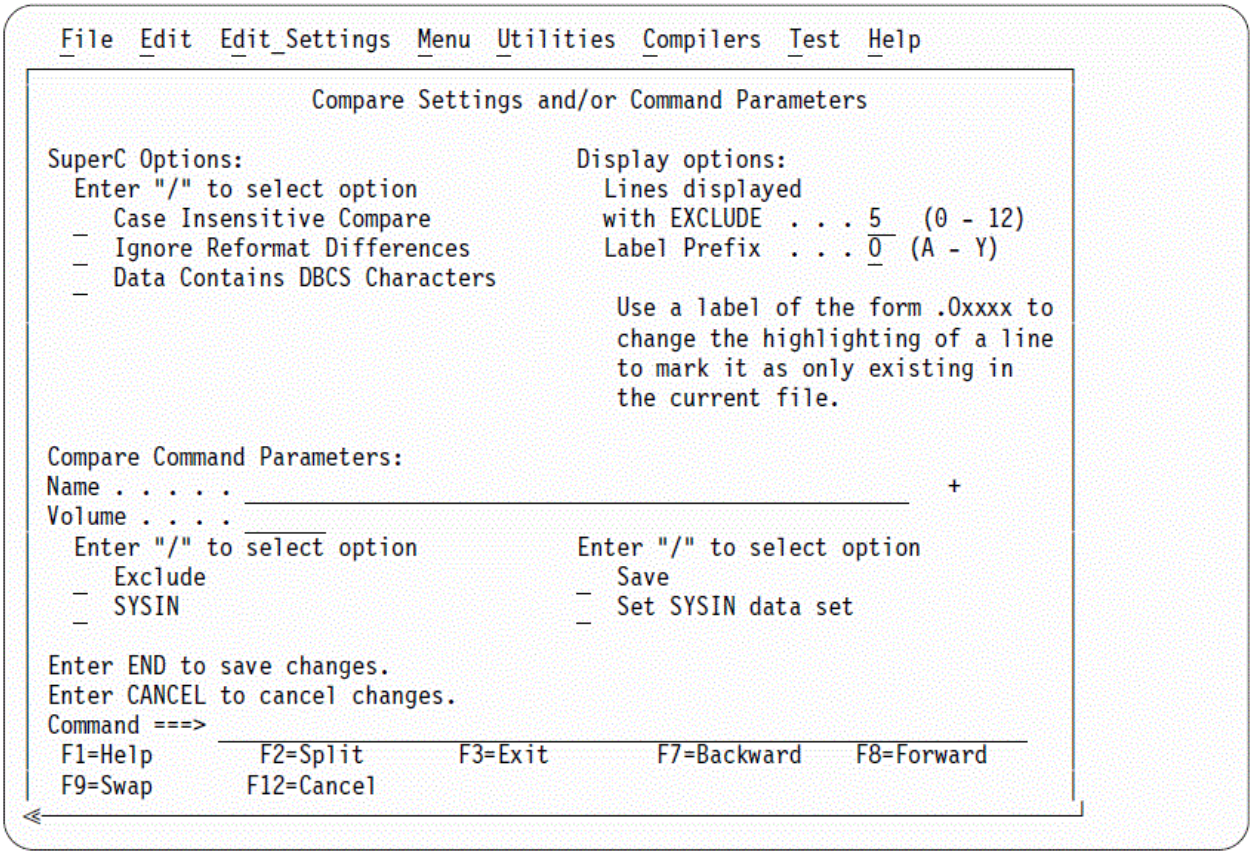


Figure 110. Edit Compare Settings and/or Command Parameters panel

To compare the data to a member in the current data set or concatenation:

1. On the command line, type:

```
COMPARE (member
```

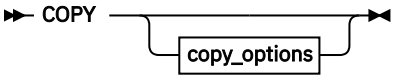
2. Press Enter.

COPY—Copy Data

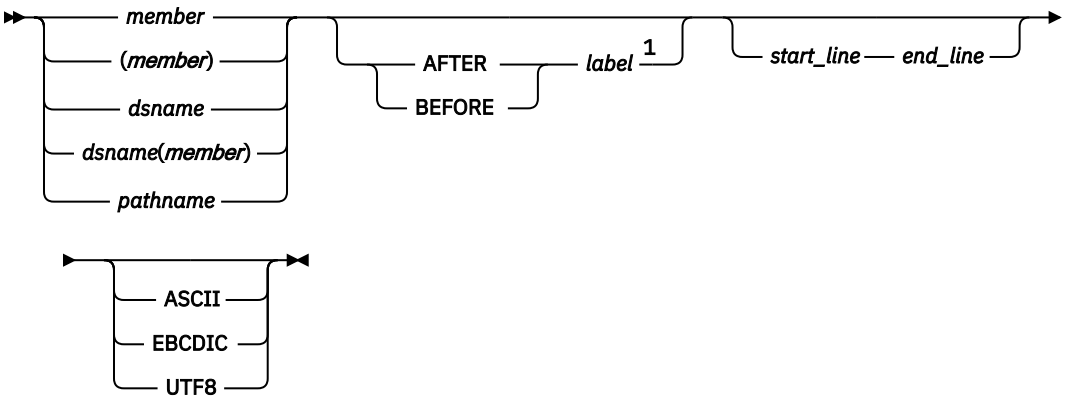
The COPY primary command copies a sequential data set, a member of a partitioned data set, or z/OS UNIX file into the data being edited.

If no options are specified with the COPY command, the Edit/View Copy panel is displayed.

Syntax



copy_options



Notes:

[1] If you don't specify the position using a label, you must specify the position by using an A or B line command.

member
A member of the ISPF library or partitioned data set that you are editing. If a name of eight or fewer characters is specified and it could be a member name or a data set name, COPY searches for a member name first. If no member is found, then the name is used as a data set name.

dsname
A partially qualified or fully qualified data set name. If the data set is partitioned you can include a member name in parentheses or select a member from a member list.

pathname
The path name for a z/OS UNIX regular file or directory. If a directory is specified, a directory selection list is displayed, allowing you to select the file to be copied. (Also, see "Specifying z/OS UNIX pathnames with edit primary and macro commands" on page 16.)

AFTER
The data is copied after the line with the specified label.

BEFORE
The data is copied before the line with the specified label.

label
Label identifying the line where the data is to be copied. It can be either a label that you define or one of the editor-defined labels, such as .ZF or .ZL.

start_line
The number of the first line of the member, data set, or z/OS UNIX file to be included in the range of lines to be copied. Must be greater than or equal to 1, and less than or equal to the number of lines in the member, data set, or z/OS UNIX file. To specify standard, ISPF, or COBOL line numbers, omit the member name, data set name, or z/OS UNIX file name to use the Extended Edit Copy panel.

end_line
The number of the last line to be included in the range of lines to be copied. Must be greater than or equal to *start_line* and less than or equal to the number of lines in the member, data set, or z/OS UNIX file.

ASCII, EBCDIC, UTF8
When one of these keywords is supplied, if the data is using a different character set to that designated by the keyword, the data being copied in from the external file is converted from the character set designated by the keyword to the character set specified for the file being edited or to the terminal character set.

The label can be either a label that you define or one of the PDF editor-defined labels, such as `.ZF` and `.ZL`.

If you have not defined a label and the ISPF editor-defined labels are not appropriate for your purpose, use the A (after) or B (before) line command to specify where the data is to be copied.

If the data set or member that you are editing is empty, you do not need to specify a destination for the data being copied.

Note: If the member name or data set name is less than 8 characters and the data set you are editing is partitioned, a like-named member is copied. If a like-named member does not exist, the name is considered to be a partially qualified data set name.

Description

COPY adds a copy of data that already exists to the data set, member, or z/OS UNIX file that you are editing. Use MOVE if you want to move data from one data set, member, or z/OS UNIX file to another, rather than just copy it.

To copy data into an empty data set, member, or z/OS UNIX file:

1. On the command line, type:

   ```
   COPY member
   ```

   or:

   ```
   COPY dsname
   ```

   or:

   ```
   COPY pathname
   ```

   The member, data set name, or path name operand is optional. If you do not specify the name of a member, data set, or z/OS UNIX file to be copied, the Edit Copy panel appears. Enter the name of the data set, member, or z/OS UNIX file on this panel.

   You can specify the numbers of the first and last lines to be copied, along with the kind of line numbers (standard, ISPFSTD, COBOL, or relative) on the Edit Copy panel. This allows you to copy only part of the data set or member.

   Note: When you select ISPFSTD line numbers and the STATS mode is ON, the editor uses the first 6 digits and ignores the 2-digit modification number. When the STATS mode is OFF, the editor uses all 8 digits.
2. Press Enter. The data is copied.

To copy data into a data set, member, or z/OS UNIX file that is *not* empty:

1. On the command line, type:

   ```
   COPY member AFTER | BEFORE label  start_line end_line
   ```

   or:

   ```
   COPY dsname AFTER | BEFORE label  start_line end_line
   ```

or:

```
COPY pathname AFTER | BEFORE label  start_line end_line
```

The *member*, *dsname*, or *pathname* operand is optional. You should omit the member name only if you do not know the member name, or if you are going to copy a sequential data set, z/OS UNIX file, or a member of a different partitioned data set.

The AFTER *label* and BEFORE *label* operands are also optional. However, if the data set, member, or z/OS UNIX file that is to receive the copied data is not empty, you must specify a destination for the copied data. Therefore, if you do not want to use a label, you can substitute either the A (after) or B (before) line command as the destination of the copied data. However, a number indicating that the A or B command should be repeated cannot follow the line command. See the descriptions of these commands for information about them.

If the data set, member, or z/OS UNIX file is not empty and you do not specify a destination, a "MOVE/COPY Pending" message appears in the upper-right corner of the panel and the data is not copied. When you type a destination and press Enter, the data is copied.

2. Press Enter. If you entered the name of a member, data set, or z/OS UNIX file, the member, data set, or z/OS UNIX file is copied. Otherwise, the edit copy panel appears. If a range of line numbers is specified, only those lines are copied. See the previous example for more information.

See “Copying and moving data” on page 41 if you need more information.

Examples

These steps show how you can copy data when you omit the member name and the ISPF editor panels appear:

1. Type COPY on the command line and specify the destination of the operation. The panel in Figure 111 on page 215 shows you that the data is to be copied after line 000700, as specified by the A (after) line command.

```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
------------------------------------------------------------------------------
EDIT       P020136.PRIVATE.PLS(INTO) - 01.00               Columns 00001 00072
Command ===> copy                                           Scroll ===> CSR
****** ***************************** Top of Data ******************************
000100
000200 $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
000300
000400 THIS IS THE MEMBER INTO WHICH THE LINES ARE TO BE COPIED.
000500
000600      +---------------------+
a 0700      |                     |
000800      |                     |
000900      |                     |
001000      |                     |
001100      +---------------------+
001200
001300 $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
001400
****** **************************** Bottom of Data ****************************


 F1=Help      F2=Split     F3=Exit      F5=Rfind     F6=Rchange   F7=Up
 F8=Down      F9=Swap     F10=Left     F11=Right    F12=Cancel
```

Figure 111. Member before data is copied

2. When you press Enter, the Edit Copy panel appears. Specify the data you want copied.

   The example in Figure 112 on page 216 copies the data set member named COPYFROM. Since you are using the Edit/View - Copy panel, you can also specify the first and last lines you want copied.

```
  Menu  RefList  Utilities  Help
_______________________________________________________________
                              Edit/View - Copy
Command ===> _____________________________________________________________
                                                              More:   -
   Project . . . PROJ1___
   Group . . . . USERID__ . . . ________ . . . ________ . . . ________
   Type  . . . . CLIST___
   Member  . . . ________       (Blank or pattern for member selection list)

From Other Partitioned or Sequential Data Set, or z/OS UNIX file:
   Data Set Name  . . ____________________________________________________
   Volume Serial  . . ______    (If not cataloged)

Data Set Password  . .          (If password protected)

Line Numbers (Blank for entire member or seq. data set)
   First line  . . . . ________
   Last line . . . . . ________
   Number type . . . . ________ (Standard, ISPFstd, COBOL, or Relative)

Data Conversion option
_   1. EBCDIC
   2. ASCII
   3. UTF-8

   Press Enter key to copy, enter End command to cancel copy.
 F1=Help      F2=Split     F3=Exit      F7=Backward  F8=Forward   F9=Swap
 F10=Actions   F12=Cancel
```

Figure 112. Edit/View - Copy panel (ISRECPY1)

3. Figure 113 on page 216 shows the contents of the COPYFROM member, which is copied into the original data set.

```
EDIT       P020136.PRIVATE.PLS(COPYFROM) - 01.00          Columns 00001 00072
****** ***************************** Top of Data ******************************
000100 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
000200 These are the lines that are to be copied.
000300  These are the lines that are to be copied.
000400 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
****** **************************** Bottom of Data ****************************
```

Figure 113. Contents of member to be copied

4. When you press Enter, the editor copies the data and displays a short message in the upper right side of the panel. Figure 114 on page 216 shows the result of the copy operation.

```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
_______________________________________________________________________
EDIT       P020136.PRIVATE.PLS(INTO) - 01.00            Member COPYFROM copied
Command ===> _____________________________________________ Scroll ===> CSR
****** ***************************** Top of Data ******************************
000100
000200 $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
000300
000400 THIS IS THE MEMBER INTO WHICH THE LINES ARE TO BE COPIED.
000500
000600      +---------------------+
000700      |                     |
000710 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
000720 These are the lines that are to be copied.
000730  These are the lines that are to be copied.
000740 @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
000800      |                     |
000900      |                     |
001000      |                     |
001100      +---------------------+
001200
001300 $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
 F1=Help      F2=Split     F3=Exit      F5=Rfind     F6=Rchange   F7=Up
 F8=Down      F9=Swap     F10=Left     F11=Right    F12=Cancel
```

Figure 114. Member after data has been copied

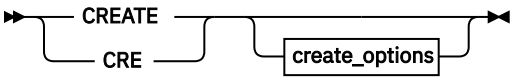
CREATE—Create Data

The CREATE primary command creates a member of a partitioned data set, a sequential data set, or z/OS UNIX file from the data you are editing.

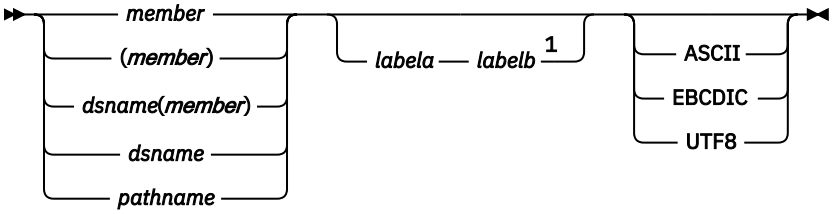
If no options are specified with the CREATE command, the Edit/View - Create panel is displayed.

Note: If you are editing a z/OS UNIX file and you issue the CREATE command to create a new z/OS UNIX file, the file permissions for the new file are set to the same values as the file permissions of the file you are currently editing. If you are editing a sequential data set or member and you issue the CREATE command to create a new z/OS UNIX file, the file permissions are set to 700 (rwx------).

Syntax



create_options



Notes:

[1] If you don't specify the group of lines using labels, you must specify the group by using C or M line commands.

member
The name of the new member added to the partitioned data set currently being edited. If you are using a concatenated sequence of libraries, the member is always written to the first library in the sequence.

labela, labelb
Labels identifying the start and end of the group of lines which are added to the new member.

For more information about using labels to identify a group of lines, see “Labels and line ranges” on page 59.

dsname(member)
The name of a different partitioned data set and new member name to be added to the partitioned data set. The data set name can be fully qualified or partially qualified.

dsname
The name of a different sequential data set to be added. The data set name can be fully qualified or partially qualified.

pathname
The path name for a z/OS UNIX regular file to be created. (Also, see “Specifying z/OS UNIX pathnames with edit primary and macro commands” on page 16.)

ASCII, EBCDIC, UTF8
When one of these keywords is supplied, if the data is using a different character set to that designated by the keyword, the data being saved in the external file is converted to the character set designated by the keyword.

Description

Note: CREATE adds a new member to a partitioned data set only if a member of the same name does not already exist. Use REPLACE if the member already exists.

To create a member of a partitioned data set, a sequential data set, or a z/OS UNIX file:

1. On the command line, type one of these commands:

```
CREATE member labela labelb
CREATE (member) labela labelb
CREATE dsname(member) labela labelb
CREATE dsname labela labelb
CREATE pathname labela labelb
```

   The *member* operand is optional unless you specify a data set name. It represents the name of the member you want to create.

   The *labela* and *labelb* operands specify the first and last lines in a group of lines used to create the new member, sequential data set, or z/OS UNIX file.

   If you omit the *labela* and *labelb* operands, you must specify the lines by using either the C (copy) or M (move) line command. See the descriptions of these commands if you need more information about them.

   If you omit the *labela* and *labelb* operands and do not enter one of the preceding line commands, a "CREATE Pending" message is displayed in the upper-right corner of the panel.

2. Press Enter. If you did not specify the name of the member, the name of another partitioned data set along with the member name, or the name of a z/OS UNIX file to be created, the Edit Create panel appears (see Figure 116 on page 219). Enter the member name on this panel and press Enter again. If you used either a pair of labels or a C line command, the data is copied from the member that you are editing into the member that you are creating. If you used the M line command, however, the data is removed from the member that you are editing and placed in the member that you are creating.

   If the data set specified does not exist, ISPF prompts you to see if the data set should be created. You can create the data set using the characteristics of the cataloged source data set as a model, or specify the characteristics for the new data set. You can suppress this function through the ISPF configuration table, causing any CREATE request for a nonexistent data set to fail.

See “Creating and replacing data” on page 41 if you need more information about the CREATE command.

Examples

These steps show how you can create a new member when you omit the member name:

1. Type CREATE on the command line and specify which lines you want to copy or move into the new data set or member. The example in Figure 115 on page 219 uses the MM (block move) line command to move a block of lines from the data.

```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
------------------------------------------------------------------------------
EDIT       USERSID.TEST(FROMDATA) - 01.00               Columns 00001 00072
Command ===> ____________________________________________ Scroll ===> PAGE
****** ***************************** Top of Data ******************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
000100 This line will be left in this member
000200 This line will be left in this member
000300 +----------------+
000400 | This is the    |
000500 | material to    |
000600 | be created in  |
000700 | another member |
000800 +----------------+
000900 This line will be left in this member
001000 This line will be left in this member
****** **************************** Bottom of Data ****************************



 F1=Help      F2=Split     F3=Exit      F5=Rfind     F6=Rchange   F7=Up
 F8=Down      F9=Swap     F10=Left     F11=Right    F12=Cancel
```

Figure 115. Member before new member is created

2. When you press Enter, the Edit/View Create panel (Figure 116 on page 219) appears. Type the name of a new member and press Enter. If you type the name of a member that already exists, an error message appears and the CREATE fails. The name of the member created for this example is TODATA.

```
  Menu  RefList  Utilities  Help
------------------------------------------------------------------------------
                               Edit/View - Create
Command ===> ____________________________________________________________
                                                                More:     +
"Current" Data Set: PDFTDEV.USERSID.MSGGEN(FLMU00)

To ISPF Library:
   Project . . . PROJ1___
   Group . . . . USERID__
   Type  . . . . CLIST___
   Member  . . . ________

To Other Sequential Data Set, Partitioned Data Set Member, or z/OS UNIX file:
   Name . . . . . TEST(TODATA)_____________________________________________  +
   Volume Serial  ______    (If not cataloged)

Data Set Password  . .           (If password protected)

Enter "/" to select option                      Data Conversion option
_  Specify pack option for "CREATE" Data Set    _  1. EBCDIC
                                                   2. ASCII
                                                   3. UTF-8

Press ENTER key to create. Enter END command to cancel create.
 F1=Help      F2=Split     F3=Exit      F7=Backward  F8=Forward   F9=Swap
 F10=Actions   F12=Cancel
```

Figure 116. Edit/View Create panel (ISRECRA1)

3. Figure 117 on page 220 shows the lines remaining in the original member after the specified lines were moved to the new member.

```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
------------------------------------------------------------------------------
EDIT       USERSID.TEST(FROMDATA) - 01.01               Member TODATA created
Command ===> ____________________________________________ Scroll ===> PAGE
****** ***************************** Top of Data ******************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
000100 This line will be left in this member
000200 This line will be left in this member
000900 This line will be left in this member
001000 This line will be left in this member
****** **************************** Bottom of Data ****************************









 F1=Help      F2=Split     F3=Exit      F5=Rfind     F6=Rchange   F7=Up
 F8=Down      F9=Swap     F10=Left     F11=Right    F12=Cancel
```

Figure 117. Member after new member has been created

4. Figure 118 on page 220 shows the contents of the new member. The data is renumbered only if both number mode and autonum mode are on. A source listing of the data is also recorded in the ISPF list data set for eventual printing if autolist mode is on. In this example, the lines have retained their original line numbers.

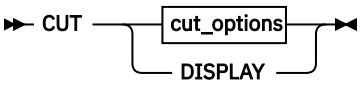
```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
------------------------------------------------------------------------------
EDIT       USERSID.TEST(TODATA) - 01.00                 Columns 00001 00072
Command ===> ____________________________________________ Scroll ===> PAGE
****** ***************************** Top of Data ******************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
000300 +----------------+
000400 | This is the    |
000500 | material to    |
000600 | be created in  |
000700 | another member |
000800 +----------------+
****** **************************** Bottom of Data ****************************









 F1=Help      F2=Split     F3=Exit      F5=Rfind     F6=Rchange   F7=Up
 F8=Down      F9=Swap     F10=Left     F11=Right    F12=Cancel
```

Figure 118. New member created

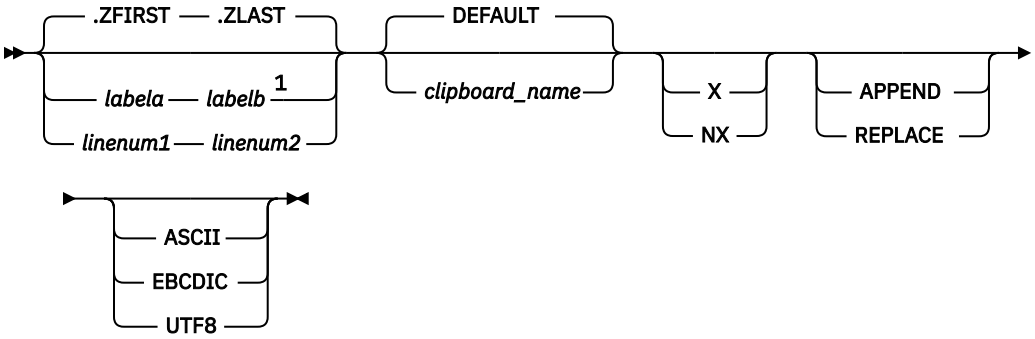
CUT—Cut and Save Lines

The CUT primary command saves lines to one of 11 named clipboards for later retrieval by the PASTE command. The lines can be appended to lines already saved by a previous CUT command or can replace existing lines in a clipboard.

Syntax



cut_options



Notes:

[1] You can also specify the group of lines using C or M line commands.

labela*, *labelb
Labels identifying the start and end of the group of lines the CUT command is to copy or move to the clipboard.

For more information about using labels to identify a group of lines, see "Labels and line ranges" on page 59.

linenum1*, *linenum2
Relative line numbers identifying the start and end of the group of lines the CUT command is to copy or move to the clipboard.

clipboard_name
The name of the clipboard to use. If you omit this parameter, the ISPF default clipboard (named DEFAULT) is used. You can define up to ten additional clipboards. The size of the clipboards and number of clipboards might be limited by installation defaults.

X
Cut only lines that are excluded from the display.

NX
Cut only lines that are not excluded from the display.

REPLACE
Replace existing data in the clipboard.

You can select REPLACE as the default by entering the EDITSET command on the editor command line. The default action depends on the setting specified in the panel displayed by the EDITSET. You should always specify REPLACE (or APPEND) in a macro because the user can change the default behavior.

APPEND
Add the data to the clipboard. You can select APPEND as the default by entering the EDITSET command on the editor command line. The default action depends on the setting specified in the panel displayed by the EDITSET. You should always specify APPEND (or REPLACE) in a macro because the user can change the default behavior.

DISPLAY
Show a list of existing clipboards. From this list you can browse, edit, clear, or rename the clipboards.

ASCII, EBCDIC, UTF8
When one of these keywords is supplied, if the data is using a different character set to that designated by the keyword, the data being placed in the clipboard is converted to the character set designated by the keyword and tagged as being in the designated character set.

Description

CUT saves copies of lines from an edit session to a clipboard for later retrieval by the PASTE command. The lines are moved or copied from the session to the named clipboard. Lines are specified by either the C (Copy) or M (Move) line commands, CC or MM block line commands, or label names. If the C or CC line commands or labels are used to identify the lines, the lines are *copied* to the clipboard. If the M or MM line commands are used to identify the lines, the lines are copied to the clipboard and deleted from the edit session (in effect, *moving* them).

All lines in the edit session are copied to the clipboard if you do not specify the lines using a label range on the CUT command, or through the C or M commands.

If you specify a clipboard name, lines are copied to that clipboard. If the specified clipboard does not yet exist, it is created. ISPF provides a default clipboard named DEFAULT. You can use up to 10 other clipboards that you define. The defined clipboards exist as long as you are logged on to TSO and are deleted when you log off.

To browse, edit, clear, or rename any of the clipboards, use the DISPLAY keyword of the CUT command:

```
CUT DISPLAY
```

Examples

This command saves to the default clipboard all the lines in the current file from the current cursor position to the last line. These lines are appended to any lines that are already in the clipboard:

```
CUT .ZCSR .ZLAST APPEND
```

To save all the lines in the current file to a clipboard named USERC1, replacing any lines already in the clipboard:

```
CUT .ZFIRST .ZLAST USERC1 REPLACE
```

This example assumes that you have APPEND set as the default behavior in the EDITSET command panel. Because all lines are copied by default, in this case you could omit the labels .ZFIRST and .ZLAST.

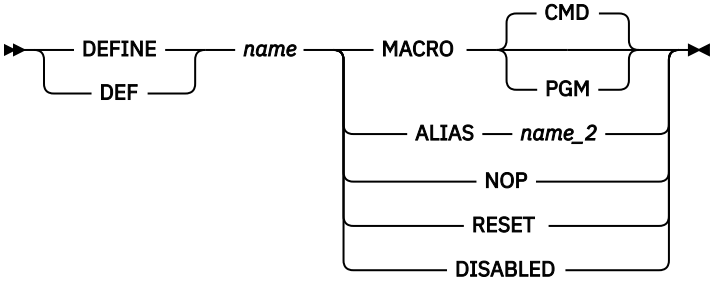
DEFINE—Define a Name

The DEFINE primary command is used to:

- Identify a macro that replaces a built-in command of the same name
- Identify programs that are edit macros
- Assign an alias to a macro or built-in command
- Make a macro or built-in command inoperable
- Reset an inoperable macro or built-in command
- Disable a macro or built-in command

DEFINE is often used with the BUILTIN command.

Syntax



name
The name for the command.

MACRO CMD
Identifies the name you are defining as a command language (CLIST or REXX exec) macro, which is called in the same way as using the SELECT service CMD keyword with a percent symbol (%) preceding the command. That means that you can specify only CLISTs or REXX EXECs.

MACRO PGM
Identifies the name that you are defining as a program (load module) macro.

ALIAS *name_2*
Identifies the name you are defining as an alias of another name, with the same characteristics. If *name_2* is already an alias, the editor replaces it with the command for which it is an alias. Therefore, it is not possible to have an alias of an alias.

NOP
Makes the name that you are defining and all of its aliases inoperable until you reset them with RESET. Therefore, when the name or an alias of the name is called, nothing is processed. NOP is similar to DISABLED, except that disabled names cannot be reset by the RESET operand.

RESET
Resets the most recent definition of the name that you are defining to the status in effect before that definition. For example, RESET makes inoperable names operable again.

DISABLED
Disables the name you are defining and all of its aliases until you completely exit the editor and return to the ISPF Primary Option Menu. Therefore, when the name or an alias of the name is entered, nothing is processed. A disabled command or macro cannot be restored by the RESET operand. To disable RESET, use delimiters around `'RESET'` to distinguish it from the keyword.

Description

The effects of a DEFINE command remain until you either issue DEFINE RESET or exit from the editor. You enter the editor when you select option 2, and you do not exit the editor until you return to the ISPF Primary Option Menu. Therefore, if you edit several members of a partitioned data set, one DEFINE at the beginning affects them all.

To temporarily override the DEFINE command, use the BUILTIN command.

Stacking DEFINE commands

Except for the DISABLED operand, the DEFINE operations are stacked. The RESET operand unstacks them. For example:

```
DEFINE A alias FIND
DEFINE A alias COPY
DEFINE A alias SAVE
```

stacks three definitions of A. Only the last one is effective. Here, A would be defined as SAVE.

This operation:

```
DEFINE A RESET
```

removes one command from the stack, making the previous command effective. In the preceding example, A would now be defined as COPY.

Examples

To define the name IJKDOIT as a CLIST or REXX macro, enter:

```
DEFINE IJKDOIT MACRO
```

To define the name SETITUP as a program macro, enter:

```
DEFINE SETITUP MACRO PGM
```

To define the name DOIT as an alias of the macro IJKDOIT, enter:

```
DEFINE DOIT ALIAS IJKDOIT
```

To define the name SAVE to have no effect, enter:

```
DEFINE SAVE NOP
```

To reset the definition of the name SAVE, enter:

```
DEFINE SAVE RESET
```

To define the name FINDIT as disabled, enter:

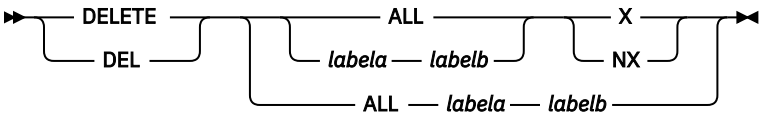
```
DEFINE FINDIT DISABLED
```

DELETE—Delete Lines

The DELETE primary command deletes lines from the data you are editing.

Note: As a precaution against error, there is no DELETE ALL command. To delete all lines, see “Description” on page 225.

Syntax



ALL
Specifies that all selected lines are deleted. The DELETE command, unlike FIND, CHANGE, and EXCLUDE, does not accept NEXT, FIRST, PREV, or LAST. ALL is required to emphasize that NEXT is not the default.

X
Restricts the lines deleted to those that are excluded.

NX
Restricts the lines deleted to those that are not excluded.

labela*, *labelb
Labels identifying the start and end of the group of lines which are deleted, including the lines with the labels. To delete one line, enter the same label twice.

For more information about using labels to identify a group of lines, see "Labels and line ranges" on page 59.

Description

To delete all lines, do one of these:

- To delete all lines by using the editor-defined labels:

```
DELETE ALL .ZFIRST .ZLAST
```

- To delete all lines by first resetting any excluded lines to make them not excluded, and then deleting all lines that are not excluded:

```
RESET; DELETE ALL NX
```

Examples

In the examples that follow, *.labela* and *.labelb* represent the two labels that show the range of lines to be deleted.

- To delete all excluded lines:

```
DELETE ALL X
```

- To delete all not excluded lines:

```
DELETE ALL NX
```

- To delete all excluded lines within a range:

```
DELETE .labela .labelb X
```

- To delete all not excluded lines within a range:

```
DELETE .labela .labelb NX
```

- To delete all lines within a range:

```
DELETE .labela .labelb
```

You can more easily determine which lines to delete in a large data set by excluding lines that meet some criterion, or by leaving all lines that meet the criterion non-excluded. Then, with DELETE you can delete many lines. For example, to delete all blank lines in a data set, type these commands on the command line and press Enter after each one:

1. First, reset all excluded lines:

```
RESET X
```

2. Then, exclude lines containing characters that are not blanks:

```
EXCLUDE ALL P'¬'
```

3. Finally, delete the non-excluded lines, which contain only blanks:

```
DEL ALL NX
```

Another way to do the same thing is this:

1. First, exclude all lines:

```
EXCLUDE ALL
```

2. Then, find all lines containing a character that is not a blank:

```
FIND ALL P'¬'
```

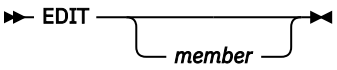
3. Finally, delete the remaining excluded lines, which contain only blanks:

```
DEL ALL X
```

EDIT—Edit from within an Edit Session

The EDIT primary command allows you to edit another sequential data set, partitioned data set member, or z/OS UNIX file during your current edit session.

Syntax



member
A member of the ISPF library or other partitioned data set you are currently editing. You may enter a member pattern to generate a member list.

Description

Editing one data set or member while you are already editing another is called *recursive editing*. To edit another data set, member, or z/OS UNIX file during your current edit session:

1. On the command line, type:

```
EDIT
```

or

```
EDIT member
```

Here, *member* represents the name of a member of the partitioned data set you are editing. The member operand is optional.

2. Press Enter.

If you specified a member name, the current library concatenation sequence finds the member. The member is displayed for editing.

If you do not specify a member name, the Edit Command Entry panel, which is identical to the regular Edit Entry panel, appears. You can enter the name of any sequential, partitioned data set, or z/OS UNIX file to which you have access. When you press Enter, the data set, member, or z/OS UNIX file is displayed for editing.

The editor suspends your initial edit session until the second-level edit session is complete. Editing sessions can be nested until you run out of storage.

3. To exit from a nested edit session, enter an END or CANCEL command. The current edit session resumes.

Examples

These steps show the use of the EDIT primary command:

1. Assume that you are editing a member named @INDEX and you need to edit a member in another data set. So, you enter the EDIT command on the command line, omitting the member operand, as shown in Figure 119 on page 227.

```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
──────────────────────────────────────────────────────────────────
EDIT       ISP.SISPSAMP(@INDEX) - 01.00                Columns 00001 00072
Command ===> edit                                       Scroll ===> PAGE
000302            that file.
000303 ISRONLY  - Sample Edit Macro
000304 ************************************************************************
000305 * PDF Sample Programs for Creating Translate Tables
000306 ************************************************************************
000307 ISRAPLTT - A sample assembler module for creating your own set of
000308            translate tables.  It contains the values for a 3278/3279
000309            APL English terminal.
000310 ISROWNTT - A sample assembler module for creating your own set of
000311            translate tables.  It contains the values for a 3278/3279
000312            English terminal.
000313 ************************************************************************
000314 * PDF Samples for Programming Languages
000315 ************************************************************************
000316 ISRASM   - Sample assembler program
000317 ISRCOBOL - Sample cobol program
000318 ISRFORT  - Fortran test program
000319 ISRPLI   - Sample PL/I program
 F1=Help      F2=Split     F3=Exit      F5=Rfind     F6=Rchange   F7=Up
 F8=Down      F9=Swap     F10=Left     F11=Right    F12=Cancel
```

Figure 119. EDIT primary command example

2. When you press Enter, the Edit Command Entry panel (Figure 120 on page 227) appears. On this panel, you enter the name of the partitioned data set and member that you want to edit:

```
  Menu  RefList  RefMode  Utilities  Workstation  Help
──────────────────────────────────────────────────────────
                        Edit Command - Entry Panel
Command ===>

ISPF Library:
   Project . . . PDFTDEV
   Group . . . . USERSID  . . . ________ . . . ________ . . . ________
   Type  . . . . MSGGEN
   Member  . . . ________      (Blank or pattern for member selection list)

Other Partitioned, Sequential or VSAM Data Set, or z/OS UNIX file:
   Name . . . . . 'ISP.SISPSAMP(ISRBOX)'                                  +
   Volume Serial  ______    (If not cataloged)

Workstation File:
   File Name  . . ____________________________________________________________

                                 Options
Initial Macro  . . . . ________   /  Confirm Cancel/Move/Replace
Profile Name . . . . . ________   _  Mixed Mode
Format Name  . . . . . ________   _  Edit on Workstation
Data Set Password  . . _____      _  Preserve VB record length
Record Length  . . . .
Line Command Table . . ________
 F1=Help      F2=Split     F3=Exit      F4=Expand    F7=Backward  F8=Forward
 F9=Swap     F10=Actions  F12=Cancel
```

Figure 120. Edit Command Entry panel (ISREDM03)

3. When you press Enter again, the member is displayed for editing, as shown in Figure 121 on page 228:

```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
-------------------------------------------------------------------------------
EDIT       ISP.SISPSAMP(ISRBOX) - 01.00                  Columns 00001 00072
Command ===> _____________________________________________ Scroll ===> PAGE
****** ***************************** Top of Data ******************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
==MSG> -CAUTION- Profile is set to STATS ON. Statistics did not exist for
==MSG>           this member, but will be generated if data is saved.
000001 /*********************************************************************/
000002 /*                                                                   */
000003 /* 5694-A01 (C) COPYRIGHT IBM CORP 1995, 2004                        */
000004 /*                                                                   */
000005 /* ISRBOX - Draw a box with its upper left corner at the             */
000006 /*          cursor position                                          */
000007 /*                                                                   */
000008 /*********************************************************************/
000009 ISREDIT MACRO
000010 ISREDIT (ROW,COL) = CURSOR             /* Get cursor position     */
000011
000012 ISPEXEC CONTROL ERRORS RETURN          /* No macro error panel    */
000013                                       /* Draw box over existing  */
 F1=Help      F2=Split     F3=Exit      F5=Rfind     F6=Rchange   F7=Up
 F8=Down      F9=Swap     F10=Left     F11=Right    F12=Cancel
```

Figure 121. Nested member editing example

EDITSET—Display the Editor Settings Dialog

The EDITSET and EDSET primary commands cause the Editor Settings dialog to begin, enabling you to modify Editor settings.

Syntax

Description

The EDITSET primary command enables you to modify the Editor settings.

The Edit and View Settings panel

Entering the EDITSET or EDSET primary commands, or choosing the Edit_Settings action bar item causes the panel shown in Figure 122 on page 229 to display:

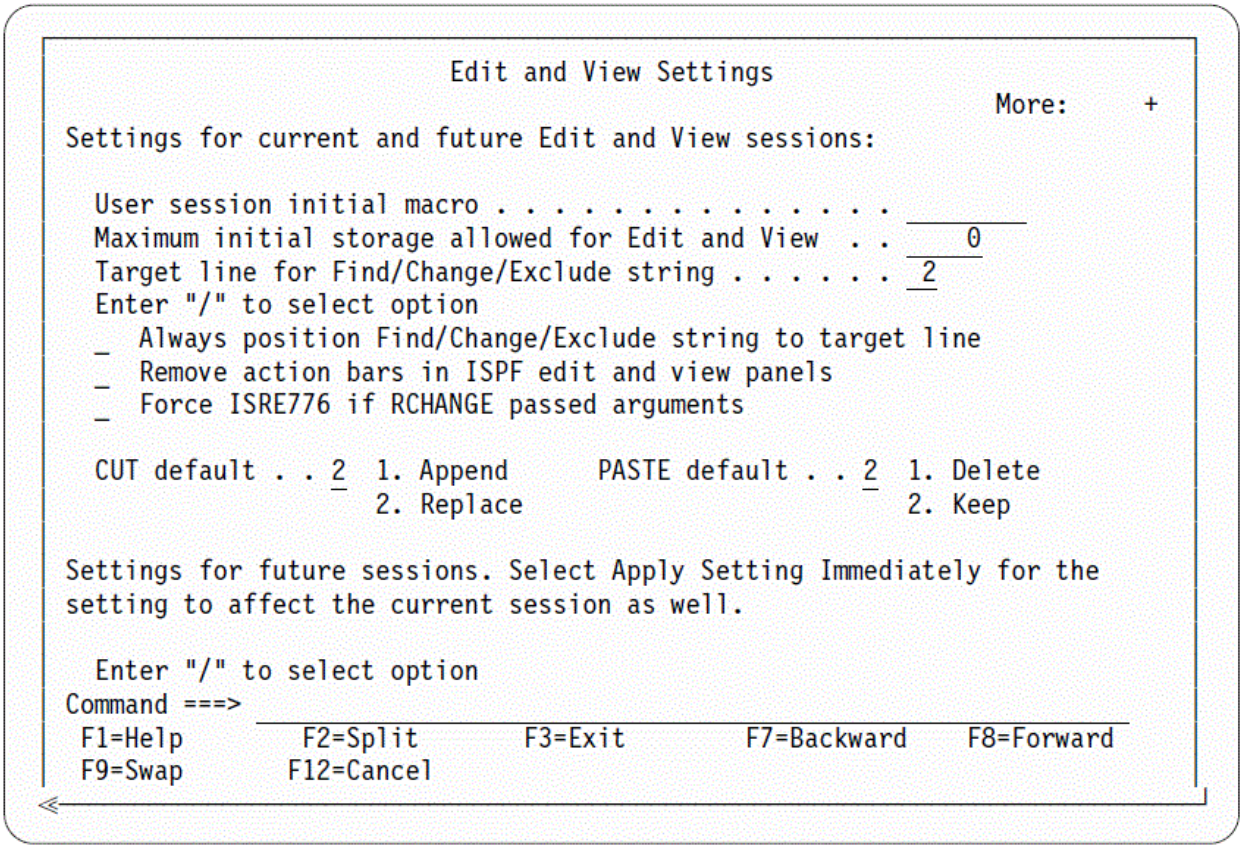


Figure 122. Edit and View Settings panel (ISREDSET)

The fields on the panel are as follows:

User session initial macro
You can specify a macro to be run before you begin editing your sequential data set or any member of a partitioned data set. This initial macro allows you to set up a particular editing environment for the Edit session you are beginning. This initial macro runs in addition to any IMACRO value in your profile.

Maximum initial storage allowed for Edit and View
The maximum amount of storage that edit and view use when initially loading the data into the edit or view session. This number is in kilobytes and is rounded to the nearest 128 KB value. If you set a limit on the initial amount of storage allowed, and a session requires more than that amount, the data is shown in BROWSE mode instead of edit or view.

A value of zero indicates that the edit session should not impose any limits on initial storage used. If this value is zero and there is not enough storage to load the data, a program error can result.

Target line for Find/Change/Exclude string
This indicates the line of the edit data display to which the target line of a FIND, CHANGE, or EXCLUDE command should be positioned. The value can be from 1 to 99, the default is 2. If the value specified is greater than the last line of the display, the target line is positioned to the last line of the display.

Always position Find/Change/Exclude string to target line
This determines whether the editor always positions the target line of a FIND, CHANGE, or EXCLUDE command to the target line specified in the "Target line for found/changed/excluded string" field, or only position the string if it is not currently on the display. The default is to only position the line if it is not on the current display.

Remove action bars in ISPF edit and view panels
If this field is selected, the action bars in the edit or view panels are not shown. This field affects only those panels that are included in ISPF, and has no effect on customized edit panels or edit panels provided with products other than ISPF.

Force ISRE776 if RCHANGE passed arguments
If this field is selected then EDIT will ensure that when RCHANGE is issued from a PF key, it does not try to process input from the command line. In this case RCHANGE will treat anything that you type on the command line as an invalid parameter and will return an error message ISRE776. For more information, see “Edit commands and PF key processing” on page 14.

CUT default

Append
If data exists on the clipboard, append the new data being cut to the end of the existing data.

Replace
If data exists on the clipboard, replace it with the new data being cut.

PASTE default

Delete
Remove the data from the clipboard after it has been pasted.

Keep
Do not remove the data from the clipboard after it has been pasted. This allows for data to be pasted multiple times.

Confirm Cancel/Move/Replace
When you select this field with a "/", a confirmation panel displays when you request one of these actions, and the execution of that action would result in data changes being lost or existing data being overwritten.

- For MOVE, the confirm panel is displayed if the data to be moved exists. Otherwise, an error message is displayed.
- For REPLACE, the confirm panel is displayed if the data to be replaced exists. Otherwise, the REPLACE command functions like the edit CREATE command, and no confirmation panel is displayed.
- For CANCEL, the confirmation panel is displayed if any data changes have been made, whether through primary commands, line commands, or typing.

  Note: Any commands or data changes pending at the time the CANCEL command is issued are ignored. Data changes are "pending" if changes have been made to the displayed edit data, but no interaction with the host (ENTER, PF key, or command other than CANCEL) has occurred. If no other changes have been made during the edit session up to that point, the confirmation panel is not displayed.

Apply Setting Immediately
Controls whether a change in the setting applies to the current edit session (immediately) or on the next edit session.

Preserve VB record length
You can select this option to cause the editor to store the original length of each record in variable-length data sets and when a record is saved, the original record length is used as the minimum length for the record.

Apply Setting Immediately
Controls whether a change in the setting applies to the current edit session (immediately) or on the next edit session.

Examples

These steps show the use of the EDITSET primary command:

1. Assume that you are editing a member named PGM8 and you want to change the setting for Confirming a Cancel, Move, or Replace action. So, you enter the EDITSET command on the command line as shown in Figure 123 on page 231.

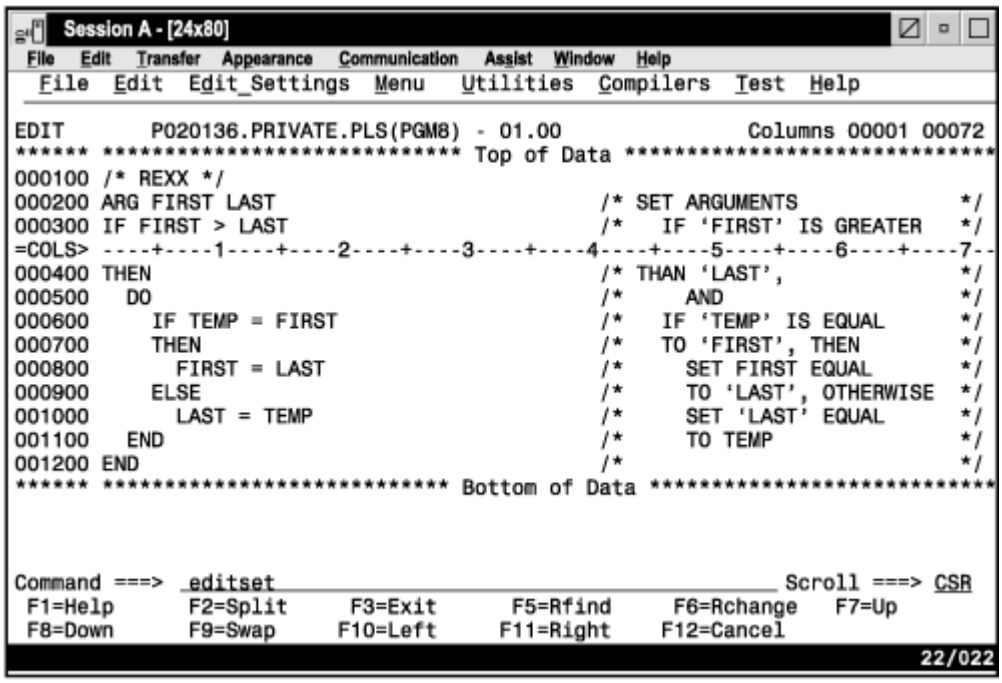


Figure 123. EDITSET primary command example

2. When you press Enter, the Edit and View Settings panel (Figure 122 on page 229) appears.
3. If necessary, scroll down to display the Confirm Cancel/Move/Replace field. Enter or remove the slash mark in the Confirm Cancel/Move/Replace field to make the setting as you want it to be.

END—End the Edit Session

The END primary command ends the editing of the current sequential data set or partitioned data set member.

Syntax

Description

To end an edit session by using END, either:

- Enter END on the command line, or
- Press a function key to which END is assigned. The default setting is F3

If no aliases have been defined for END, the editor's response to END depends on:

- Whether changes were made to the data during your current edit session
- If changes were made, whether SAVE was entered after the last change
- The setting of number mode, autonum mode, stats mode, autolist mode, and autosave mode in the edit profile
- Whether you were editing a member that was an alias of another member

For additional explanation, see “Ending an edit session” on page 12.

Examples

To end the current edit session:

1. On the command line, type:

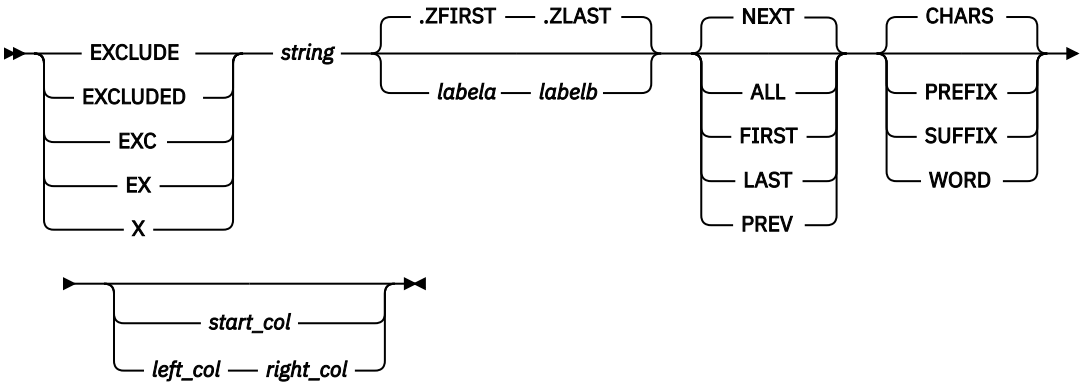
```
END
```

2. Press Enter.

EXCLUDE—Exclude Lines from the Display

The EXCLUDE primary command hides lines that contain a search string from view and replaces them with a dashed line. To see the lines again, you enter either the FLIP, RESET or RESET EXCLUDED command.

Syntax



string
The search string you want to exclude. See "Finding, seeking, changing, and excluding data" on page 44.

labela*, *labelb
Labels identifying the start and end of the group of lines which the EXCLUDE command is to search.

For more information about using labels to identify a group of lines, see "Labels and line ranges" on page 59.

NEXT
Starts at the first position after the current cursor location and searches ahead to find the next occurrence of string.

ALL
Starts at the top of the data and searches ahead to find all occurrences of string.

FIRST
Starts at the top of the data and searches ahead to find the first occurrence of string.

LAST
Starts at the bottom of the data and searches backward to find the last occurrence of string.

PREV
Starts at the current cursor location and searches backward to find the previous occurrence of string.

CHARS
Locates string anywhere the characters match.

PREFIX
Locates string at the beginning of a word.

SUFFIX
Locates string at the end of a word.

WORD
String is delimited on both sides by blanks or other non-alphanumeric characters.

start_col
The first column to be included in the range of columns to be searched. When you specify only one column, the editor finds the string only if the string starts in the specified column.

left_col
Number of the first column the EXCLUDE command is to search.

right_col
Number of the last column the EXCLUDE command is to search.

Note:

1. For more information about restricting the search to only a portion of each line, see "Limiting the search to specified columns" on page 54.
2. The EXCLUDE command allows you to control the starting point and the direction of the search by positioning the cursor and using either the NEXT or PREV operand. For more information, see "Starting point and direction of the search" on page 52.

Description

You can use the EXCLUDE command with the FIND and CHANGE commands to find a search string, change it, and exclude the line that contains the string from the panel.

To exclude the next non-excluded line that contains the letters ELSE without specifying any other qualifications:

1. On the command line, type:

```
EXCLUDE ELSE
```

2. Press Enter. Since no other qualifications were specified, the letters ELSE can be:
   - Uppercase or a mixture of uppercase and lowercase
   - At the beginning of a word (prefix), the end of a word (suffix), or the entire word (word)
   - Anywhere within the current boundaries

To exclude the next line that contains the letters ELSE, but only if the letters are uppercase:

1. On the command line, type:

```
EXCLUDE C'ELSE'
```

2. Press Enter. This type of exclusion is called a character string exclusion (note the C that precedes the search string) because it excludes the next line that contains the letters ELSE only if the letters are found in uppercase. However, since no other qualifications were specified, the exclusion occurs no matter where the letters are found on a non-excluded line, as outlined in the previous list.

For more information, including other types of search strings, see "Finding, seeking, changing, and excluding data" on page 44.

Examples

The example shown here excludes the first non-excluded line in the data set that contains the letters ELSE. However, the letters must occur on or between lines labeled .E and .S and they must be the first four letters of a word:

```
EXCLUDE ELSE .E .S FIRST PREFIX
```

The example shown here excludes the last non-excluded line in the data set that contains the letters ELSE. However, the letters must occur on or between lines labeled .E and .S and they must be the last four letters of a word.

```
EXCLUDE ELSE .E .S LAST SUFFIX
```

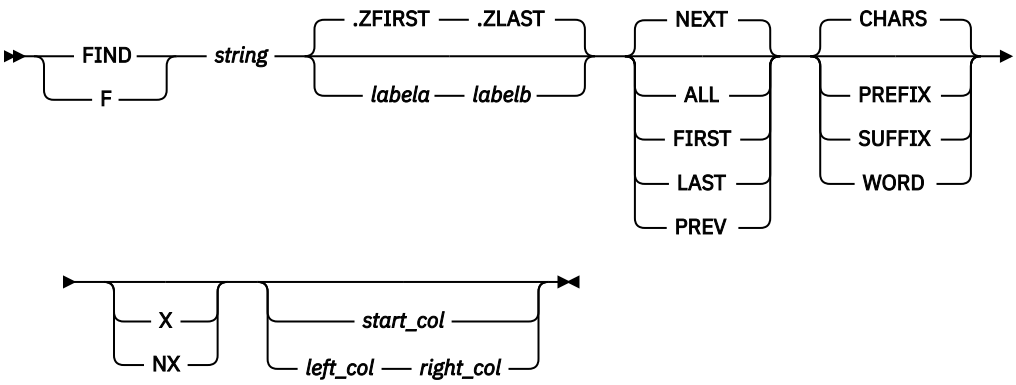
The example shown here excludes the first non-excluded line that immediately precedes the cursor position and that contains the letters ELSE. However, the cursor must not be positioned ahead of the lines labeled .E and .S. Also, the letters must occur on or between lines labeled .E and .S; they must be standalone characters (not part of any other word); and they must exist within columns 1 and 5:

```
EXCLUDE ELSE .E .S PREV WORD 1 5
```

FIND—Find a Data String

The FIND primary command locates one or more occurrences of a search string.

Syntax



string
The search string you want to find. See “Finding, seeking, changing, and excluding data” on page 44.

labela*, *labelb
Labels identifying the start and end of the group of lines which FIND is to search.

For more information about using labels to identify a group of lines, see “Labels and line ranges” on page 59.

NEXT
Starts at the first position after the current cursor location and searches ahead to find the next occurrence of string.

ALL
Starts at the top of the data and searches ahead to find all occurrences of string.

FIRST
Starts at the top of the data and searches ahead to find the first occurrence of string.

LAST
Starts at the bottom of the data and searches backward to find the last occurrence of string.

PREV
Starts at the current cursor location and searches backward to find the previous occurrence of string.

CHARS
Locates string anywhere the characters match.

PREFIX
Locates string at the beginning of a word.

SUFFIX
Locates string at the end of a word.

WORD
String is delimited on both sides by blanks or other non-alphanumeric characters.

X
Scans only lines that are excluded from the display.

NX
Scans only lines that are not excluded from the display.

start_col
The first column to be included in the range of columns to be searched. When you specify only one column, the editor finds the string only if the string starts in the specified column.

left_col
Number of the first column the FIND command is to search.

right_col
Number of the last column the FIND command is to search.

Note:

1. For more information about restricting the search to only a portion of each line, see "Limiting the search to specified columns" on page 54.
2. The FIND command allows you to control the starting point and the direction of the search by positioning the cursor and using either the NEXT or PREV operand. For more information, see "Starting point and direction of the search" on page 52.

Description

You can use the FIND command with the EXCLUDE and CHANGE commands to find a search string, change it, and exclude the line that contains the string from the panel.

To find the next occurrence of the letters ELSE without specifying any other qualifications:

1. On the command line, type:

```
FIND ELSE
```

2. Press Enter. Since no other qualifications were specified, the letters ELSE can be:
   - Uppercase or a mixture of uppercase and lowercase
   - At the beginning of a word (prefix), the end of a word (suffix), or the entire word (word)
   - In either an excluded or a non-excluded line
   - Anywhere within the current boundaries

To find the next occurrence of the letters ELSE, but only if the letters are uppercase:

1. On the command line, type:

```
FIND C'ELSE'
```

2. Press Enter. This type of search is called a character string search (note the C that precedes the search string) because it finds the next occurrence of the letters ELSE only if the letters are in uppercase. However, since no other qualifications were specified, the letters can be found anywhere in the data set or member, as outlined in the preceding list.

For more information, including other types of search strings, see "Finding, seeking, changing, and excluding data" on page 44.

Examples

The example shown here finds the first occurrence in the data set of the letters ELSE. However, the letters must occur on or between lines labeled .E and .S and they must be the first four letters of a word:

```
FIND ELSE .E .S FIRST PREFIX
```

The example shown here finds the last occurrence in the data set of the letters ELSE. However, the letters must occur on or between lines labeled .E and .S; they must be the last four letters of a word; and they must be found in an excluded line.

```
FIND ELSE .E .S LAST SUFFIX X
```

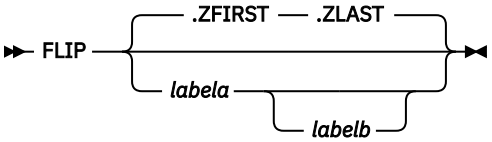
The example shown here finds the first occurrence of the letters ELSE that immediately precedes the cursor position. However, the cursor must not be positioned ahead of the lines labeled .E and .S. The letters must occur on or between lines labeled .E and .S; they must be standalone characters (not part of any other word); they must be found in a non-excluded line; and they must exist within columns 1 and 5:

```
FIND ELSE .E .S PREV WORD NX 1 5
```

FLIP—Reverse Exclude Status of Lines

The FLIP primary command reverses the exclude status of a specified group of lines or of all the lines in a file, including data, information, message, and note lines.

Syntax



labela, labelb
Labels identifying the start and end of the group of lines for which FLIP is to reverse the exclude status. If *labelb* is not supplied, then the single line identified by *labela* is flipped.

For more information about using labels to identify a group of lines, see “Labels and line ranges” on page 59.

Description

The FLIP primary command reverses the exclude status of a range of lines you specify with labels. It can also reverse the exclude status of all the lines in a file. FLIP excludes all lines that are currently visible, and makes all excluded lines visible. For example, if you have used the `'X ALL;FIND ALL xyz'` command to find lines containing a string (`xyz`), you can use FLIP to see the lines which do not contain the string.

The range is optional. If no range is specified, the exclude status is reversed for all of the lines in the file.

To reverse the exclude status of all the lines in a file:

1. Enter this command on the command line:

   ```
   FLIP
   ```

2. Press Enter.

   All the excluded lines in the file are displayed, and all the previously displayed lines are excluded.

To reverse the exclude status of a range of lines:

1. Enter this command on the command line:

   ```
   FLIP .A .B
   ```

   Actual values are substituted for .a and .b and can be defined by an edit macro or by the user.
2. Press Enter.

   All the lines with the specified range that were previously excluded are displayed, and all the lines within the specified range that were displayed are excluded.

Examples

In the example shown in Figure 124 on page 237, the edit session contains 10 lines:

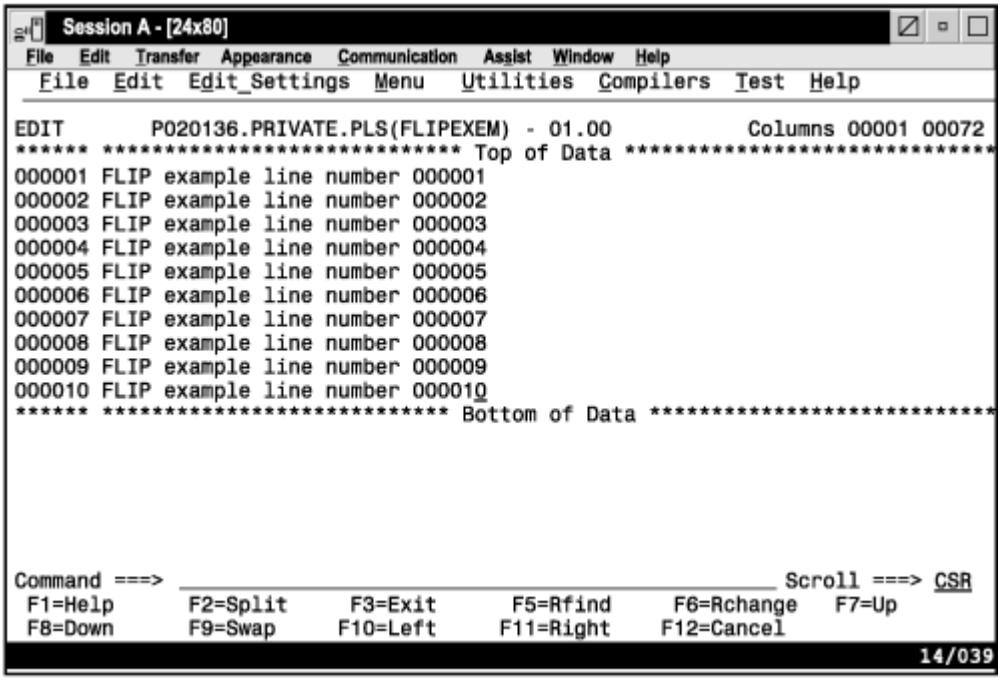


Figure 124. Example of data set

After excluding lines 4 through 7, the data set looks like Figure 125 on page 238:

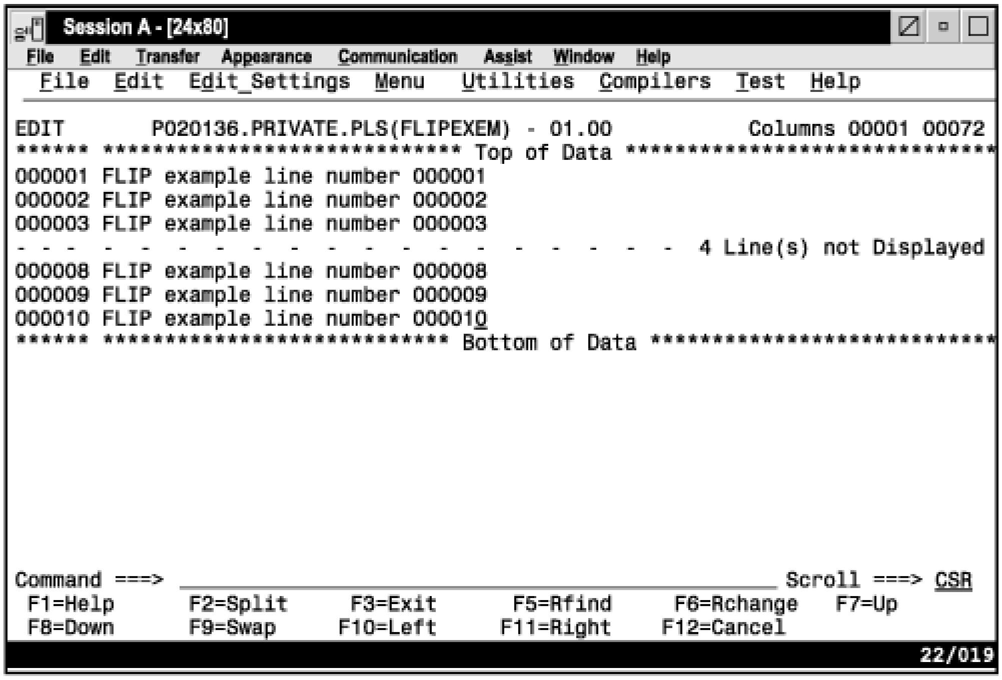


Figure 125. Example of data set with excluded lines

After executing FLIP, all previously excluded lines are shown. All previously visible lines are excluded, as shown in Figure 126 on page 238.

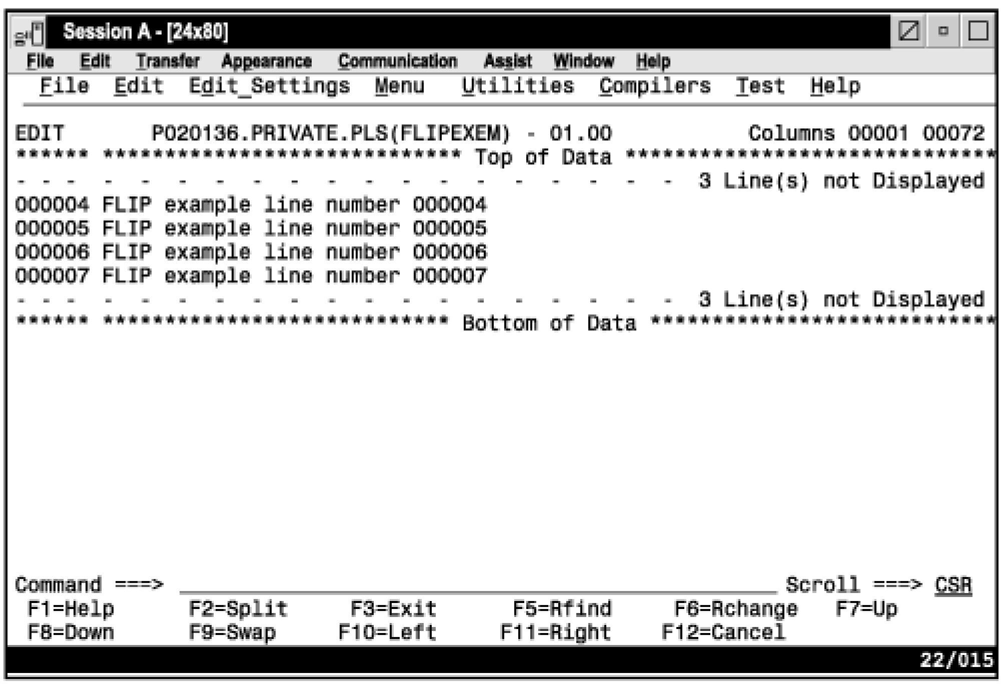
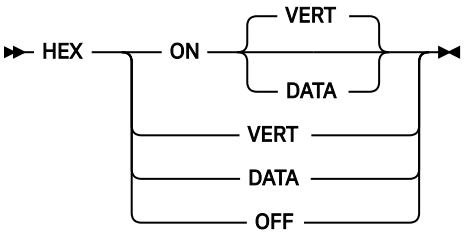


Figure 126. Example of data set using FLIP on excluded lines

HEX—Display Hexadecimal Characters

The HEX primary command sets hexadecimal mode, which determines whether data is displayed in hexadecimal format.

Syntax



ON VERT
Displays the hexadecimal representation of the data vertically (two rows per byte) under each character.

ON DATA
Displays the hexadecimal representation of the data as a string of hexadecimal characters (two per byte) under the characters.

OFF
Does not display hexadecimal representation of the data.

Note: The command, HEX OFF, cancels the effect of any previous HX or HXX commands.

Description

The HEX command determines whether the editor displays hexadecimal representation in a vertical or data string format. See Figure 128 on page 240 and Figure 129 on page 241 for examples of these two formats.

When the editor is operating in hexadecimal mode, three lines are displayed for each source line. The first line shows the data in standard character form, while the next two lines show the same data in hexadecimal representation. This applies to every line except profile lines (=PROF>), excluded line messages (- - -), message lines (==MSG>), and informational lines (======).

Besides normal editing on the first of the three lines, you can change any characters by typing over the hexadecimal representations.

You can also use the FIND, CHANGE, and EXCLUDE commands to find, change, or exclude invalid characters or any specific hexadecimal character, regardless of the setting of hexadecimal mode. See the discussion of picture strings and hexadecimal strings under “Finding, seeking, changing, and excluding data” on page 44.

Examples

Suppose you are editing the data set member shown in Figure 127 on page 240:

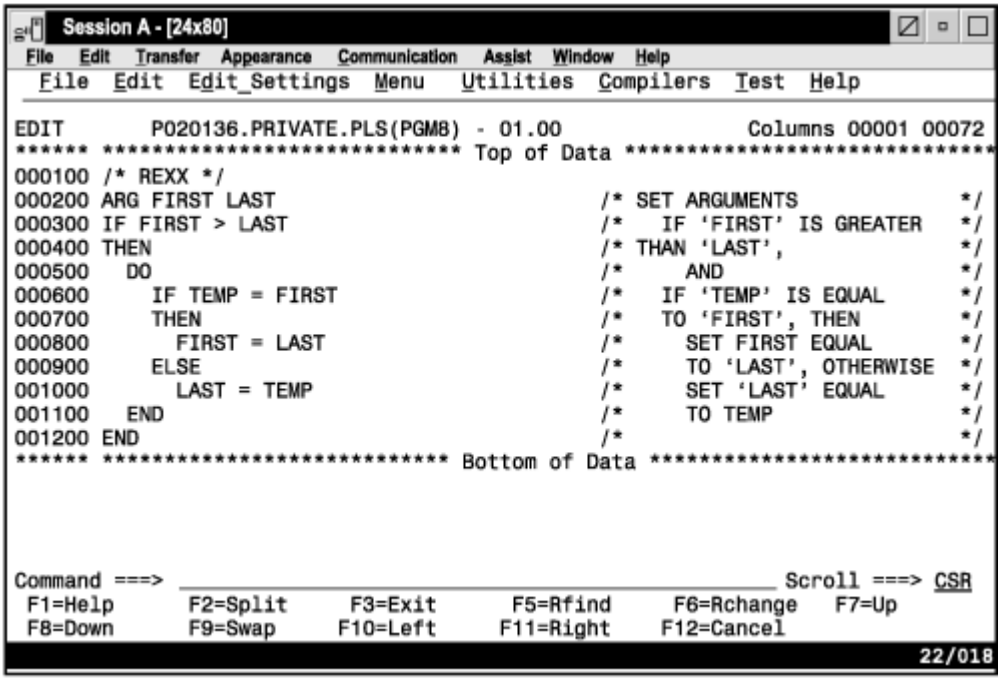


Figure 127. Member with hexadecimal mode off

Pressing Enter causes the hexadecimal value for each character on the panel, including blanks, to be displayed in vertical format, as shown in Figure 128 on page 240.

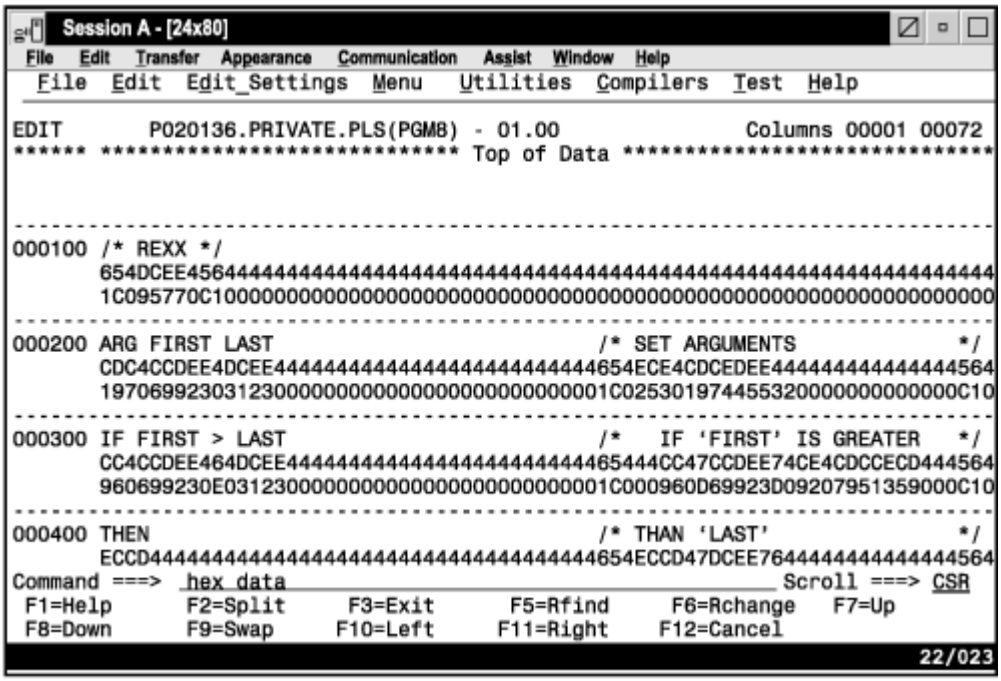


Figure 128. Hexadecimal display, vertical representation

You can enter the HEX DATA command to change the display to data format, as shown in Figure 129 on page 241.

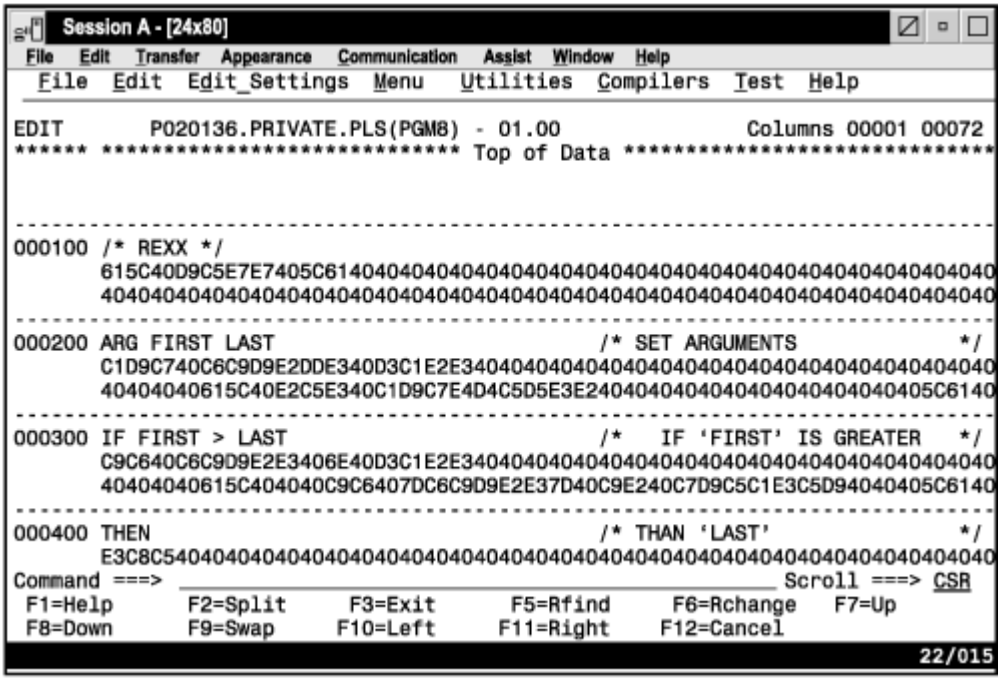


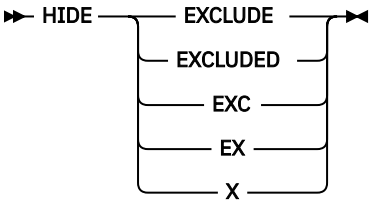
Figure 129. Hexadecimal display, data representation

HIDE—Hide Excluded Lines Message

The HIDE command removes the "*n* Line(s) not Displayed" messages from the display where lines have been excluded by the EXCLUDE command.

The HIDE function has dependencies on the value of the ISPF variable ZHIDEX, panel attributes, and extended highlighting support of the terminal. These dependencies are described in the section "Providing customized Browse and Edit panels" in the *ISPF Planning and Customisation Guide*.

Syntax



X

Removes each "*n* Line(s) not Displayed" message from the display and underscores the line number field of the preceding line.

Description

The HIDE command removes the "*n* Line(s) not Displayed" messages from the display where lines have been excluded by the EXCLUDE command. Instead the line number field of the preceding line is underscored (where the terminal supports the underscore attribute) to indicate that part of the data is not being displayed.

The RESET HIDE command redisplays the excluded lines messages.

Examples

In Figure 130 on page 242, the edit session shows that three lines are excluded after line 000020 and one line is excluded after line 000060:

```
EDIT       SBURNF.PRIVATE.DATA(HIDEXMP) - 01.01            Columns 00001 00072
****** ***************************** Top of Data ******************************
000010  example text line number 00010
000020  example text line number 00020
- - -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  3 Line(s) not Displayed
000060  example text line number 00060
- - -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  -  1 Line(s) not Displayed
000080  example text line number 00080
000090  example text line number 00090
000100  example text line number 00100
****** **************************** Bottom of Data ****************************









Command ===> hide x                                          Scroll ===> CSR
```

Figure 130. Before the HIDE primary command

Figure 131 on page 242 shows the edit session after the HIDE X command is entered. Note that the line number fields for lines 000020 and 000060 are underscored.

```
EDIT       SBURNF.PRIVATE.DATA(HIDEXMP) - 01.01            Columns 00001 00072
****** ***************************** Top of Data ******************************
000010  example text line number 00010
000020  example text line number 00020
000060  example text line number 00060
000080  example text line number 00080
000090  example text line number 00090
000100  example text line number 00100
****** **************************** Bottom of Data ****************************









Command ===>                                                 Scroll ===> CSR
```

Figure 131. After the HIDE primary command

HILITE—Enhanced Edit Coloring

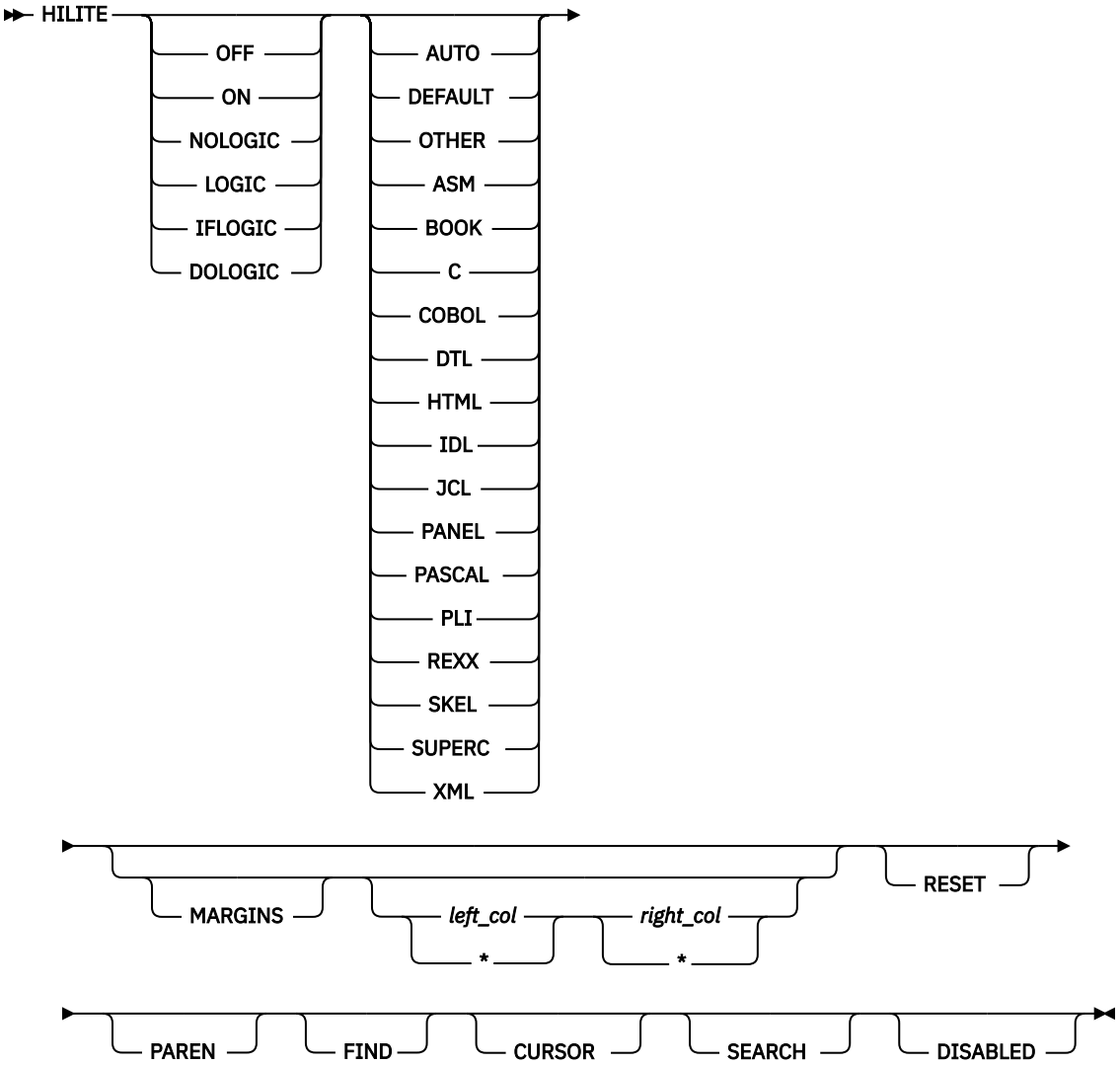
HILITE is used to control the use of color in the editor by changing the settings for the enhanced color and language-sensitive editing features.

Note: Language-sensitive and enhanced coloring of the edit session is only available when enabled by the installer or the person who maintains the ISPF product. For information on enabling the enhanced color function, see *z/OS ISPF Planning and Customizing*.

HILITE with *no* operands presents a dialog (see "The HILITE dialog" on page 33) that allows you to change coloring options, and to see which keywords are supported for each language.

Language and logic hiliting is not supported for ASCII or UTF-8 editing sessions and the HILITE command is not available during these edit sessions.

Syntax



ON
Sets program coloring ON and turns LOGIC coloring off.

OFF
Sets coloring OFF, with the exception of cursor, find, and parenthesis highlighting.

LOGIC
LOGIC highlighting matches logical language-specific keywords in the same color. If an unmatched *closing* keyword is found, such as END for PL/I or :eul. for BookMaster, it is highlighted in reverse video pink *only* if HILITE LOGIC is active. When logic is being highlighted, only comments are highlighted along with it.

Logic highlighting is available only for PL/I, PL/X, REXX, OTHER, C, SKELS, Pascal, and BookMaster. HILITE LOGIC turns on both IFLOGIC and DOLOGIC.

Note: LOGIC highlighting can be turned off by issuing HILITE ON, HILITE NOLOGIC, or HILITE RESET commands. Changing the HILITE language does not change the LOGIC setting.

IFLOGIC
Turns on IF/ELSE logic matching. IFLOGIC matches IF and ELSE statements. When IFLOGIC is enabled, unmatched ELSE keywords are highlighted in reverse video pink.

DOLOGIC
Turns on DO/END logic matching. DOLOGIC matches logical blocks such as DO/END in PL/I or :ol/:eol in BookMaster. For the C language, DOLOGIC matches curly braces ({ and }). C trigraphs for curly braces are not recognized and are not supported by DOLOGIC highlighting. When DOLOGIC is enabled, unmatched logical block terminators (such as END keywords in PL/I, :e tags in BookMaster or right braces (}) in C) are highlighted in reverse video pink.

NOLOGIC
Same as ON.

AUTO
Allows ISPF to determine the language. See "Automatic language selection" on page 29 for more information.

DEFAULT
Highlights the data in a single color.

OTHER
Highlight the data as a pseudo-PL/I language. Limited CLIST support is also provided by OTHER.

ASM
Highlights the data as Assembler.

BOOK
Highlights the data as BookMaster.

C
Highlights the data as C.

COBOL
Highlights the data as COBOL

DTL
Highlights the data as Dialog Tag Language.

HTML
Highlights the data as HTML.

IDL
Highlights the data as IDL.

JCL
Highlights the data as MVS Job Control Language.

PANEL
Highlights the data as ISPF Panel Language.

PASCAL
Highlights the data as Pascal.

PLI
Highlights the data as PL/I.

REXX
Highlights the data as REXX.

SKEL
Highlights the data as ISPF Skeleton Language.

SUPERC
Highlights the data as a SuperC Listing.

XML
Highlights the data as XML.

MARGINS [*left-margin* | * [*right-margin* | *]]
Specifies either or both of the left-margin or right-margin parameters for languages C, PL/I, and PL/X. The MARGINS keyword can be included on the same command that includes one of these languages.

It cannot be specified when the language AUTO is specified, even if the language would subsequently be determined to be C, PL/I, or PL/X.

left-margin
The left hand margin for processing the language source. The value must be within the range as defined by the language. The maximum value is 254 for C, 100 for PL/I, and 65 for PL/X. If *left-margin* exceeds the last input column or if an asterisk (*) is specified, the default left margin is obtained from the ISPF configuration table keyword for this language (HILITE_MARGIN_C, HILITE_MARGIN_PLI, or HILITE_MARGIN_PLX).

right-margin
The right hand margin for processing the language source. The value must be within the range as defined by the language. The maximum value is 255 for C, 200 for PL/I, and 80 for PL/X. If *right-margin* exceeds the last input column or if an asterisk (*) is specified, the default right margin is obtained from the ISPF configuration table keyword for this language (HILITE_MARGIN_C, HILITE_MARGIN_PLI, or HILITE_MARGIN_PLX).

RESET
Resets defaults (LANG AUTO, COLOR ON, LOGIC OFF, FIND ON and CURSOR ON).

PAREN
Toggles parenthesis matching. When parenthesis matching is active, only comments are specially colored. All other code appears in the default color. Note that extra parenthesis highlighting is always active when highlighting is active.

FIND
The HILITE FIND command toggles the highlighting color of any string that would be found by an RFIND. The user can select the highlight color. The default is reverse video white.

Only non-picture strings are supported, and the only additional qualifiers recognized are hex strings (X'...'), character strings (C'...'), text strings (T'...'), WORD, PREFIX and SUFFIX, and boundaries specified in the FIND command. Hex strings may be highlighted, but non-displayable characters are not highlighted. Labels are ignored when FIND strings are highlighted.

Because FIND highlighting is not quite as robust as the FIND command itself, the editor may highlight more occurrences of the FIND string than FIND would actually locate. The FIND operand toggles the display of search strings. If HILITE FIND is issued when FIND highlighting is in effect, FIND highlighting is disabled. Similarly, if FIND highlighting is disabled, the HILITE FIND command enables it.

Note: RESET has been enhanced, through the addition of a FIND operand, to temporarily disable the highlighting of FIND strings until the next FIND, RFIND, CHANGE, or RCHANGE command is issued. RESET with the FIND operand (or no operands at all), temporarily disables the highlighting of FIND strings.

CURSOR
The CURSOR operand toggles the highlighting of the phrase that contains the cursor in a user selectable color. The default is white.

Cursor highlighting in Edit is performed in a manner similar to the way it is done in Browse. The entire phrase from the previous blank to the next blank is highlighted. The CURSOR operand toggles cursor highlighting. If HILITE CURSOR is issued when CURSOR highlighting is in effect, CURSOR highlighting is disabled. Similarly, if CURSOR highlighting is disabled, the HILITE CURSOR command enables it.

SEARCH
HILITE SEARCH finds the first unmatched END, ELSE, }, or) above the last displayed line on the screen. If a mismatched item is found, the file is scrolled so that the mismatch is at the top of the screen. The search for mismatches only occurs for lines above the last displayed line, so you may need to scroll to the bottom of the file before issuing the HI SEARCH command.

Search is not available when the DEFAULT language operand is used. Search for language keywords is only supported for languages which supported by the logic option.

DISABLED
Turns off all HILITE features and removes all action bars. This benefits performance at the expense of function. Since DISABLED status is not stored in the edit profile, you need to reenter this operand each time you enter the editor. When DISABLED is in effect, keylists are unavailable for that edit session.

Description

The HILITE primary command can be used to highlight, in user-specified colors, many language-specific constructs, program logic features, the phrase containing the cursor, and any strings that match the previous FIND operation or those that would be found by an RFIND or RCHANGE request. In addition, when HILITE is entered with no operands, a dialog appears that allows you to set default colors for the data area in non-program files, for any characters typed since the previous Enter or PF key entry, and for strings located by FIND.

Both HI and HILIGHT are valid synonyms for HILITE.

When the code page being used is not the English codepage, the HILITE primary command does not detect key sequences if the control character within the key sequence has a different binary representation in the code page being used from the binary representation in the codepage used for English. For example, in the C language, a '\' is used as an escape sequence character to influence the interpretation of the next character; however, the '\' has a different binary representation in different codepages.

Note:

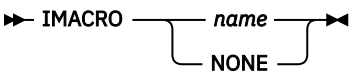
1. Highlighting is *not* available for edit sessions that involve:
   - Data sets with record lengths greater than 255
   - Mixed mode edit sessions (normally used when editing DBCS data)
   - Formatted data
2. Five character labels starting with the letter "o", in the form ".Oxxxx", are used by the COMPARE command. Any labels of this form can cause unpredictable highlighting results.

IMACRO—Specify an Initial Macro

The IMACRO primary command saves the name of an initial macro in the current edit profile.

See “Initial macros” on page 24 for more information on creating and using initial macros.

Syntax



name
The name of the initial macro to be run when you are editing the data set type that matches the current edit profile. This macro is run before any data appears.

For more information about displaying and defining a profile, see “Displaying or defining an edit profile” on page 17.

NONE
Indicates that no macro is to be run at the beginning of each edit session. The edit profile shows a value of NONE when no initial macro has been specified.

Examples

To save STARTUP as the initial macro, type:

```
IMACRO STARTUP
```

To reset the profile with no initial macro, type:

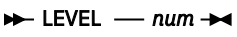
```
IMACRO NONE
```

LEVEL—Specify the Modification Level Number

The LEVEL primary command allows you to control the modification level that is assigned to a member of an ISPF library.

See "Version and modification level numbers" on page 26 for more information about level numbers.

Syntax



num
The modification level. It can be any number from 0 to 99.

Description

To specify the modification level number:

1. On the command line, type:

   ```
   LEVEL num
   ```

   where *num* is the new level number.
2. Press Enter.

Examples

In Figure 132 on page 248, the version and modification level numbers on line 1 show that this is Version 1, Modification 3 (`01.03`). Type LEVEL 0 on the command line to reset the modification level number to `00`.

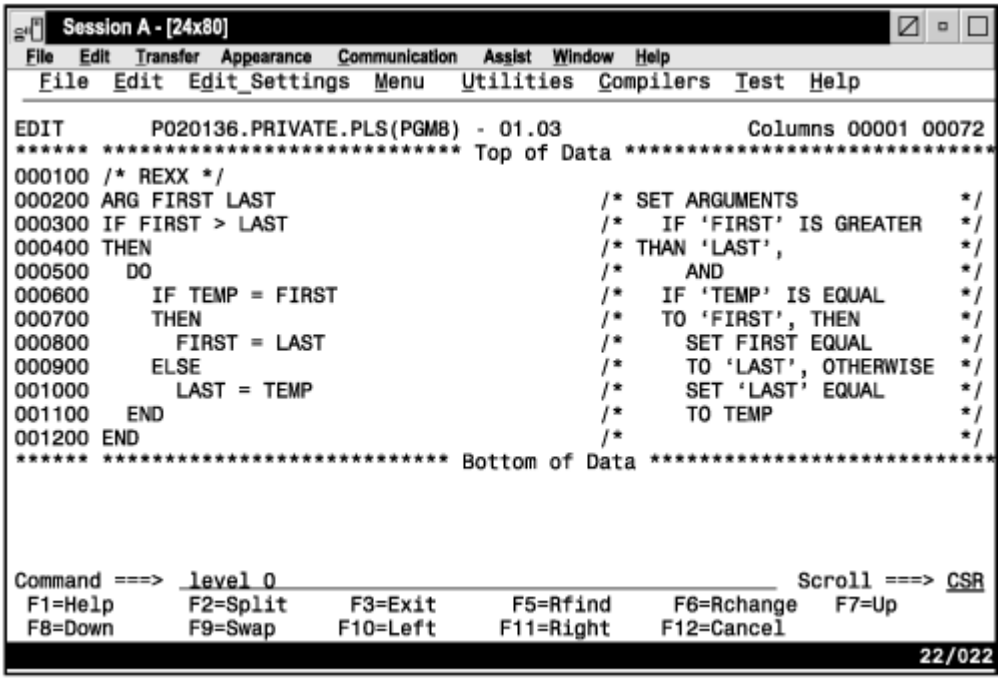


Figure 132. Member with modification level of 03

After you press Enter, the editor resets the modification level, as shown in Figure 133 on page 248.

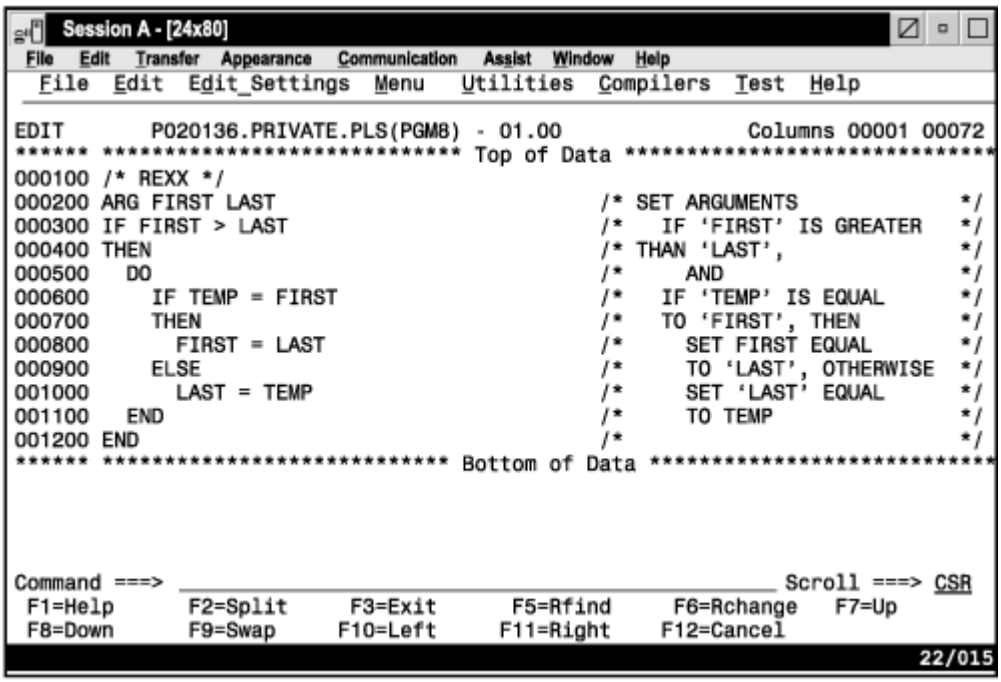


Figure 133. Member with modification level reset to 00

LF—realign data on the ASCII linefeed character

The LF primary command allows you to realign the data being edited by interpreting the ASCII linefeed character X'0A'. The LF primary command is not available when editing a z/OS UNIX file. Instead, use the

ASCII edit facility to automatically realign the data in a z/OS UNIX file based on the ASCII linefeed and carriage return characters. See "Working with ASCII data" on page 50.

Note: If the data is saved, it is saved in the realigned state. There is no command to reverse the alignment. The command should not be executed twice against the data, as the blanks following the linefeed character will be interpreted as part of the data for the next line.

Syntax

See "Restructuring data based on the linefeed character" on page 51 for more information.

Examples

To realign the data being edited by interpreting the ASCII linefeed character X'0A':

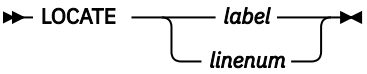
```
LF
```

LOCATE—Locate a Line

The LOCATE primary command allows you to scroll up or down to a specified line. The line then appears as the first line on the panel. There are two forms of LOCATE: specific and generic.

Syntax

Specific LOCATE syntax



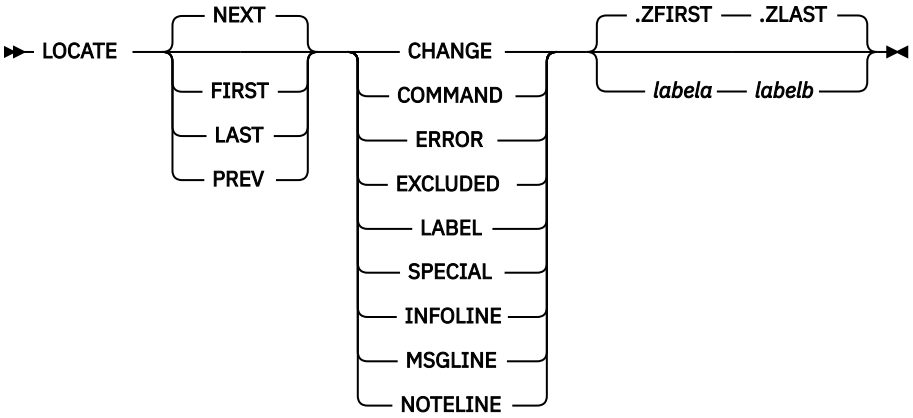
The specific form of the LOCATE command positions a particular line at the top of the panel. You must specify either a line number or a label.

label
A previously assigned label.

linenum
An edit line number. If that line number exists, it appears at the top. If the line number does not exist, the line with the next lower number appears at the top of the data area.

The *linenum* operand is a numeric value of up to 8 digits. You do not need to type leading zeros. If the operand contains 6 or fewer digits, it refers to the number in the line command field to the left of each line. If *linenum* contains 7 or 8 digits, it refers to the sequence numbers in the data records. For NUMBER ON STD, the editor refers to the *modification flag*. For NUMBER OFF, it refers to the *ordinal line number* (first=1, fifth=5, and so on). For NUMBER ON COBOL, it refers to the number in the line command field, which is the data sequence number. See "Sequence number format and modification level" on page 26 for more information.

Generic LOCATE syntax



The generic LOCATE command positions the panel to the first, last, next, or previous occurrence of a particular kind of line.

FIRST
Searches from the first line, proceeding forward.

LAST
Searches from the last line, proceeding backward.

NEXT
Searches from the first line of the page displayed, proceeding forward.

PREV
Searches from the first line of the page displayed, proceeding backward.

CHANGE
Searches for a line with a change flag (==CHG>).

COMMAND
Searches for a line with a pending line command.

ERROR
Searches for a line with an error flag (==ERR>).

EXCLUDED
Searches for an excluded line.

LABEL
Searches for a line with a label.

SPECIAL
Searches for a special non-data (temporary) line:

- Bounds line flagged as =BNDS>
- Column identification lines flagged as =COLS>
- Information lines flagged as ======
- Mask lines flagged as =MASK>
- Message lines flagged as ==MSG>
- Note lines flagged as =NOTE=
- Profile lines flagged as =PROF>
- Tabs line flagged as =TABS>

INFOLINE
Searches for information lines flagged with ======

MSGLINE
Searches for message lines flagged with ==MSG>

NOTELINE
Searches for note lines flagged with =NOTE=

labela*, *labelb
Labels identifying the start and end of the group of lines to be searched.

For more information about using labels to identify a group of lines, see "Labels and line ranges" on page 59.

Examples

To find the next special line, type:

```
LOCATE SPE
```

To find the first error line (==ERR>), type:

```
LOCATE ERR FIRST
```

To find the next line with a label, type:

```
LOC NEXT LABEL
```

To find the next excluded line between .START and .END, type:

```
LOC X .START .END
```

To find the first excluded line between .E and .S, type:

```
L FIRST .E .S X
```

To find the first message line, type:

```
LOCATE FIRST MSGLINE
```

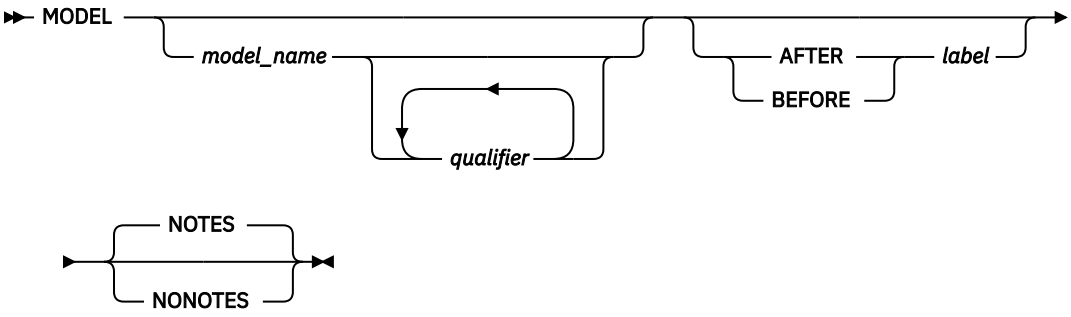
MODEL—Copy a Model into the Current Data Set

The *model name* form of the MODEL primary command copies a specified dialog development model before or after a specified line.

The *class name* form of the MODEL primary command changes the model class that the editor uses to determine which model you want. For more information on edit models, see Chapter 4, "Using edit models," on page 69.

Syntax

Model name syntax



If you omit the model name or a required qualifier, or if there is a validation error, the editor displays a series of selection panels from which you can select the desired information.

model_name
The name of the model to be copied, such as VGET for the VGET service model. This operand can also be one of the options listed on a model selection panel, such as V1 for the VGET service model. See *z/OS ISPF Planning and Customizing* for a list of models and model names.

qualifier
The name of a model on a secondary model selection panel, such as TBCREATE for the TBCREATE service model. This operand can also be one of the options listed on a model selection panel, such as G1 for the TBCREATE service model.

For example, a model selection panel allows you to enter T1 to choose table models. Another model selection panel then appears for choosing table models, such as G1 for the TBCREATE service model. Therefore, your MODEL primary command could use either TABLES or T1 as the model-name operand and either TBCREATE or G1 at the qualifier operand. The simplest way would be to use TBCREATE or G1 as the model-name operand and omit the qualifier operand. See *z/OS ISPF Planning and Customizing* for a list of models and model names.

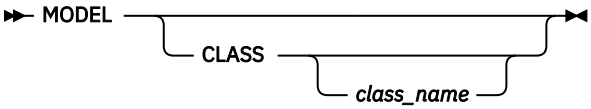
AFTER *label*
Identifies the line after which the model is to be copied. If you have not defined a label, use the A or B line command to specify the destination. The only time this operand or the BEFORE label operand is not required is when the data set or member is empty.

BEFORE *label*
Identifies the line before which the model is to be copied. If you have not defined a label, use the A or B line command to specify the destination. The only time this operand or the AFTER label operand is not required is when the data set or member is empty.

NOTES
Overrides the current edit profile setting for note mode, to include any notes that are part of the model.

NONOTES
Overrides the current edit profile setting for note mode, to exclude any notes that are part of the model.

Class name syntax



If you omit *class_name*, or if there is a validation error, the editor displays a series of selection panels from which you can select the desired information.

CLASS
When entered without the optional class_name operand, the editor displays the Model Classes panel, from which you can select a model class. When entered with the class_name operand, the macro specifies that the current model class is to be replaced by class_name. In both cases, the new class name is used for all models from that point on, until you change the model class again or end the edit session.

class_name
Specifies a new class for the current edit session. It must be a name on the Model Classes panel or an allowable abbreviation. The model class coincides with the type of model, such as REXX, COBOL, or FORTRAN.

Examples

You are editing a new member named NEWMEM and have not decided which service to use first. Figure 134 on page 253 shows the display screen for NEWMEM. Type MODEL on the command line without any operands. Here, the model name form of the MODEL command is used and the A (after) line command is used instead of the AFTER operand.

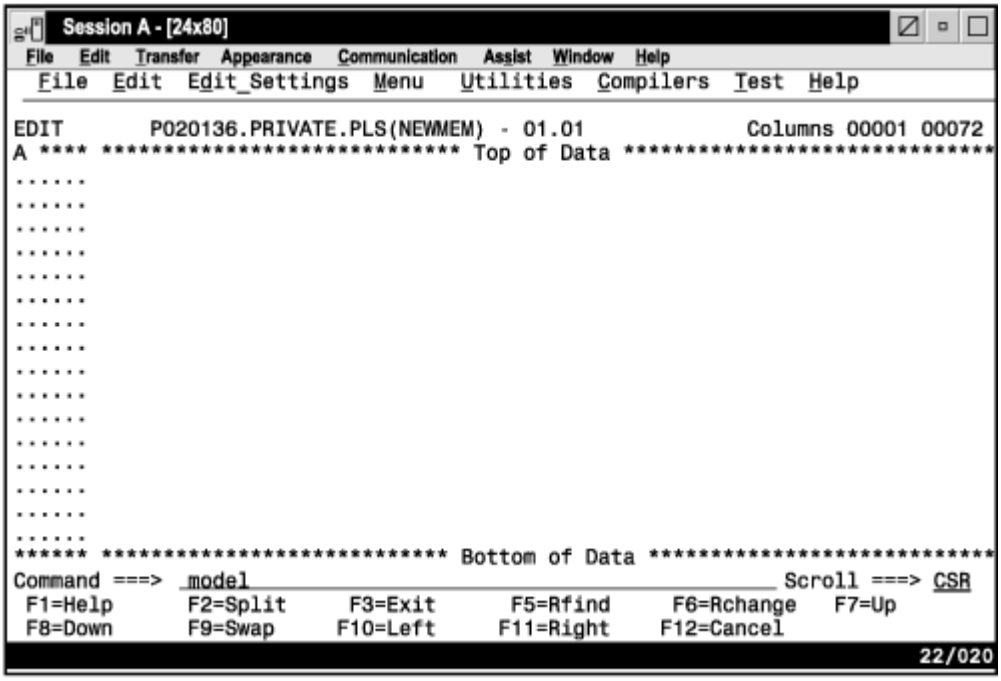


Figure 134. Before Model command

The data set type is EXEC, so the editor displays the REXX Models panel (Figure 135 on page 253) when you press Enter. To begin with the VGET service, you type V1 on the Option line and press Enter.

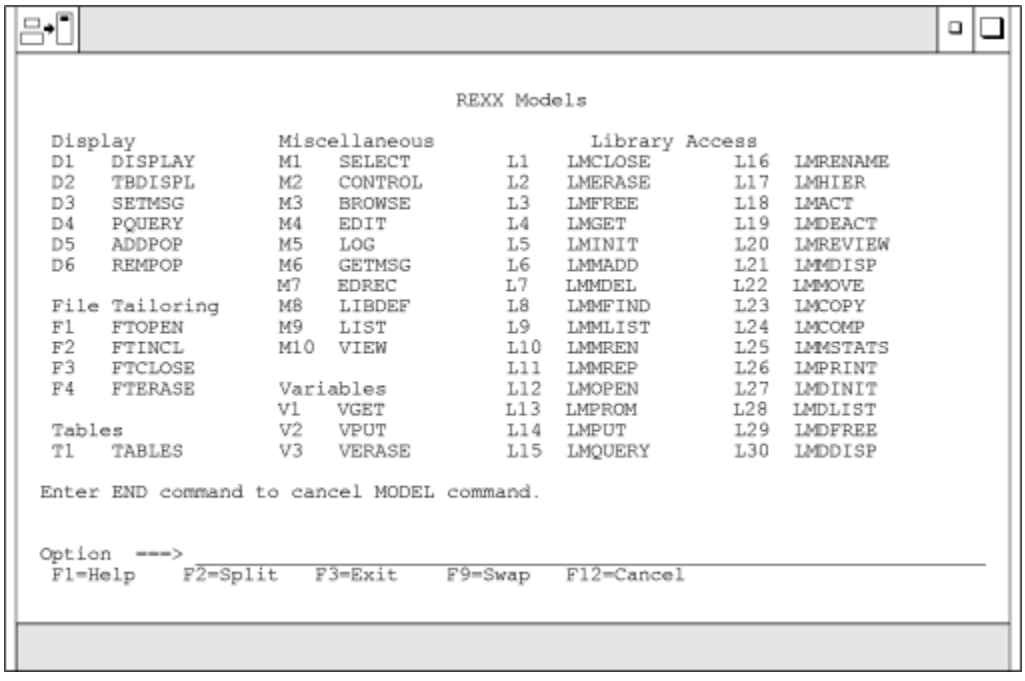


Figure 135. REXX Models panel (ISREMRXC)

The editor inserts the VGET service model into the NEWMEM member, as shown in Figure 136 on page 254. Because the edit profile is set to NOTE ON, the model's notes are also included.

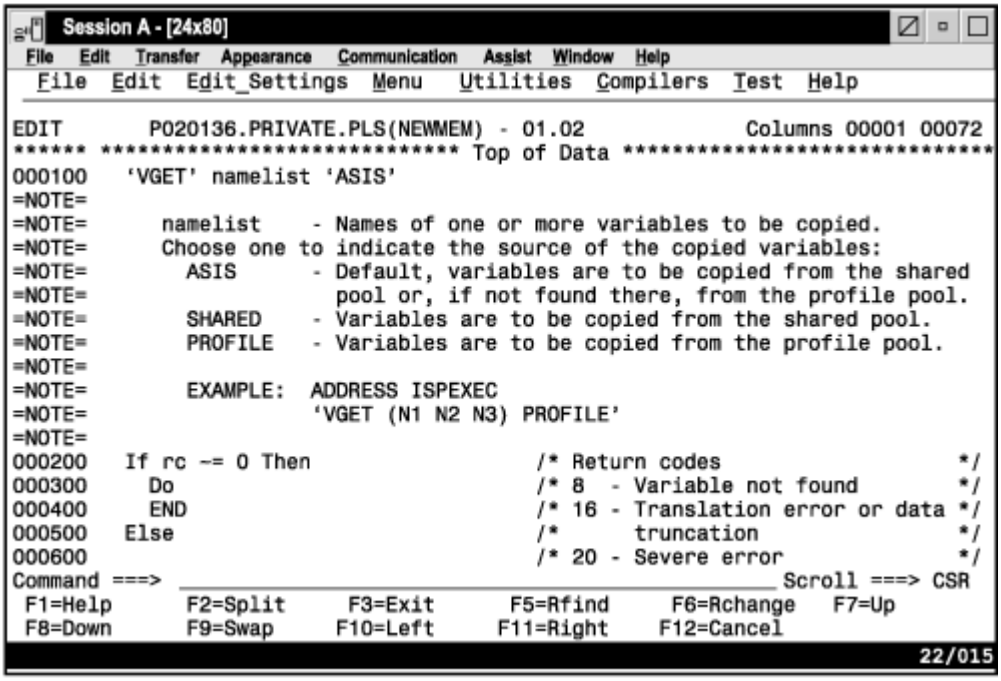


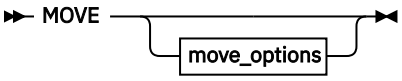
Figure 136. REXX model of VGET service

MOVE—Move Data

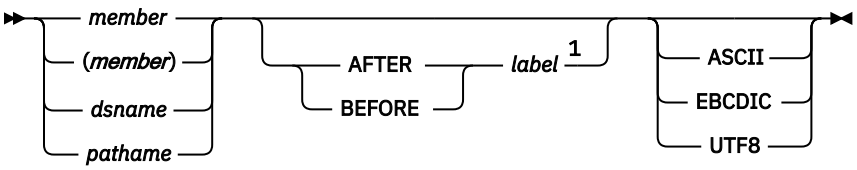
The MOVE primary command moves a sequential data set, member of a partitioned data set, or z/OS UNIX file into the data being edited.

If no options are specified with the MOVE command, the Edit/View Move panel is displayed.

Syntax



move_options



Notes:

[1] If you don't specify the position using a label, you must specify the position by using an A or B line command.

member
A member of the ISPF library or partitioned data set you are editing.

dsname
A partially qualified or fully qualified data set name. If the data set is partitioned you can include a member name in parentheses or select a member from a member list.

pathname
The pathname for a z/OS UNIX regular file or directory. If a directory is specified, a directory selection list is displayed, allowing you to select the file to be moved. (Also, see "Specifying z/OS UNIX pathnames with edit primary and macro commands" on page 16.)

AFTER
The data is moved after the line with the specified label.

BEFORE
The data is moved before the line with the specified label.

label
Label identifying the line where the data is to be copied. It can be either a label that you define or one of the editor-defined labels, such as .ZF or .ZL.

ASCII, EBCDIC, UTF8
When one of these keywords is supplied, if the data is using a different character set to that designated by the keyword, the data being moved in from the external file is converted from the character set designated by the keyword to the character set specified for the file being edited or to the terminal character set.

The label can be either a label you define or one of the editor-defined labels, such as `.ZF` and `.ZL`. If you have not defined a label and the editor-defined labels are not appropriate for your purpose, use the A (after) or B (before) line command to specify the data's destination.

If the data set, member, or z/OS UNIX file that you are editing is empty, you do not need to specify a destination for the data being moved.

Note: If the member name or data set name is less than 8 characters and the data set you are editing is partitioned, a like-named member is copied. If a like-named member does not exist, the name is considered to be a partially qualified data set name.

Description

MOVE adds data that already exists to the data set, member, or z/OS UNIX file that you are editing. Use MOVE if you want to move data rather than copy it from one data set, member, or z/OS UNIX file to another.

The member, sequential data set, or z/OS UNIX file is deleted after the move. For a concatenated sequence of ISPF libraries, the deletion occurs only if the member was in the first library.

To move data into an empty data set, member, or z/OS UNIX file:

1. On the command line, type:

   ```
   MOVE member
   ```

   or:

   ```
   MOVE dsname
   ```

   or:

   ```
   MOVE pathname
   ```

   The member, dsname, and path name operands are optional. If you do not specify the name of a member, data set, or z/OS UNIX file to be moved, the Edit Move panel appears. Enter the data set, member name, or z/OS UNIX file on this panel.
2. Press Enter. The data is moved.

To move data into a data set, member, or z/OS UNIX file that is not empty:

1. On the command line, type:

   ```
   MOVE member AFTER | BEFORE label
   ```

   or:

   ```
   MOVE dsname AFTER | BEFORE label
   ```

   or:

   ```
   MOVE pathname AFTER | BEFORE label
   ```

   The member, dsname, and path name operands are optional.

   The AFTER label and BEFORE label operands are optional, also. However, if the data set, member, or z/OS UNIX file that is to receive the moved data is not empty, you must specify a destination for the moved data. Therefore, if you do not use a label, substitute either the A (after) or B (before) line command as the destination of the moved data. However, a number indicating that the A or B command should be repeated cannot follow the line command.

   If the data set, member, or z/OS UNIX file is not empty and you do not specify a destination, a "MOVE/COPY Pending" message is displayed in the upper-right corner of the panel and the data is not moved. When you type a destination and press Enter, the data is moved.
2. Press Enter. If you entered the name of a member, data set , or z/OS UNIX file, the member, data set, or z/OS UNIX file is moved. Otherwise, the Edit Move panel appears. See the previous example for more information.

See “Copying and moving data” on page 41 if you need more information.

Examples

These steps show how you can move data when you omit the member name and the editor panels appear:

1. Type MOVE on the command line and specify the destination of the operation. In Figure 137 on page 256, the data is to be moved after line 000400, as specified by the A (after) line command.

```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
──────────────────────────────────────────────────────────────────────────────
EDIT       USERSID.TEST(DESTDATA) - 01.01               Columns 00001 00072
Command ===> move                                          Scroll ===> PAGE
****** ***************************** Top of Data ******************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
000100 $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
000200 This is the member into which the lines are to be moved.
000300      +---------------------+
a00400      |                     |
000500      |                     |
000600      |                     |
000700      |                     |
000800      +---------------------+
000810
000900 $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
001000
****** **************************** Bottom of Data ****************************


 F1=Help      F2=Split     F3=Exit      F5=Rfind     F6=Rchange   F7=Up
 F8=Down      F9=Swap     F10=Left     F11=Right    F12=Cancel
```

   Figure 137. Member before data is moved
2. When you press Enter, the Edit/View Move panel appears. Specify the data you want moved.

   This example (Figure 138 on page 257) moves the data set member named TODATA.

```
  Menu  RefList  Utilities  Help
_______________________________________________________________________
                                Edit/View Move
Command ===>

"Current" Data Set: USERSID.TEST(DESTDATA)

From ISPF Library:
   Project . . . PDFTDEV
   Group . . . . USERSID  . . . ________ . . . ________ . . . ________
   Type  . . . . MSGGEN
   Member  . . . ________   (Blank or pattern for member selection list)

From Other Partitioned or Sequential Data Set, or z/OS UNIX file:
   Name . . . . . TEST(TODATA)                                          +
   Volume Serial  ______    (If not cataloged)

Data Set Password  . .           (If password protected)

Data Conversion option
_  1. EBCDIC
   2. ASCII
   3. UTF-8

Press ENTER key to move. (Member or sequential data set may be deleted)
Enter END command to cancel move.

 F1=Help      F2=Split     F3=Exit      F4=Expand    F7=Backward  F8=Forward
 F9=Swap     F10=Actions  F12=Cancel
```

Figure 138. Edit Move panel (ISREMOV1)

3. Figure 139 on page 257 shows the contents of the TODATA member which is moved into the original data set. This panel is shown only for this example, so you can see the data that is being moved. It is not displayed during a move sequence.

```
EDIT       USERSID.TEST(TODATA) - 01.00                  Columns 00001 00072
Command ===> ____________________________________________ Scroll ===> PAGE
****** ***************************** Top of Data ******************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
000100 +----------------+
000200 | This is the    |
000300 | material to    |
000400 | be created in  |
000500 | another member |
000600 +----------------+
****** **************************** Bottom of Data ****************************
```

Figure 139. Data set to be moved

4. When you press Enter, the editor moves the data and displays a short message in the upper right corner of the panel. Figure 140 on page 258 shows the result of using MOVE.

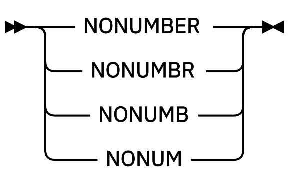
```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
 ───────────────────────────────────────────────────────────────────
EDIT       USERSID.TEST(DESTDATA) - 01.01                  Member TODATA moved
Command ===> ________________________________________________ Scroll ===> PAGE
****** ***************************** Top of Data ******************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
000100 $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
000200 This is the member into which the lines are to be moved.
000300      +---------------------+
000400      |                     |
000410 +-----------------+
000420 | This is the     |
000430 | material to     |
000440 | be created in   |
000450 | another member |
000460 +-----------------+
000500      |                     |
000600      |                     |
000700      |                     |
000800      +---------------------+
000810
 F1=Help      F2=Split     F3=Exit      F5=Rfind     F6=Rchange   F7=Up
 F8=Down      F9=Swap     F10=Left     F11=Right    F12=Cancel
```

Figure 140. Member after data has been moved

NONUMBER—Turn Off Number Mode

The NONUMBER primary command turns off number mode, which controls the numbering of lines in the current data.

Syntax



Description

You can also use NUMBER OFF to turn off number mode.

When number mode is off, NONUMBER prevents any verification of valid line numbers, generation of sequence numbers, and the renumbering of lines that normally occurs when autonum mode is on.

Examples

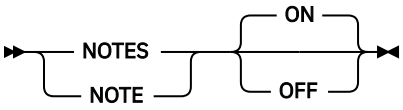
To turn number mode off by using NONUMBER, enter this command:

```
NONUMBER
```

NOTES—Display Model Notes

The NOTES primary command sets note mode, which controls whether notes are displayed when a dialog development model is inserted into the data.

Syntax



ON
Displays explanatory notes when a model is copied into the data being edited or when notes are added to the edit session by an edit macro.

OFF
Does not display explanatory notes.

Description

Note mode is saved in the edit profile. To check the setting of note mode:

1. On the command line, type:

```
PROFILE 4
```

2. Press Enter. The note mode setting appears as either `NOTE  ON` or `NOTE  OFF` on the fourth line of the edit profile.

You can set the note mode with a primary command and then use the NOTES or NONOTES operand on the MODEL command to override the default mode for a particular model.

See "MODEL—Copy a Model into the Current Data Set" on page 251 for information about copying dialog development models.

Examples

To set note mode on:

1. On the command line, type:

```
NOTES ON
```

2. Press Enter. The next time you insert a model, the explanatory notes appear along with the model.

To set note mode off:

1. On the command line, type:

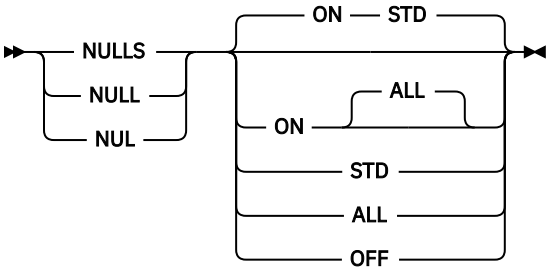
```
NOTES OFF
```

2. Press Enter. The next time you insert a model, the explanatory notes are not displayed along with the model.

NULLS—Control Null Spaces

The NULLS primary command sets nulls mode, which determines whether trailing spaces in each data field are written to the panel as blanks or nulls.

Syntax



ON STD
Specifies that in fields containing any blank trailing space, the space is written as one blank followed by nulls. If the field is entirely empty, it is written as all blanks.

ON ALL
Specifies that all trailing blanks and all-blank fields are written as nulls.

OFF
Specifies that trailing blanks in each data field are written as blanks.

Description

Blank characters (X'40') and null characters (X'00') both appear as blanks. When you use the I (insert) line command, the data entry area appears as blanks for NULLS ON STD and as nulls for NULLS ON ALL.

Trailing nulls simplify use of the Ins (insert) key on the IBM 3270 keyboard. You can use this key to insert characters on a line if the line contains trailing nulls.

Besides using the NULLS command, you can create nulls at the end of a line by using the Erase EOF or Del (delete) key. Null characters are never stored in the data; they are always converted to blanks.

Note: When you swap screens in split screen mode, the nulls are replaced by spaces until you press an interrupt key, such as Enter, or a function key.

Examples

To set nulls mode on with all trailing blanks and all-blank fields written as nulls, enter this command:

```
NULLS ON ALL
```

To set nulls mode on with blank trailing space written as one blank followed by nulls and empty fields written as all blanks, enter this command:

```
NULLS ON STD
```

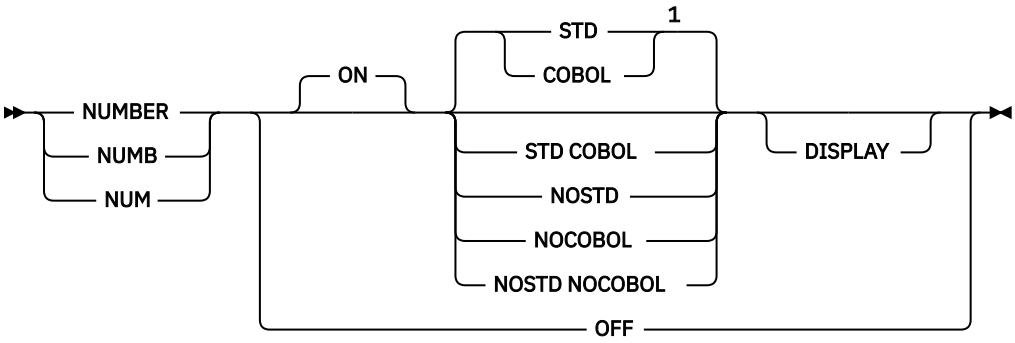
To set nulls mode off and thus have trailing blanks in each data field, enter this command:

```
NULLS OFF
```

NUMBER—Generate Sequence Numbers

The NUMBER primary command sets number mode, which controls the numbering of lines in the current data.

Syntax



Notes:

[1] STD is the default for non-COBOL data set types. COBOL is the default for COBOL data set types.

ON
Automatically verifies that all lines have valid numbers in ascending sequence and renumbers any lines that are either unnumbered or out of sequence. You can also use RENUM to turn number mode on and renumber lines.

The editor interprets the STD, COBOL, and DISPLAY operands only when number mode is turned on.

OFF
Turns number mode off. You can also use NONUMBER to turn number mode off. If you alter or delete sequence numbers and enter NONUMBER on the Command line at the same time, the editor issues the message `Some input data ignored` and discards the data typed over the sequence numbers. The editor converts the original sequence numbers to data.

STD
Numbers the data in the standard sequence field.

COBOL
Numbers the data in the COBOL field.

STD COBOL
Numbers the data in both fields.

If both STD and COBOL numbers are generated, the STD number is determined and then used as the COBOL number. This can result in COBOL numbers that are out of sequence if the COBOL and STD fields were not synchronized. Use RENUM to force synchronization.

NOSTD
Turns standard number mode off.

NOCOBOL
Turns COBOL number mode off.

NOSTD NOCOBOL
Turns both the standard number mode and COBOL number mode off.

DISPLAY
Causes the width of the data window to include the sequence number fields. Otherwise, the width of the window does not include the sequence number fields. When you display a data set with a logical record length of 80 and STD numbering, the sequence numbers are not shown unless you are using a 3278 Model 5 terminal, which displays 132 characters. Automatic left or right scrolling is performed, if required, so that the left most column of the data window is the first column displayed.

Description

Attention: If number mode is off, make sure the first 6 columns of your data set are blank before turning COBOL number mode on. Otherwise, the data in these columns is replaced by sequence numbers. If that happens and if edit recovery or SETUNDO is on, you can use the UNDO command

to recover the data. You can also use CANCEL at any time to end the edit session without saving the data.

When number mode is on, NUMBER verifies that all lines have valid numbers in ascending sequence. It renumbers any lines that are either unnumbered or out of sequence, but it does not otherwise change existing numbers.

In number mode, the editor automatically generates sequence numbers in the data for new lines created when data is copied or inserted. The editor also automatically renumbers the data when it is saved if autonum mode is in effect.

If the number overlays the shift-in (SI) or shift-out (SO) characters, the double-byte characters appear incorrectly and results are unpredictable.

Examples

To number data in the standard sequence field, enter this command:

```
NUMBER ON STD
```

To number data in both the standard and COBOL fields and include sequence numbers in the display, enter this command:

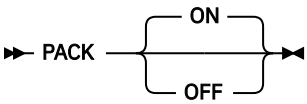
```
NUMBER ON STD COBOL DISPLAY
```

PACK—Compress Data

The PACK primary command sets pack mode, which controls whether the data is to be stored in packed format.

The PACK command saves the pack mode setting in the edit profile. See “Packing data” on page 16 for more information about packing data.

Syntax



ON
Saves data in packed format.

Note: You cannot specify PACK ON when editing a z/OS UNIX file.

OFF
Saves data in unpacked (standard) format.

Examples

To set pack mode on, enter this command:

```
PACK ON
```

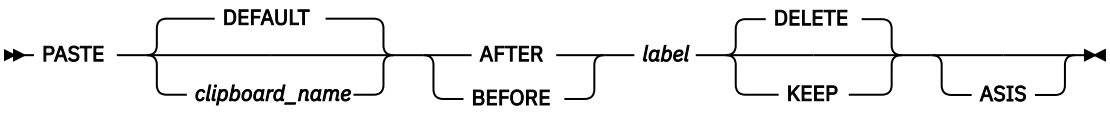
To set pack mode off, enter this command:

```
PACK OFF
```

PASTE—Move or Copy Lines from Clipboard

The PASTE primary command moves or copies lines from a clipboard into an edit session.

Syntax



clipboard_name
The name of the clipboard to use. If you omit this parameter, the ISPF default clipboard (named DEFAULT) is used. You can define up to ten additional clipboards. The size of the clipboards and number of clipboards might be limited by installation defaults.

AFTER *label*
The data is copied after the line with the specified label.

BEFORE *label*
The data is copied before the line with the specified label.

KEEP
The copied lines are not removed from the clipboard.

DELETE
The copied lines **are** removed from the clipboard.

ASIS
The PASTE command determines the character set of the data in the clipboard. If this is different to the character set being used for the file being edited an automatic conversion occurs for the data being pasted into the file.

If ASIS is specified, then the automatic conversion does not take place.

Note:

1. You should always specify KEEP or DELETE in an edit macro because the default behavior may have been changed by the user.
2. You can specify the default behavior (KEEP or DELETE) using the EDITSET primary command.

Description

PASTE copies or moves lines from a specified clipboard to the current edit session. If lines in the clipboard are longer than the lines in the edit session, they are truncated.

Only the data portion of the line is saved in the clipboard. Line numbers are *not* saved. If the data was CUT from a data set that had sequence numbers and is PASTEd into an edit session without sequence numbers, or if it was CUT from a data set without sequence numbers and PASTEd into a session with sequence numbers, some shifting of data is likely to occur.

Examples

To paste data from the default clipboard to the line after the last line in the edit session:

```
PASTE AFTER .ZLAST
```

To paste data from the default clipboard to the line after the first line in the edit session, without clearing the contents of the clipboard:

```
PASTE AFTER .ZFIRST KEEP
```

PRESERVE—Enable Saving of Trailing Blanks

The PRESERVE primary command enables or disables the saving of trailing blanks in the editor. This gives you the ability to override the setting for the "Preserve VB record length" field on the edit entry panel.

Syntax

ON
The editor preserves the record length of the record when the data is saved.

OFF
Turns truncation on. ISPF removes trailing blanks when saving variable-length files.

Regardless of the PRESERVE setting, if a line has a length of zero, ISPF saves 1 blank.

Description

PRESERVE ON causes the editor to save trailing blanks for variable length files. The number of blanks saved for a particular record is determined by one of these:

- The original length of the record when it was read in to the editor
- The number of blanks required to pad the record length specified by the SAVE_LENGTH edit macro command
- The length of the record that was saved on disk during a previous SAVE request in the same edit session

PRESERVE OFF causes the editor to truncate trailing blanks. If a line is empty ISPF saves 1 blank.

Use of the PRESERVE command does not prevent the editor from working on data past the specified record length. The length set and returned by the PRESERVE command is only used when the data is written and does not affect the operation of other edit functions.

Examples

To enable the editor to remove trailing blanks when data is saved, enter this command:

```
PRESERVE OFF
```

To save the trailing blanks, enter this command:

```
PRESERVE ON
```

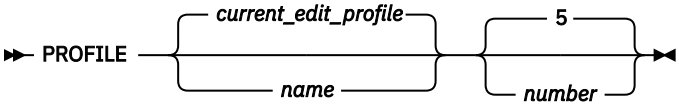
PROFILE—Control and Display Your Profile

There are three forms of the PROFILE primary command:

- The *control* form displays your current edit profile, defines a new edit profile, or switches to a different edit profile.
- The *lock* form locks or unlocks the current edit profile.
- The *reset* form specifies that the site-wide configuration for new edit profiles is to be used.

Syntax

Profile control



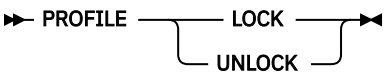
name
The profile name. It can consist of up to 8 alphanumeric characters, the first of which must be alphabetic. The edit profile table is searched for an existing entry with the same name. That profile is then read and used. If one is not found, a new entry is created in the profile table.

If you omit this operand, the current edit profile is used.

number
The number of lines, from 0 through 9, of profile data to be displayed. When you type `0` as the number, no profile data is displayed. When no operands are entered, the first five lines, which contain the =PROF> flags, are always displayed. However, the =MASK> and =TABS> lines are not displayed if they contain all blanks; if the =MASK> or =TABS> lines do contain data they are displayed, followed by the =COLS> line.

For more information about displaying and defining a profile, see "Displaying or defining an edit profile" on page 17.

Profile LOCK syntax



LOCK
Specifies that the current values in the profile are saved in the edit profile table and are not modified until the profile is unlocked. The current copy of the profile can be changed, either because of commands you enter that modify profile values (BOUNDS and NUMBER, for example) or because of differences in the data from the current profile settings. However, unless you unlock the edit profile, the saved values replace the changes when you end the edit session.

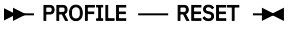
CAPS, NUMBER, STATS, and PACK mode are automatically changed to fit the data. These changes occur when the data is first read or when data is copied into the data set. Message lines (==MSG>) are inserted in the data set to show you which changes occurred.

Note: To force CAPS, NUMBER, STATS, or PACK mode to a particular setting, use an initial macro. Be aware, however, that if you set number mode on, data may be overlaid.

UNLOCK
Specifies that the editor saves changes to profile values.

See "Locking an edit profile" on page 19 for more information about locking and unlocking the profile.

Profile RESET syntax



RESET
Specifies that the ZDEFAULT profile is to be removed and the site-wide configuration for new edit profiles is to be used.

Description

To display the current edit profile:

1. On the command line, type:

   ```
   PROFILE number
   ```

2. Press Enter. The current edit profile appears.

To switch edit profiles or define a new edit profile without displaying the new profile:

1. On the command line, type:

   ```
   PROFILE name 0
   ```

   where *name* is the name of the edit profile to which you want to switch. This also specifies that no lines are to be displayed. If you want to display the new profile, you can omit the number or enter a number from 1 to 9.

2. Press Enter. The profile specified by the *name* operand becomes the active edit profile, but is not displayed if you entered 0. If the profile does not exist, an entry is created for it in the edit profile table, using the values of the current edit profile.

To lock the current edit profile:

1. On the command line, type:

   ```
   PROFILE LOCK
   ```

2. Press Enter. The values in the current edit profile are saved in the edit profile table. From this point on, any changes you make to the current edit profile affect only the current edit session. Values that were saved when the current profile was locked are used the next time you begin an edit session with this profile.

To unlock an edit profile:

1. On the command line, type:

   ```
   PROFILE UNLOCK
   ```

2. Press Enter. From this point on, any changes that you make to the current edit profile replace any values that may have been saved for this profile in the edit profile table. Also, these changes are saved when you end the current edit session.

Examples

Figure 141 on page 267 shows a typical edit profile for a REXX data set. The display results from entering PROFILE with no operands. The =TABS> and =MASK> lines appear because they contained data. If they had been empty, they would not have appeared.

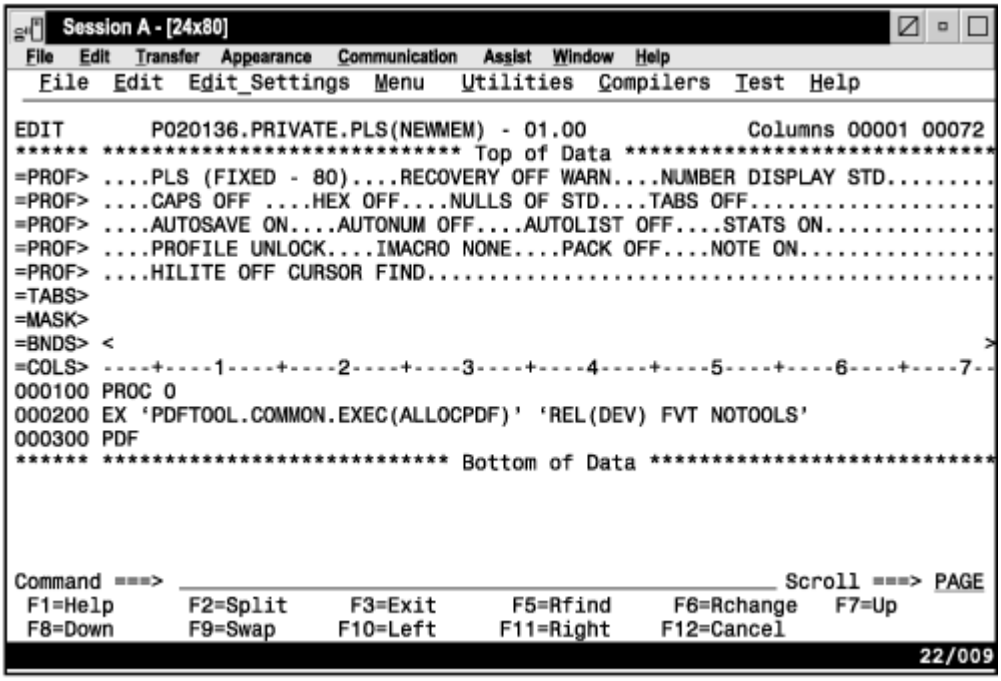


Figure 141. Edit Profile display

The sample profile contains:

- The first profile line (=PROF>) shows the profile name (EXEC), the data set record format and length (FIXED - 80), and the settings for edit recovery mode (RECOVERY ON) and number mode (NUMBER ON STD).
- The second profile line shows the settings for caps mode (CAPS ON), hexadecimal mode (HEX OFF), nulls mode (NULLS OFF), tabs mode (TABS OFF), and UNDO mode (SETUNDO STG).
- The third profile line shows the settings for the auto modes: autosave (AUTOSAVE ON), autonum (AUTONUM OFF), and autolist (AUTOLIST OFF). It also shows the setting for stats mode (STATS ON).
- The fourth profile line shows the lock status of the EXEC profile (PROFILE UNLOCK), the name, if any, of the initial macro called at the beginning of the edit session (IMACRO NONE), and the settings for pack mode (PACK OFF) and note mode (NOTE ON).
- The fifth profile line shows the current hilite status (HILITE OFF).
- The last four lines of the edit profile show the tabs settings (=TABS>), edit mask (=MASK>), bounds settings (=BNDS>), and the column position line (=COLS>).

RCHANGE—Repeat a Change

RCHANGE repeats the change requested by the most recent CHANGE command.

Syntax

Description

You can use this command to repeatedly change other occurrences of the search string. After a *string* NOT FOUND message appears, the next RCHANGE issued starts at the first line of the current range for a

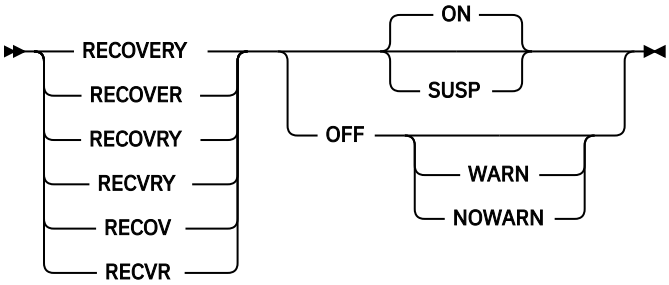
forward search (FIRST or NEXT specified) or the last line of the current range for a backward search (LAST or PREV specified).

Note: RCHANGE is normally assigned to a program function key, although you can issue it directly from the command line.

RECOVERY—Control Edit Recovery

RECOVERY sets edit recovery mode, which allows you to recover data after a system failure or power outage.

Syntax



ON
The system creates and updates a recovery data set for each change.

OFF
The system does not create and update a recovery data set.

WARN
This operand no longer has a practical function due to a software change. However, the primary command continues to accept the operand for compatibility reasons.

NOWARN
This operand no longer has a practical function due to a software change. However, the primary command continues to accept the operand for compatibility reasons.

SUSP
This operand functions the same as the ON operand.

Note: When SETUNDO is enabled during installation, both the RECOVERY primary command and edit macro command continue to accept the NOWARN and WARN keywords for compatibility reasons, but the value is ignored. NOWARN will always be in effect.

Description

You cannot edit data recursively while you are in recovery.

Attention:

If the data set to be recovered was edited by another user before edit recovery, the changes made by the other user will be lost if you save the recovered data.

See “Undoing edit interactions” on page 66 for more information.

To turn on edit recovery mode:

1. On the command line, type:

```
RECOVERY ON
```

RECOVERY can be abbreviated REC. This command can also ensure that your edit session is not lost due to a system failure.

2. Press Enter. The editor begins recording an audit trail of your interactions. After a system failure, the editor uses that record to reestablish the edit session at the time of failure.

Note: For edit recovery to work properly, the data set to be recovered, the edit recovery data set, and the edit recovery table all must exist, be cataloged, and be intact. For example, with RECOVERY on, uncataloging a data set and then trying to recover it fails.

To turn off edit recovery mode:

1. On the command line, type:

```
RECOVERY OFF
```

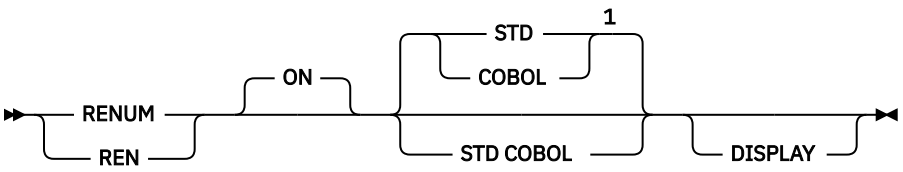
2. Press Enter. The editor stops recording your interactions. Edit recovery is not available following a system failure. When an edit session is recovered, the data is scrolled all the way to the left when the recovery edit session begins.

See "Edit recovery" on page 38 for more information about edit recovery.

RENUM—Renumber Data Set Lines

RENUM immediately turns on number mode and renumbers all lines, starting with number 100 and incrementing by 100. For members exceeding 10 000, the increment is less than 100.

Syntax



Notes:

[1] STD is the default for non-COBOL data set types. COBOL is the default for COBOL data set types.

ON
Automatically verifies that all lines have valid numbers in ascending sequence and renumbers any lines that are either unnumbered or out of sequence. It also turns number mode on and renumbers lines.

The STD, COBOL, and DISPLAY operands are interpreted only when number mode is turned on.

STD
Numbers the data in the standard sequence field. This is the default for all non-COBOL data set types.

COBOL
Numbers the data in the COBOL field. This is the default for all COBOL data set types.

Attention:

If number mode is off, make sure the first 6 columns of your data set are blank before using either the NUMBER ON COBOL or NUMBER ON STD COBOL command. Otherwise, the data in these columns is replaced by the COBOL sequence numbers. If that happens and if edit recovery or SETUNDO is on, you can use the UNDO command to recover the data. Or, you can use CANCEL at any time to end the edit session without saving the data.

STD COBOL
Numbers the data in both fields.

If both STD and COBOL numbers are generated, the STD number is determined and then used as the COBOL number. This can result in COBOL numbers that are out of sequence if the COBOL and STD fields are not synchronized. Use RENUM to synchronize them.

DISPLAY
Causes the width of the data window to include the sequence number fields. Otherwise the width of the window does not include the sequence number fields. When you display a data set with a logical record length of 80 and STD numbering, the sequence numbers are not shown unless you are using a 3278 Model 5 terminal, which displays 132 characters. The editor automatically scrolls left or right, if required, so that the left most column of the data window is the first column to appear.

Description

To renumber all lines using the standard sequence fields only:

```
RENUM STD
```

To renumber all lines using both the standard and COBOL sequence fields:

```
RENUM STD COBOL
```

To renumber all lines using the COBOL sequence fields only:

```
RENUM COBOL
```

To renumber all lines using both the standard and COBOL sequence fields and specifying that the data window is to include the sequence number fields:

```
RENUM STD COBOL DISPLAY
```

To renumber all lines by using the standard sequence fields only and specifying that the data window is to include the sequence number fields:

```
RENUM DISPLAY
```

Here, the DISPLAY operand is the only operand needed because STD is the default.

Examples

In Figure 142 on page 271, the line numbers are not incremented uniformly. Type RENUM on the command line. Figure 143 on page 271 shows how the lines are renumbered after you press Enter.

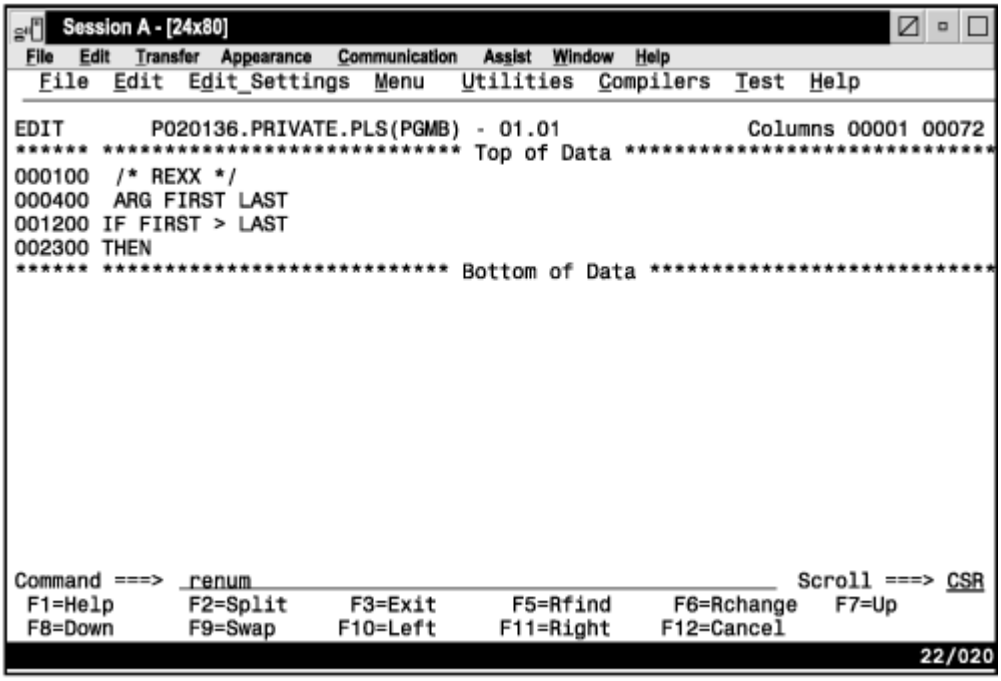


Figure 142. Member before lines are renumbered

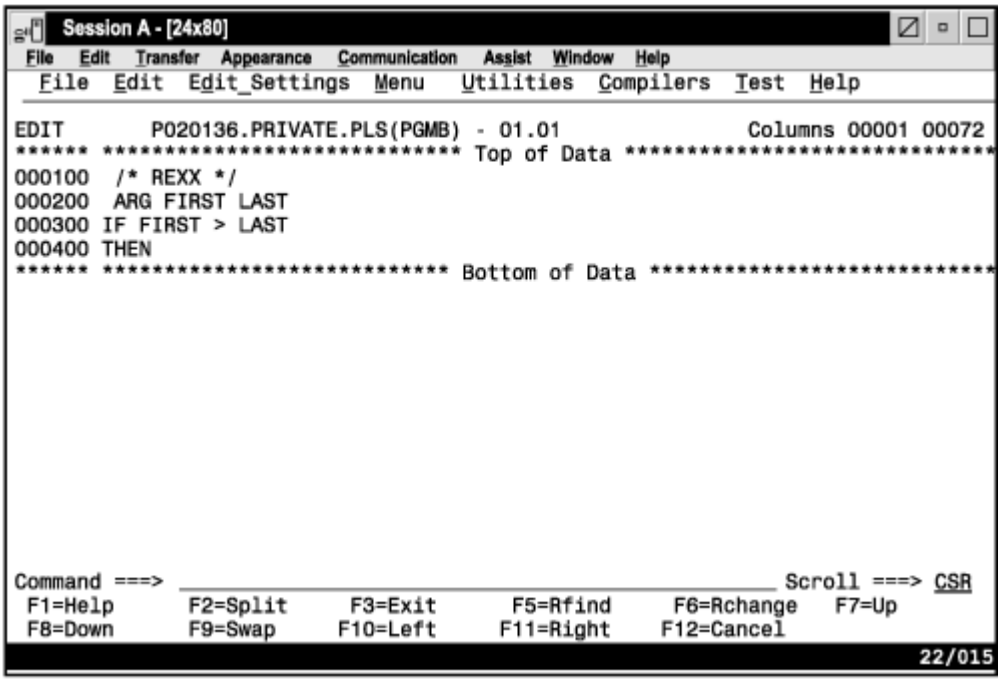


Figure 143. Member after lines are renumbered

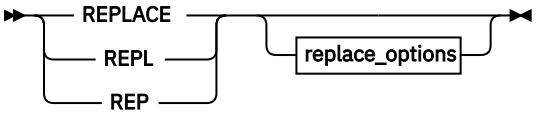
REPLACE—Replace Data

The REPLACE primary command replaces a sequential data set, member of a partitioned data set, or z/OS UNIX file with data you are editing. If a member or z/OS UNIX file you want to replace does not exist, the editor creates it. If a member you want to replace exists and the member is in a PDSE version 2 data set that is configured for member generations, the editor creates a new generation of the member. This new

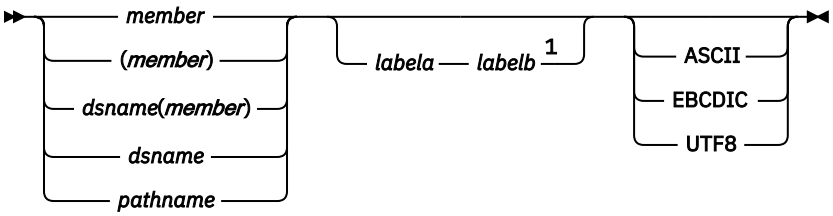
generation becomes the current generation (also known as generation zero). The editor cannot create a new sequential data set.

If no options are specified with the REPLACE command, the Edit/View Replace panel is displayed.

Syntax



replace_options



Notes:

[1] If you don't specify the group of lines using labels, you must specify the group by using C or M line commands.

member
The name of the member to be replaced in the partitioned data set currently being edited. If a name of eight characters or fewer is specified and it could be a member name or a data set name, REPLACE searches for a member name first. If no member is found, then the name is used as a data set name. If the member does not exist, the editor creates it. If you are using a concatenated sequence of libraries, the editor writes the member to the first library in the sequence. This operand is optional.

To replace a sequential data set or a member of a different partitioned data set, enter REPLACE without a member operand. The editor displays the Edit Replace panel, from which you can enter the data set name.

dsname
A partially qualified or fully qualified sequential data set you want to replace.

pathname
The pathname for a z/OS UNIX regular file to be replaced. If the file does not exist, it is created. (Also, see “Specifying z/OS UNIX pathnames with edit primary and macro commands” on page 16.)

***dsname*(*member*)**
A partially qualified or fully qualified partitioned data set and member you want to replace.

labela*, *labelb
Labels identifying the start and end of the group of lines to replace the member or data set.

For more information about using labels to identify a group of lines, see “Labels and line ranges” on page 59.

ASCII, EBCDIC, UTF8
When one of these keywords is supplied, if the data is using a different character set to that designated by the keyword, the data being replaced in the external file is converted to the character set designated by the keyword.

Description

To replace a member of a partitioned data set, a sequential data set, or a z/OS UNIX file:

1. On the command line, type one of these commands:

```
REPLACE member labela labelb
REPLACE (member) labela labelb
REPLACE dsname labela labelb
REPLACE dsname(member) labela labelb
REPLACE pathname labela labelb
```

   The *member* operand is optional unless you specify the name of a partitioned data set. It represents the name of the member that you want to replace. If you specify a data set name only, it must be a sequential data set.

   The *labela* and *labelb* operands are optional, also. They represent a pair of labels that show the first and last lines in a group of lines used to replace the member.

   If you omit the *labela* and *labelb* operands, you must specify the lines by using either the C (copy) or M (move) line command. See the descriptions of these commands if you need more information about them.

   If you omit the *labela* and *labelb* operands, and do not enter one of the preceding line commands, a "REPLACE Pending" message is displayed in the upper-right corner of the panel.

2. Press Enter. If you did not specify the name of a member, data set, or z/OS UNIX file, the Edit/View Replace panel is displayed. Enter the name of the member, data set, or z/OS UNIX file to be replaced on this panel and press Enter again. If you used either a pair of labels or a C line command, the data is copied from the member, data set, or z/OS UNIX file that you are editing into the member, data set, or z/OS UNIX file that you are replacing. If you used the M line command, however, the data is removed from the member, data set, or z/OS UNIX file that you are editing and placed in the member, data set, or z/OS UNIX file that you are replacing.

   If the data set specified does not exist, ISPF prompts you to see if the data set should be created. You can create the data set using the characteristics of the cataloged source data set as a model, or specify the characteristics for the new data set. You can suppress this function through the ISPF configuration table, causing any CREATE request for a nonexistent data set to fail.

See “Creating and replacing data” on page 41 for more information about the REPLACE command.

Examples

These steps show how you can replace a member when you omit the member name. These same steps apply when you create data.

1. Type REPLACE and specify which lines you want to copy or move into the data set or member. The example in Figure 144 on page 274 uses the MM (block move) line command to move a block of lines from the data.

```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
_____________________________________________________________________
EDIT       USERSID.TEST(FROMDATA) - 01.02               Columns 00001 00072
Command ===> replace                                    Scroll ===> PAGE
****** ***************************** Top of Data ******************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
000100 This line will be left in this member
000200 This line will be left in this member
MM0300 +----------------+
000400 | This is the    |
000500 | material to    |
000600 | be created in  |
000700 | another member |
MM0800 +----------------+
000900 This line will be left in this member
001000 This line will be left in this member
****** **************************** Bottom of Data ****************************



 F1=Help      F2=Split     F3=Exit      F5=Rfind     F6=Rchange   F7=Up
 F8=Down      F9=Swap     F10=Left     F11=Right    F12=Cancel
```

Figure 144. Member before other member is replaced

2. When you press Enter, the Edit/View Replace panel (Figure 145 on page 274) appears. Type the name of the member to be replaced and press Enter. A member is created when you type the name of a member that does not already exist. The name of the member replaced in this example is DELDATA.

```
  Menu  RefList  Utilities  Help
_____________________________________________________________________
                              Edit/View Replace
Command ===> ____________________________________________________________

"Current" Data Set: USERSID.TEST(FROMDATA)

To ISPF Library:
   Project . . . PDFTDEV
   Group . . . . USERSID  . . .  _______ . . . ________ . . . ________
   Type  . . . . MSGGEN
   Member  . . . ________

To Other Sequential Data Set, Partitioned Data Set Member, or z/OS UNIX file:
   Name . . . . . TEST(DELDATA)_____________________________________________  +
   Volume Serial . . ______    (If not cataloged)

Data Set Password  . .           (If password protected)

Enter "/" to select option             Data Conversion option
_ Pack "Replace" Data Set              _ 1. EBCDIC
                                         2. ASCII
                                         3. UTF-8

Press ENTER key to replace. Enter END command to cancel replace.
 F1=Help      F2=Split     F3=Exit      F7=Backward  F8=Forward   F9=Swap
F10=Actions  F12=Cancel
```

Figure 145. Edit/View Replace panel (ISRERPL1)

3. Figure 146 on page 275 shows the lines remaining in the data being edited after the specified lines were moved.

```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
------------------------------------------------------------------------------
EDIT       USERSID.TEST(FROMDATA) - 01.03               Member DELDATA created
Command ===> _____________________________________________ Scroll ===> PAGE
****** ***************************** Top of Data ******************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
000100 This line will be left in this member
000200 This line will be left in this member
000900 This line will be left in this member
001000 This line will be left in this member
****** **************************** Bottom of Data ****************************









 F1=Help      F2=Split     F3=Exit      F5=Rfind     F6=Rchange   F7=Up
 F8=Down      F9=Swap     F10=Left     F11=Right    F12=Cancel
```

Figure 146. Member after the other member has been replaced

4. Figure 147 on page 275 shows the contents of the replaced member.

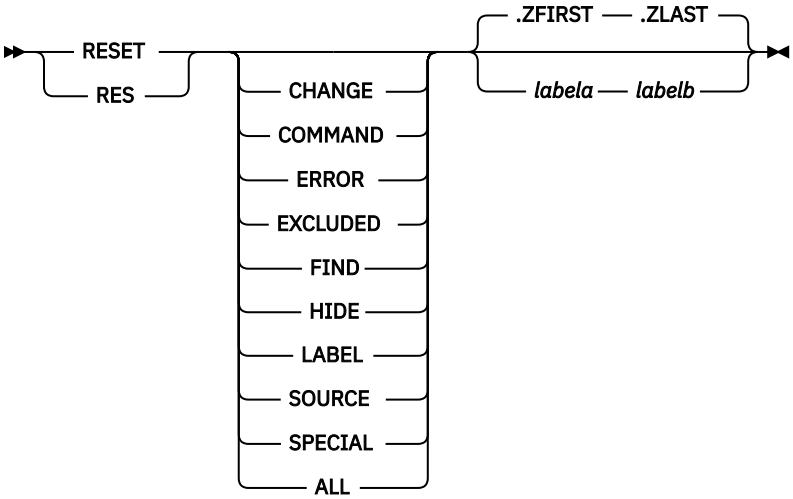
```
  File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
------------------------------------------------------------------------------
EDIT       USERSID.TEST(DELDATA) - 01.02                  Columns 00001 00072
Command ===> _____________________________________________ Scroll ===> PAGE
****** ***************************** Top of Data ******************************
==MSG> -Warning- The UNDO command is not available until you change
==MSG>           your edit profile using the command RECOVERY ON.
000300 +----------------+
000400 | This is the    |
000500 | material to    |
000600 | be created in  |
000700 | another member |
000800 +----------------+
****** **************************** Bottom of Data ****************************









 F1=Help      F2=Split     F3=Exit      F5=Rfind     F6=Rchange   F7=Up
 F8=Down      F9=Swap     F10=Left     F11=Right    F12=Cancel
```

Figure 147. Other member replaced

RESET—Reset the Data Display

The RESET primary command can restore line numbers in the line command field when those line numbers have been replaced by labels, pending line commands, error flags, and change flags. RESET can also delete special lines from the display, redisplay excluded lines and excluded lines messages, and temporarily disable the highlighting of FIND strings.

Syntax



You can type the operands in any order. If you do not specify any operands, RESET processes all operands except LABEL.

CHANGE
Removes ==CHG> flags from the line command field.

COMMAND
Removes any pending line commands from the line command field.

ERROR
Removes ==ERR> flags from the line command field.

EXCLUDED
Redisplays any excluded line.

FIND
Turns off highlighting of FIND strings until the next FIND, RFIND, CHANGE, or RCHANGE command. SEEK and EXCLUDE do not return the highlighting of FIND strings in this manner.

The resetting of FIND highlighting does not honor the range specified on the RESET command.

HIDE
Redisplays all "*n* Line(s) not Displayed" messages for excluded lines that were hidden through the HIDE command.

LABEL
Removes labels from the line command field.

SOURCE
Revert back from ASCII editing mode to EBCDIC editing mode such that the data is not translated from or to ASCII when displaying and receiving input from the terminal.

SPECIAL
Deletes any temporary line from the panel:

- Bounds line flagged as =BNDS>
- Column identification lines flagged with =COLS>
- Information lines flagged with ======
- Mask lines flagged as =MASK>
- Message lines flagged as ==MSG>
- Note lines flagged with =NOTE=
- Profile lines flagged as =PROF>
- Tabs line flagged as =TABS>

ALL
Removes all changes to the line number field.

labela*, *labelb
Labels identifying the start and end of the group of lines to be reset.

For more information about using labels to identify a group of lines, see “Labels and line ranges” on page 59.

Description

RESET scans every line of data. If you want to delete a small number of special lines, you can get faster response time if you use the D (delete) line command.

Examples

To reset all lines except those that contain labels:

```
RESET
```

To reset only the lines that contain labels:

```
RESET LABEL
```

To reset only the lines that contain pending line commands:

```
RESET COMMAND
```

To reset only the lines that contain ==ERR> flags:

```
RESET ERROR
```

To reset only the lines that contain ==CHG> flags:

```
RESET CHANGE
```

To reset only the special (temporary) lines:

```
RESET SPECIAL
```

To reset only the excluded lines:

```
RESET EXCLUDED
```

To reset only the excluded lines messages:

```
RESET HIDE
```

To reset all lines between and including the .START and .STOP labels, except those that contain labels:

```
RESET .START .STOP
```

RFIND—Repeat Find

RFIND locates the search string defined by the most recent SEEK, FIND, or CHANGE command, or excludes a line containing the search string defined by the previous EXCLUDE command.

RFIND can be used repeatedly to find other occurrences of the search string. After a "*string* NOT FOUND" message is displayed, the next RFIND issued starts at the first line of the current range for a forward

search (FIRST or NEXT specified), or the last line of the current range for a backward search (LAST or PREV specified).

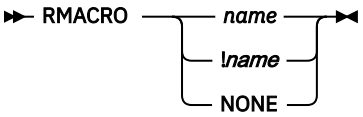
Syntax

Note: RFIND is normally assigned to a program function key, although you can issue it directly from the command line.

RMACRO—Specify a Recovery Macro

RMACRO saves the name of a recovery macro in the edit profile.

Syntax



name
The name of the recovery macro to be run. The name can be preceded by an exclamation point (!) to show that it is a program macro.

NONE
The name to prevent a recovery macro from being run.

Description

To specify the name of a recovery macro:

1. On the command line, type:

   ```
   RMACRO name
   ```

   where name is the name of the recovery macro that you want to run.
2. Press Enter.

See “Recovery macros” on page 109 for more information.

Examples

To define RESTART as the recovery macro, type:

```
RMACRO RESTART
```

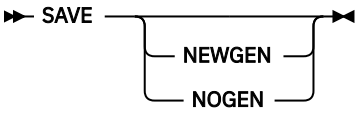
To reset the profile with no recovery macro, type:

```
RMACRO NONE
```

SAVE—Save the Current Data

SAVE saves edited data without ending your edit session. Generally, you do not need to use SAVE if recovery mode is on. See AUTOSAVE, CANCEL, and END for more information about saving data.

Syntax



NEWGEN
Applicable only when editing a member in a PDSE version 2 data set that is configured for member generations. Saves the member in a new generation. This new generation becomes the current generation, also known as generation zero. The generation being edited is left unchanged. This is the default behavior when editing the current generation.

NOGEN
Applicable only when editing a member in a PDSE version 2 data set that is configured for member generations. Saves the member to the same generation that is being edited. This is the default behavior when editing a non-current generation.

Note: The default SAVE behavior when editing a non-current member generation can be changed in the ISPF site configuration table.

Description

SAVE writes the data to the same data set from which it was retrieved unless you specified a concatenated sequence of partitioned data sets on the Edit Entry panel. In that case, the data is saved in the first library in the concatenation sequence, regardless of from which library it came. For a sequential data set, the complete data set is rewritten. For a partitioned data set, the member is rewritten with the same member name.

For a member in a PDSE version 2 data set that is configured for member generations, the behavior depends on the member generation being edited:

- When editing the current generation, also known as generation zero, the default behavior is to write the member to a new generation.
- When editing a non-current generation, the default behavior is to write the member to the same generation that is being edited. The default behavior when editing a non-current generation can be changed in the ISPF site configuration table.
- These default behaviors for member generations can be overridden using the NEWGEN and NOGEN keywords.

If stats mode is on, the library statistics for the member are automatically updated.

If both number mode and autonum mode are on, the data is automatically renumbered before it is saved.

If SAVE cannot successfully rewrite the data because of I/O errors or insufficient space, the system displays a message in the upper-right corner of the panel, accompanied by an audible alarm, if installed. You can then try to save the data in another data set by taking these steps:

1. Enter CREATE or REPLACE with no operand on the command line. Use CREATE only if the destination is a member of a partitioned data set, such as an ISPF library member.
2. Type CC on the first and last data lines to specify that all lines are to be copied. Then press Enter.
3. Fill in the data set and member name of the alternate library on the Edit Create or Edit Replace panel, and press Enter.

When a space ABEND such as D37 occurs, ISPF deallocates the data set so that you can swap to another screen or user ID and reallocate the data set. This does not occur for data sets that were edited using the DDNAME parameter of the EDIT service.

See “Creating and replacing data” on page 41 for more information.

Examples

To save the data in the data set or member that you are editing:

1. On the command line, type:

```
SAVE
```

2. Press Enter.

When you are editing generation zero of a member in a PDSE version 2 data set and you want the data to be saved to the same generation (rather than create a new generation):

1. On the command line, type:

```
SAVE NOGEN
```

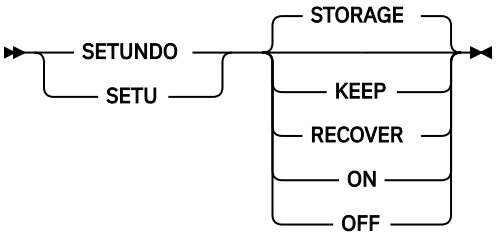
2. Press Enter.

SETUNDO—Set the UNDO Mode

The SETUNDO primary command determines whether the UNDO command is available and how the history of changes should be managed.

Note: The SETUNDO command is ignored if UNDO from storage is not enabled by the installer or person who maintains the ISPF product. For information on enabling UNDO from storage, see *z/OS ISPF Planning and Customizing*.

Syntax



STORAGE
Enables the saving of edit changes in storage. If the setting is changed, and the profile lines are displayed, the profile lines show the value (SETUNDO STG) after the change. Valid abbreviations for STORAGE are STO, STG, STOR and STORE.

KEEP
Has the same effect as STORAGE except the UNDO buffers are not cleared when a SAVE is issued.

Note: The effect of KEEP (UNDO buffers not cleared when a SAVE is issued) ceases if SETUNDO is subsequently issued without the KEEP keyword.

RECOVER
Enables the saving of edit changes through the recovery file only. If recovery is off, it is turned on by this command. If the setting is changed and the profile lines are displayed, the profile lines show the value (SETUNDO REC) after the change. A valid abbreviation for RECOVER is REC.

ON
The same as STORAGE.

OFF
Disables the saving of edit changes in storage. If SETUNDO OFF is specified and recovery is on, then a state of SETUNDO RECOVER is set and UNDO is available from the recovery file. All transactions on the storage UNDO chain are removed, and no changes before SETUNDO OFF can be undone (unless

RECOVERY ON is specified). If the setting is changed and the profile lines are displayed, the profile lines show the value (SETUNDO OFF or SETUNDO REC) after the change.

Description

SETUNDO allows you to specify how changes you make during your edit session are to be recorded and used by the UNDO command. UNDO can be run when either SETUNDO or RECOVERY is on. Changes can be recorded in storage, in the recovery file, or in both places. Saving the changes in storage only is the fastest method.

To enable recording in storage:

1. On the command line, type one of these commands:
   - `SETUNDO STORAGE`
   - `SETUNDO KEEP`
   - `SETUNDO`
2. Press Enter.

The value of ON is accepted to complement the OFF state.

To use the recovery file:

1. On the command line, type:

   ```
   SETUNDO RECOVER
   ```

2. Press Enter.

If RECOVERY is off, it is turned on by this command.

To turn off recording and disable the UNDO command, enter:

```
SETUNDO OFF
```

Note: If recovery is on, setting SETUNDO OFF is the same as specifying SETUNDO REC, and the recovery file is used for UNDO.

Examples

The edit profile shown in Figure 148 on page 282 shows SETUNDO set to STORAGE and RECOVERY OFF.

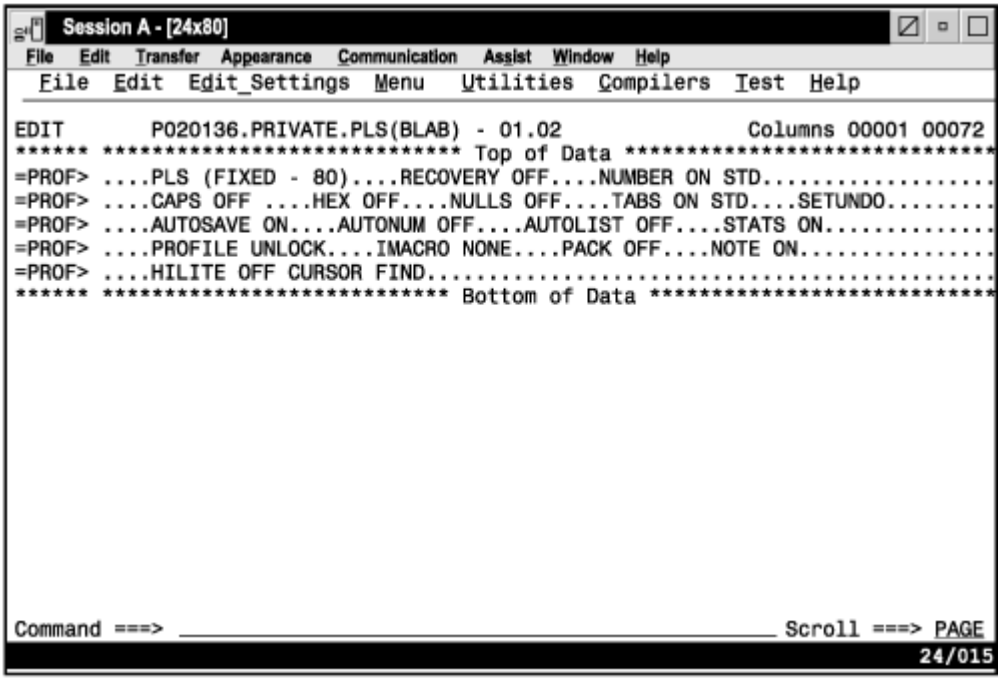
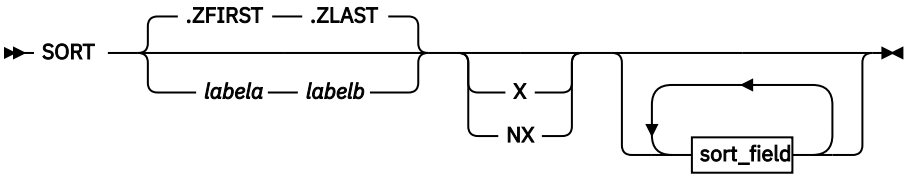


Figure 148. SETUNDO STORAGE and RECOVERY OFF

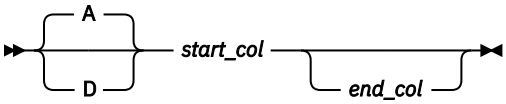
SORT—Sort Data

The SORT primary command puts data in a specified order.

Syntax



sort_field



labela, labelb
Labels identifying the start and end of the group of lines to be sorted.

For more information about using labels to identify a group of lines, see “Labels and line ranges” on page 59.

X
Sorts only lines that are excluded.

NX
Sorts only lines that are not excluded.

sort_field
Specifies the field to be used in sorting data. You can specify up to five sort fields using these operands:

A
Specifies ascending order. It can either precede or follow the column specification.

D
Specifies descending order. It can either precede or follow the column specification.

start_col
Defines the starting column of the field that is to be compared. It must be within the current boundaries.

end_col
Defines the ending column of the field that is to be compared. It must be within the current boundaries. If it is not supplied, then the ending column is the current right boundary. For more information on boundaries, see "Edit boundaries" on page 23.

If you specify several fields, you must specify both the starting and ending columns of each field. The fields cannot overlap. If you supply the sort order for one field, you must supply it for all fields.

Description

SORT operates in two different modes, based on the hexadecimal mode status. If hexadecimal mode is on, the data is ordered according to its hexadecimal representation. If hexadecimal mode is off, data is sorted in the collating sequence defined for the national language being used.

Sorting data without operands

For SORT with no operands, the editor compares the data within the current boundaries character by character, and then orders it line by line in the proper collating sequence. It ignores data outside the current boundaries during both operations. Therefore only the data inside the current boundaries is changed. Labels, excluded lines, line numbers, and change, error, and special line flags are considered associated with the data, and therefore point to the same data fields after the sort as they did before the sort.

For example, if you issue a CHANGE ALL that changes the first, third, and sixth lines in a data set, these lines are flagged with the change flag, ==CHG>. If you then issue a SORT command that results in the former lines 1, 3, and 6 becoming the first, second and third lines of the sorted file, the changed line flags would now exist on the first, second and third lines of the sorted data set.

It is important to properly set the boundaries before issuing SORT. SORT is a powerful tool for editing data that may be formatted in multiple columns. You can set the boundaries, for example, to the first half of a record and sort one column of data. Then you can set the boundaries to the last half of the record and sort a second column of data.

Limiting the SORT command

Sorting is limited to data within the current boundaries. You can specify up to five sort fields by labeling starting and ending columns. You can also identify each field as having data sorted in either ascending or descending order.

Optionally, you can limit sorting to a range of lines by specifying the labels of the first and last lines of the range. You can also limit sorting to either excluded or non-excluded lines.

If you have labels or line ranges that are between the labels or line ranges specified with SORT, you can keep SORT from rearranging them by:

- Excluding them before you enter SORT
- Using the NX operand to sort only lines that are not excluded

For more information, see the definition of the NX operand and "EXCLUDE—Exclude Lines from the Display" on page 232.

Sorting DBCS data

When sorting data that contains DBCS character strings, you must ensure that no DBCS string crosses the boundaries. Also, all records must have the same format at the boundaries, although the format of the left and right boundaries can differ.

If a boundary divides a DBCS character, or if all records do not have the same format at the boundaries, the result is unpredictable.

Examples

This form of the SORT command sorts in ascending order. The start-column is the left boundary and the end-column is the right boundary:

```
SORT
```

This form of the SORT command sorts in descending order. The start-column is the left boundary and the end-column is the right boundary:

```
SORT D
```

This form of the SORT command sorts in ascending order. The start-column is column 5 and the end-column is the right boundary:

```
SORT 5
```

This form of the SORT command sorts in descending order. The start-column is column 5 and the end-column is the right boundary:

```
SORT 5 D
```

SOURCE—describe format of data

The SOURCE primary command instructs the editor to treat the source data as though it is in the specified format and converts it from that format to the CCSID of the terminal for display purposes, although the data remains unchanged within the file. When you input or modify data at the terminal, the editor translates the data entered from the CCSID of the terminal to the specified format prior to storing the data in the file.

Syntax

The SOURCE ASCII primary command is not available when editing a z/OS UNIX file. Instead, use the ASCII edit facility to have the data automatically translated from ASCII to the CCSID of the terminal.

character_encoding
The type of character encoding to be used for translating data when displaying or receiving input from the terminal.

Valid values are:

- ASCII

See “Working with ASCII data” on page 50 for more information.

Examples

To set source mode to ASCII:

```
SOURCE ASCII
```

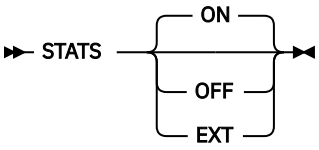
To revert back to normal mode, use the RESET command:

```
RESET SOURCE
```

STATS—Generate Library Statistics

The STATS primary command sets stats mode, which creates and maintains statistics for a member of a partitioned data set.

Syntax



ON
Creates or updates library statistics when the data is saved.

If extended statistics are enabled in the site configuration and any of the line number statistic values exceed 65535, the statistics are automatically stored as extended statistics. Otherwise, the statistics are automatically stored as non-extended statistics. Extended statistics contain extended line count fields that can store values up to 2147483647; non-extended statistics do not contain the extended line count fields and can only store line number values up to 65535. If extended statistics are not enabled in the site configuration, 65535 is stored for line number statistic values that exceed 65535.

OFF
Does not create or update library statistics.

If STATS mode is off when you save a member, any previous statistics are lost.

EXT
Has the same function as ON.

See “Statistics for PDS members” on page 25 for more information.

Examples

To set stats mode on:

```
STATS ON
```

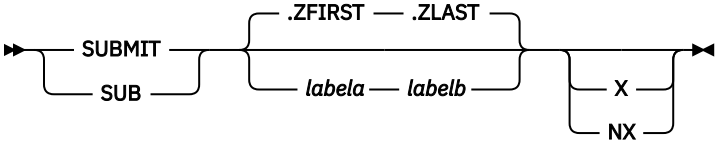
To set stats mode off:

```
STATS OFF
```

SUBMIT—Submit Data for Batch Processing

The SUBMIT primary command submits the member or data set you are editing (or the part of the member or data set defined by the range of line pointers or the X or NX parameters) to be processed as a batch job.

Syntax



labela*, *labelb
Labels identifying the start and end of the group of lines to be submitted.

For more information about using labels to identify a group of lines, see "Labels and line ranges" on page 59.

X
Submits only lines that are excluded from the display.

NX
Submits only lines that are not excluded from the display.

Description

The editor does not supply a job statement when you enter the SUBMIT command. You can supply job statements as part of the data being submitted. When you supply a job statement, only the job name is logged to the ISPF log data set to ensure the protection of sensitive data.

If the file being edited is described as ASCII or UTF-8 then the data submitted to the internal reader is converted to EBCDIC.

ISPF uses the TSO SUBMIT command to submit the job.

Examples

To submit lines between labels .START and .END as a batch job:

```
SUBMIT .START .END
```

To submit all of the data as a batch job:

```
SUBMIT
```

To submit only non-excluded lines as a batch job:

```
SUBMIT NX
```

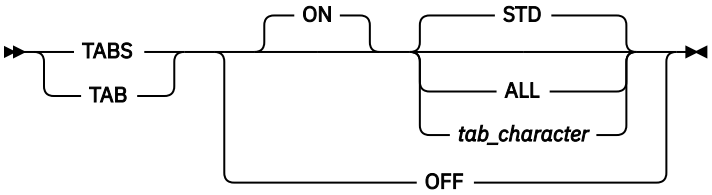
TABS—Define Tabs

The TABS primary command:

- Turns tabs mode on and off
- Defines the logical tab character
- Controls the insertion of attribute bytes at hardware tab positions defined with TABS

Use PROFILE to check the setting of tabs mode and the logical tab character. See "Using tabs" on page 63 if you need more information about using tabs.

Syntax



ON
Turns tabs mode on, which means that logical tabs can be used to break up strings of data. This is the default operand. If no other operands are included, all hardware tab positions (asterisks) that contain a blank or null character are activated because STD is also a default operand. The "TABS ON STD" message is displayed in the profile.

OFF
Turns tabs mode off, which means that logical tabs cannot be used. Attribute bytes are deleted from all hardware tab positions, causing the Tab Forward and Tab Backward keys to ignore hardware tabs defined on the =TABS> line. Blanked-out characters that occupy these positions reappear. The "TABS OFF" message is displayed in the profile.

STD
Activates all hardware tab positions (asterisks) that contain a blank or null character. The editor inserts attribute bytes, which cannot be typed over, at these positions. STD is the default operand. You can use the Tab Forward and Tab Backward keys to move the cursor one space to the right of the attribute bytes. The "TABS ON STD" message is displayed in the profile.

ALL
Causes an attribute byte to be inserted at all hardware tab positions. Characters occupying these positions are blanked out and the attribute bytes cannot be typed over. The Tab Forward and Tab Backward keys can be used to move the cursor one space to the right of these attribute bytes. The "TABS ON ALL" message is displayed in the profile.

tab_character
Defines a single character that is not a number, letter, or command delimiter as the logical tab character. This character is used with hardware tab definitions. The "TABS ON *tab_character*" message is displayed in the profile.

You can enclose the character in quotes (' or "), although this is not necessary unless a quote or a comma (,) is used as the tab character.

The *tab_character* operand causes the data string that follows the logical tab character to align itself one space to the right of the first available hardware tab position when you press Enter. No attribute bytes are inserted.

If no hardware tabs are defined, the editor aligns the data vertically. If software tabs are defined, the first data string is aligned under the first software tab position and the remaining data strings are aligned at the left boundary. If neither software nor hardware tabs are defined, the editor aligns all the data strings at the left boundary.

With the *tab_character* operand, the Tab Forward and Tab Backward keys ignore hardware tab positions because no attribute bytes are inserted.

You can type the operands in any order, but keep these rules in mind:

- The *tab_character* and ALL operands cannot be used together, because the *tab_character* operand does not allow ISPF to insert attribute bytes at tab positions, while the ALL operand does.
- The TABS primary command has no effect on software tabs. Whenever software tabs are defined, you can always press Enter to move the cursor to a software tab position in the data, even if tabs mode is off. Attribute bytes are not inserted at software tab positions.

Examples

Define the number sign (#) as a logical tab character by typing this command and pressing Enter:

```
TAB #
```

Now, enter the COLS line command by typing C0LS in the line command field and pressing Enter. A partial =COLS> line with positions 9 through 45 is shown in the example.

To use the logical tab character you have defined (#), you also need at least one hardware tab. For this example, we will assume that three hardware tabs have already been defined in columns 20, 30, and 40:

```
=COLS> -1----+----2----+----3----+----4----+
=TABS>            *         *         *
```

If you then type this information on a line:

```
#$4237#$ 596#$  81
```

the data $4237 is repositioned after the first tab column, defined by an * in the =TABS line, when you press Enter. The $ 596 is repositioned after the next tab column and so forth, as follows:

```
=COLS> -1----+----2----+----3----+----4----+
=TABS>            *         *         *
                   $4237     $ 596     $  81
```

UNDO—Reverse Last Edit Interaction

The UNDO primary command allows you to remove the data modifications of a previous interaction.

Note: The SETUNDO command is ignored if UNDO from storage is not enabled by the installer or person who maintains the ISPF product. For information on enabling UNDO from storage, see *z/OS ISPF Planning and Customizing*.

Syntax

Description

Each time you enter UNDO, it reverses edit interactions, one at a time, in the order in which they have been entered. To use UNDO, you must have either RECOVERY on or SETUNDO on. You can undo only those changes made after RECOVERY or SETUNDO was turned on. SETUNDO and RECOVERY can be specified in your edit profile. You can also use the edit macro command ISREDIT SETUNDO to turn UNDO processing on and off. See “SETUNDO—Set UNDO Mode” on page 404 for more information.

RECOVERY is now optional and is not required to run UNDO. Performance improves if the editor is run with SETUNDO STORAGE and RECOVERY OFF. In this mode, non-data changes, such as setting line labels, adding note lines, and inserting blank lines, can be undone by UNDO even if no data changes have been made. With RECOVERY ON, only changes made after (and including) the first change to edit data can be undone.

Note: Changes made by initial edit macros cannot be undone.

See “Understanding differences in SETUNDO processing” on page 67 for more information on the differences between SETUNDO RECOVER and SETUNDO STORAGE processing.

Each time you press Enter, an interaction occurs between you and ISPF. If you combine line and primary commands in one entry, ISPF considers this one interaction. Therefore, UNDO would cause all of the

commands to be reversed. ISPF also considers running edit macros that contain a combination of macro commands and assignment statements, while entering a combination of edit line and primary commands at the same time, as one interaction.

Profile changes, such as HEX ON, LEVEL, and CAPS, cannot be undone separately. Profile changes are associated with the data change that came before them, and can be undone only when preceded by a data change. The data change and the profile change are undone at the same time. For example, if you make a change to the data, change the version number, set caps off, turn hex on, and then enter UNDO, the version number, caps setting, and hex mode all revert to the way they existed before the data change. The data change is also undone.

Note: UNDO is not accepted if any line commands or data changes are also specified since it would be unclear what is to be undone.

To undo the last changes:

1. Type on the command line:

```
UNDO
```

2. Press Enter.

Note: UNDO is reset by SAVE. Once you save your data for the current edit session, you can no longer recover any interactions made before the data was saved.

Failures in recovery processing due to I/O errors no longer terminate the UNDO function if SETUNDO STORAGE is active. When UNDO is processed, the editor scrolls the data all the way to the left.

See “Undoing edit interactions” on page 66 for more information.

Examples

You are editing the member shown in Figure 149 on page 289 and decide to delete all of the lines. You have type the block form of the D (DELETE) command in the line command field.

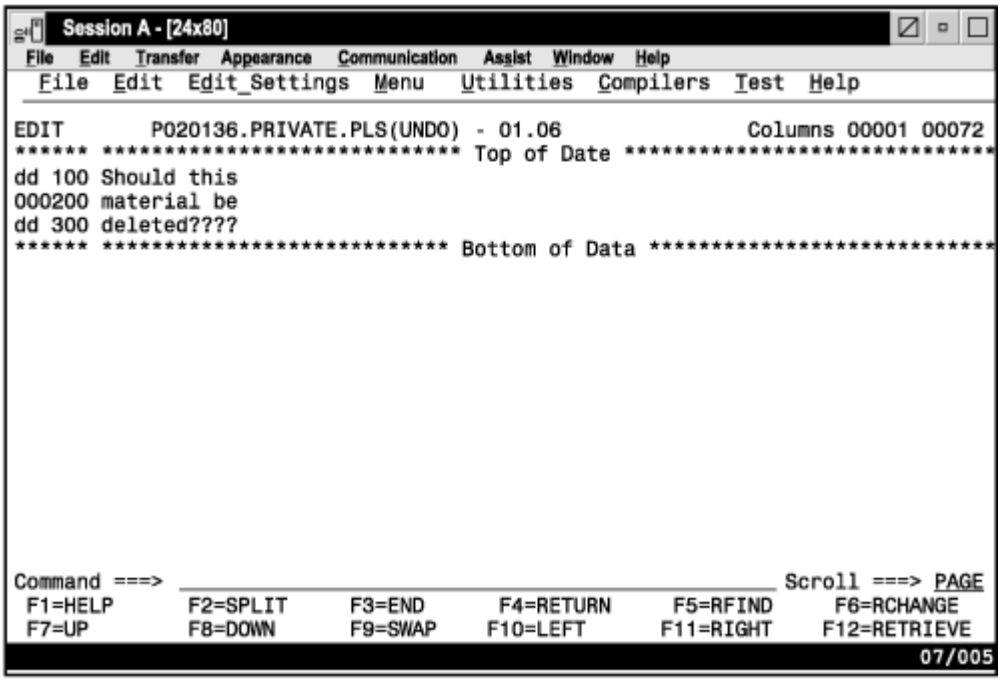


Figure 149. Member before lines are deleted

Figure 150 on page 290 shows the member after the lines have been deleted. However, you have changed your mind and want to put the lines back again. Therefore, type UNDO on the command line.

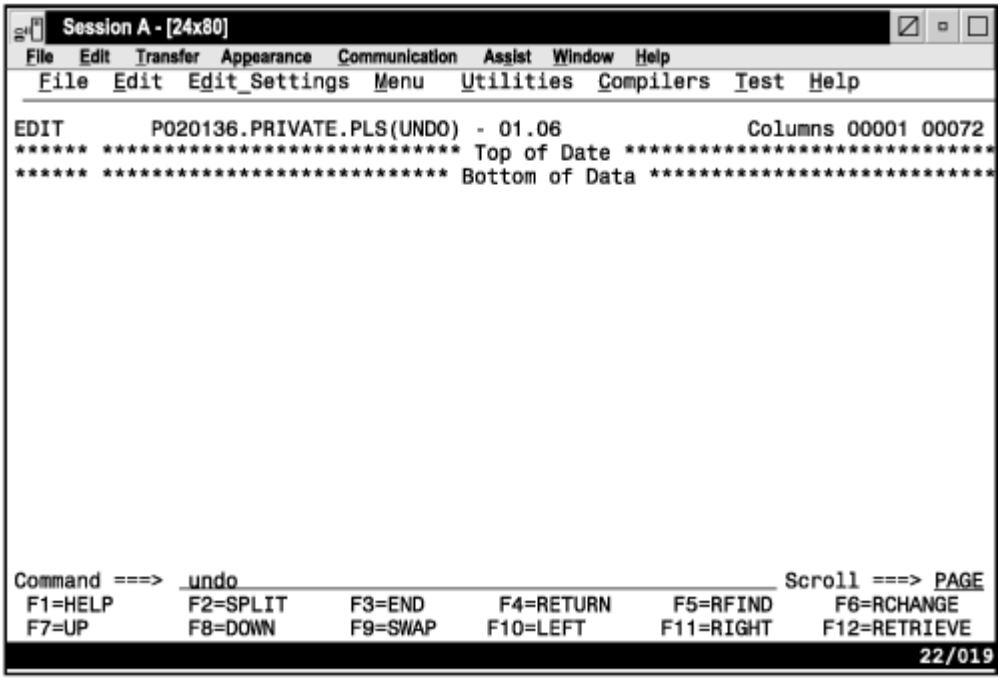


Figure 150. Member after lines are deleted

Figure 151 on page 290 shows the member after UNDO has been entered and the deleted lines have been restored.

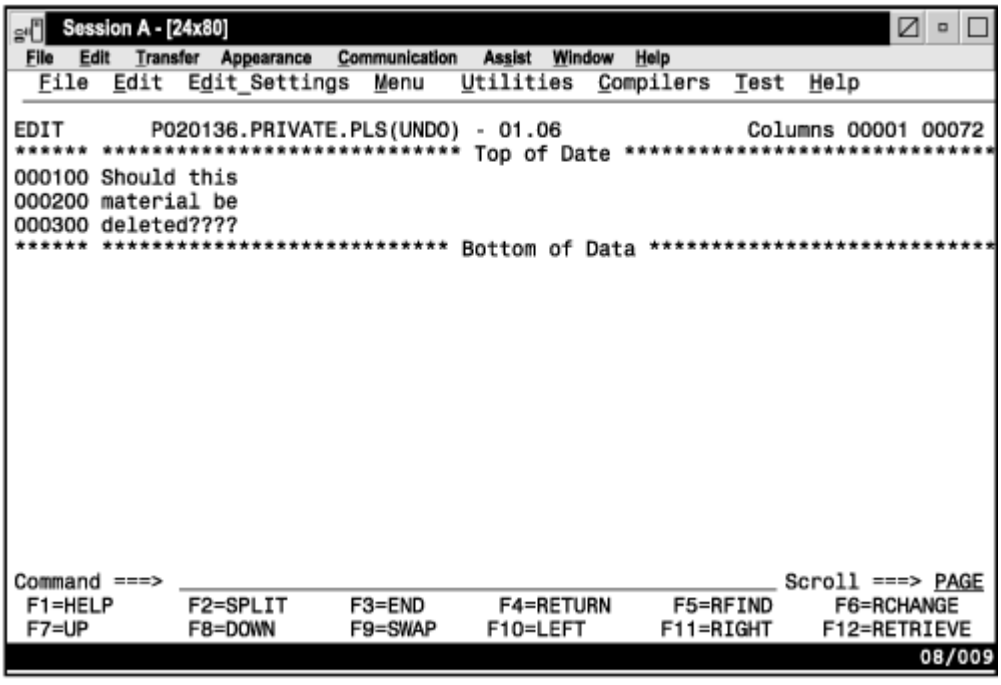
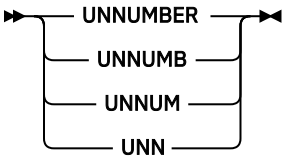


Figure 151. Member after lines have been restored

UNNUMBER—Remove Sequence Numbers

The UNNUMBER primary command sets all sequence fields to blanks, turns off number mode, and positions the data so that column 1 is the first column displayed.

Syntax



Description

UNNUMBER is valid only when number mode is also on. The standard sequence field, the COBOL sequence field, or both, are blanked out. If you alter or delete sequence numbers and enter UNNUMBER on the command line at the same time, the editor issues the message `Some input data ignored` and discards the data you typed over the sequence numbers.

To set all sequence fields to blanks, turn number mode off, and position the panel so that column 1 is the first column to appear:

```
UNNUMBER
```

Examples

You are editing the member in Figure 152 on page 291 and you want to turn off the sequence numbers. Enter UNNUMBER on the command line.

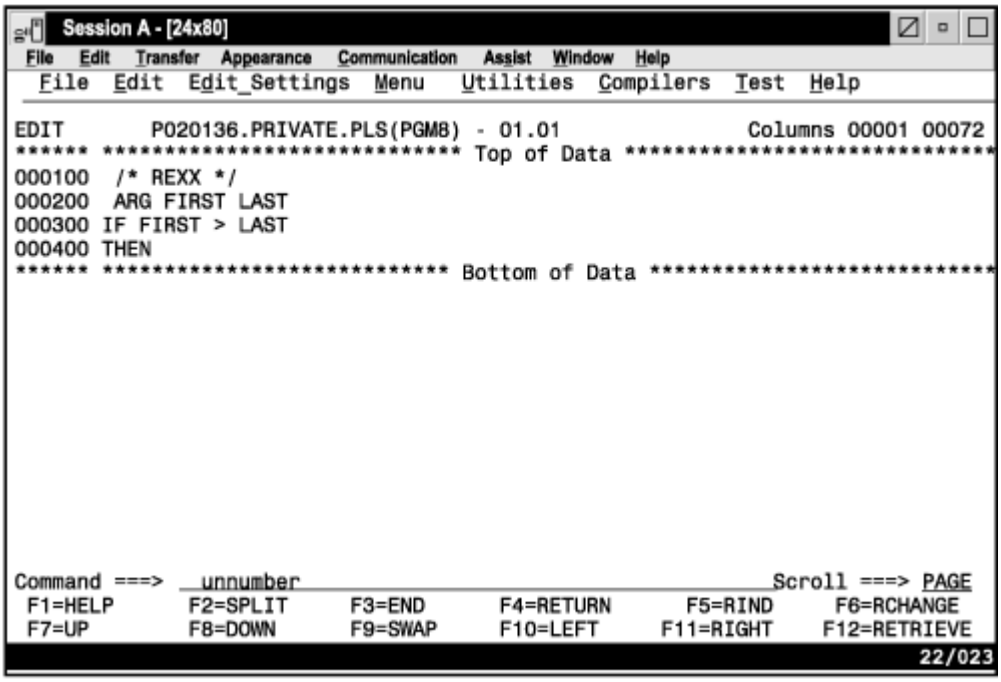


Figure 152. Member before lines are unnumbered

Figure 153 on page 292 shows the member after the sequence numbers have been turned off.

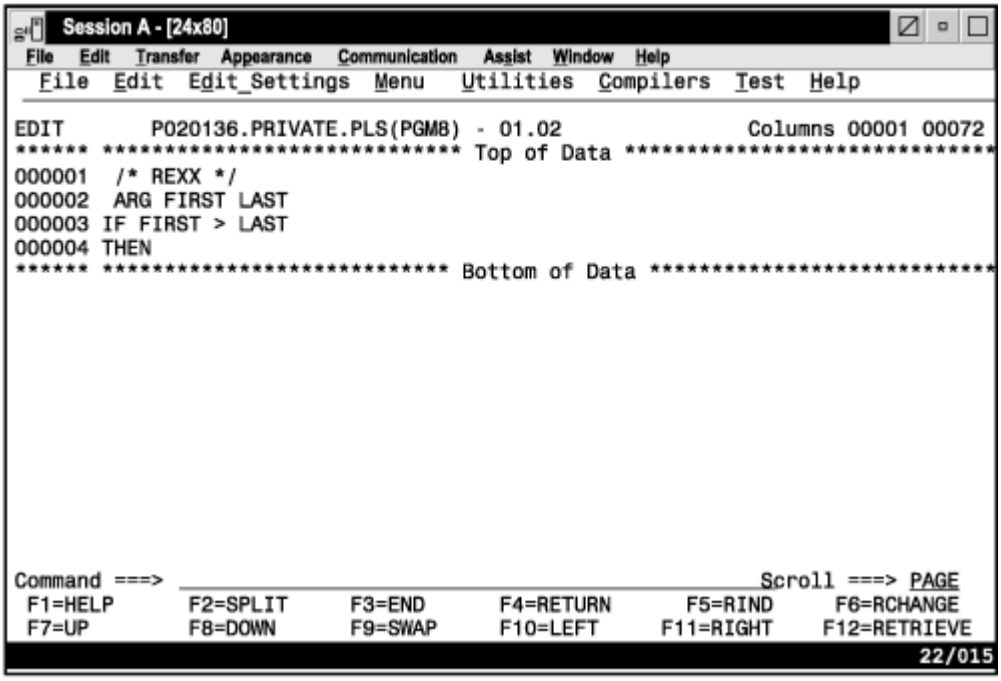
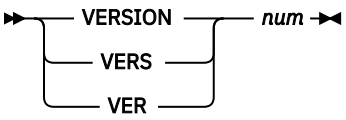


Figure 153. Member after lines are unnumbered

VERSION—Control the Version Number

The VERSION primary command allows you to change the version number assigned to a member of an ISPF library.

Syntax



num
The version number. It can be any number from 1 to 99.

Description

To change the version number of the member that you are editing:

1. On the command line, type:

   ```
   VERSION num
   ```

   where *num* is the new version number.
2. Press Enter.

See “Version and modification level numbers” on page 26, for more information about version numbers.

Examples

Version and modification level numbers are shown on the first line of an edit data display in the format VV.MM, where VV is the version number and MM is the modification level number.

You are editing the member shown in Figure 154 on page 293 and you want to change the version number from 01 to 02. Enter VERSION on the command line.

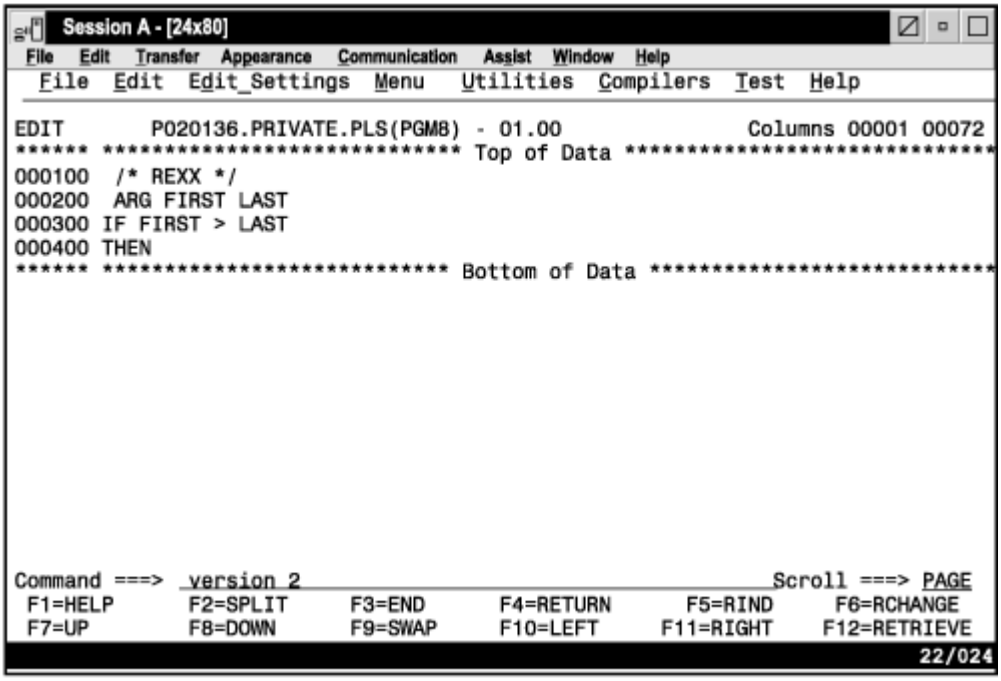


Figure 154. Member before version number is changed

Figure 155 on page 293 shows the member with the changed version number.

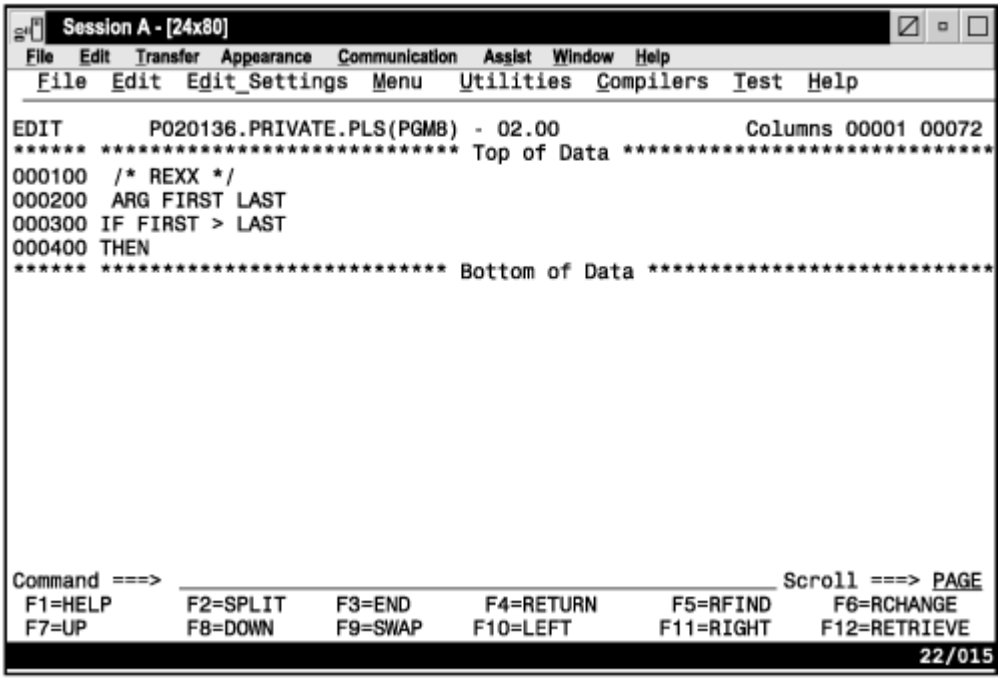


Figure 155. Member after version number is changed

VIEW—View from within an Edit Session

The VIEW primary command allows you to view a sequential data set, partitioned data set member, or z/OS UNIX file during your current edit session.

Syntax

member
A member of the ISPF library or other partitioned data set you are currently editing. You may enter a member pattern to generate a member list.

Description

To view a data set, member, or z/OS UNIX file during your current edit session:

1. On the command line, type:

   ```
   VIEW
   ```

   or

   ```
   VIEW member
   ```

   Here, *member* represents the name of the partitioned data set you are editing. The *member* operand is optional.
2. Press Enter. If you specified a member name, the current library concatenation sequence finds the member. The member is displayed for viewing. If you do not specify a member name, the View Command Entry panel, which is similar to the regular View Entry panel, appears. You can enter the name of any sequential data set, partitioned data set, or z/OS UNIX file to which you have access. When you press Enter, the data set, member, or z/OS UNIX file is displayed for viewing. The editor suspends your initial edit session until the view session is complete. Viewing sessions can be nested until you run out of storage.
3. To exit from the view session, enter the END command. The current edit session resumes.

Examples

To view member YYY of the current library concatenation:

1. On the command line, type:

   ```
   VIEW YYY
   ```
2. Press enter.

Chapter 11. Edit macro commands and assignment statements

This chapter documents intended Programming Interfaces that allow the customer to write programs to obtain the services of ISPF. It also describes the edit macro commands and assignment statements available for the PDF component. Edit macro commands and assignment statements must be included in edit macros that you create.

Macro commands and assignment statements cannot be entered individually from the edit command line. However, once you have created an edit macro, you can use the macro just like any other Edit primary command. You can run an edit macro by:

- Typing the macro name on the command line and pressing Enter
- Pressing a function key to which the macro has been assigned, if any

Note: Edit macro commands should not be confused with TSO commands. Although both are programs, edit macros must not be prefixed with the word 'TSO' when they are invoked.

All edit macros must have an ISREDIT MACRO statement as the first edit command. For more information see “Syntax” on page 366.

Each command description in this documentation consists of:

Syntax
A syntax diagram for coding the macro command, including a description of any required or optional operands.

Description
An explanation of the function and operation of the command. This description also refers to other commands that can be used with this command.

Return Codes
A description of codes returned by the macro command. For all commands, a return code of 20 or higher implies a severe error. See “Return codes from user-written edit macros” on page 109 and “Return codes from PDF edit macro commands” on page 110 for more information.

Examples
Sample usage of the macro command.

Edit macro command summary

This table summarizes the edit macro commands. See the complete description of the commands on the referenced page.

Table 18. Summary of the macro commands

Topic	Description
“AUTOLIST—Set or Query Autolist Mode” on page 300	Sets the current autolist mode or retrieves the value and places it in a variable.
“AUTONUM—Set or Query Autonum Mode” on page 301	Sets the current autonum mode or retrieves the value and places it in a variable.
“AUTOSAVE—Set or Query Autosave Mode” on page 303	Sets the current autosave mode or retrieves the value and places it in a variable.

Table 18. Summary of the macro commands (continued)	
Topic	**Description**
"BLKSIZE—Query the Block Size" on page 304	Returns the block size of the data set being edited in a specified variable.
"BOUNDS—Set or Query the Edit Boundaries" on page 305	Sets the left and right boundaries or retrieves the values and places them in variables.
"BROWSE—Browse from within an Edit Session" on page 307	Browses another member in the data set.
"BUILTIN—Process a Built-In Command" on page 308	Processes a built-in command even if a macro or macro statement with the same name has been defined.
"CANCEL—Cancel Edit Changes" on page 308	Ends the edit session without saving any changes.
"CAPS—Set or Query Caps Mode" on page 309	Sets caps mode.
"CHANGE—Change a Search String" on page 310	Changes a data string to another string.
"CHANGE_COUNTS—Query Change Counts" on page 313	Retrieves the values set by the most recently processed CHANGE command and places these values in variables.
"COMPARE—Edit Compare" on page 314	Compares a library member or data set with the data being edited.
"COPY—Copy Data" on page 317	Copies a member of the library into the member being edited.
"CREATE—Create a Data Set or a Data Set Member" on page 318	Creates a new member from the data that is being edited.
"CURSOR—Set or Query the Cursor Position" on page 320	Sets the relative line and column number of the cursor or retrieves the values and places them in variables.
"CUT—Cut and Save Lines" on page 322	Cut and save lines.
"DATA_CHANGED—Query the Data Changed Status" on page 323	Retrieves the data changed status and places it in a variable.
"DATA_WIDTH—Query Data Width" on page 324	Retrieves the logical data width and places it in a variable.
"DATAID—Query Data ID" on page 325	Retrieves the data ID for the data set being edited and places it in a variable.
"DATASET—Query the Current and Original Data Set Names" on page 326	Retrieves the name of a data set and places it in a variable.
"DEFINE—Define a Name" on page 327	• Assigns an alias to a macro or built-in command. • Disables the use of a macro or built-in command. • Identifies a macro that replaces a built-in command of the same name. • Identifies programs that are edit macros.
"DELETE—Delete Lines" on page 329	Deletes lines from the data.
"DISPLAY_COLS—Query Display Columns" on page 330	Retrieves the column numbers for the first and last data columns on the panel and places them in variables.

Table 18. Summary of the macro commands (continued)	
Topic	**Description**
“DISPLAY_LINES—Query Display Lines” on page 331	Retrieves the relative line numbers of the first and last data lines that would appear if the macro ended and places them in variables.
“DOWN—Scroll Down” on page 332	Scrolls data down from the current panel position.
“EDIT—Edit from within an Edit Session” on page 333	Edits another member in the data set (recursive editing).
“END—End the Edit Session” on page 334	Ends the edit session.
“EXCLUDE—Exclude Lines from the Display” on page 335	Marks lines in the data that should not appear.
“EXCLUDE_COUNTS—Query Exclude Counts” on page 337	Retrieves the values set by the most recently processed EXCLUDE command and places them in variables.
“FIND—Find a Search String” on page 338	Locates a search string. It is recommended that you do not use FIND in a macro because any excluded data string found is shown on the panel. Use SEEK to perform the identical function without changing the line's exclude status.
“FIND_COUNTS—Query Find Counts” on page 340	Retrieves values set by the most recently processed FIND or RFIND command and places them in variables.
“FLIP—Reverse Exclude Status of Lines” on page 341	Reverses the exclude status of a specified group of lines in a file or of all the lines in a file.
“FLOW_COUNTS—Query Flow Counts” on page 342	Retrieves values set by the most recently processed TFLOW command and places them in variables.
“HEX—Set or Query Hexadecimal Mode” on page 342	Sets the hexadecimal mode or retrieves the value and places it in a variable.
“HIDE—Hide Excluded Lines Message” on page 344	Removes the "*n* Line(s) not Displayed" messages from the display where lines have been excluded by the EXCLUDE command.
“HILITE—Enhanced Edit Coloring” on page 345	Highlights in user-specified colors many language-specific constructs, program logic features, the phrase containing the cursor, and any strings that match the previous FIND operation or those that would be found by an RFIND or RCHANGE request. Can also be used to set default colors for the data area in non-program files and for any characters typed since the previous Enter or function key entry.
“IMACRO—Set or Query an Initial Macro” on page 349	Sets or retrieves the value for the initial macro in the profile and places it in a variable.
“INSERT—Prepare Display for Data Insertion” on page 350	Displays one or more lines for data entry.
“LABEL—Set or Query a Line Label” on page 352	Sets or retrieves the values for the label on the specified line and places them in variables.
“LEFT—Scroll Left” on page 353	Scrolls data left from the current panel position.
“LEVEL—Set or Query the Modification Level Number” on page 354	Sets the modification level number or retrieves the value and places it in a variable.

Table 18. Summary of the macro commands (continued)

Topic	Description
"LINE—Set or Query a Line from the Data Set" on page 355	Sets or retrieves the data from the data line and places it in a variable.
"LINE_AFTER—Add a Line to the Current Data Set" on page 357	Adds a line after the specified line.
"LINE_BEFORE—Add a Line to the Current Data Set" on page 358	Adds a line before the specified line.
"LINE_STATUS—Query Source and Change Information for a Line in a Data Set" on page 360	Retrieves source and change information for a specified data line.
"LINENUM—Query the Line Number of a Labeled Line" on page 362	Retrieves the relative line number of a specified label and places it in a variable.
"LOCATE—Locate a Line" on page 363	Locates a line.
"LRECL—Query the Logical Record Length" on page 365	Returns the logical record length of the data being edited in a variable.
"MACRO—Identify an Edit Macro" on page 366	Identifies a command as a macro. MACRO is required for all macros and must be the first command in a CLIST or REXX exec macro that is not a CLIST or REXX exec statement or the first edit command in a program macro.
"MACRO_LEVEL—Query the Macro Nesting Level" on page 367	Retrieves the nesting level of the macro being run and places it in a variable.
"MACRO_MSG—Set or Query the Macro Message switch" on page 368	Sets or retrieves the value of the macro_msg switch, which controls whether macro processing delivers ISPF messages to the macro.
"MASKLINE—Set or Query the Mask Line" on page 368	Sets or retrieves the value of the mask line, which controls the display formatting of input.
"MEMBER—Query the Current Member Name" on page 370	Retrieves the name of the ISPF library member currently being edited and places it in a variable.
"MEND—End a Macro in the Batch Environment" on page 370	Ends a macro that is running in the batch environment. MEND is obsolete.
"MODEL—Copy a Model into the Current Data Set" on page 370	Copies a specified dialog development model before or after a specified line.
"MOVE— Move a Data Set or a Data Set Member" on page 372	Moves a member of a data set and places it after or before the line specified.
"NONUMBER—Turn Off Number Mode" on page 373	Turns off number mode.
"NOTES—Set or Query Note Mode" on page 374	Sets the current note mode or retrieves the value and places it in a variable.
"NULLS—Set or Query Nulls Mode" on page 375	Sets the current nulls mode or retrieves the value and places it in a variable.
"NUMBER—Set or Query Number Mode" on page 376	Sets the current number mode or retrieves the value and places it in a variable.
"PACK—Set or Query Pack Mode" on page 379	Sets the current pack mode or retrieves the value and places it in a variable.

Table 18. Summary of the macro commands (continued)	
Topic	**Description**
“PASTE—Move or Copy Lines from Clipboard” on page 380	Move or copy lines from a clipboard.
“PRESERVE—Enable Saving of Trailing Blanks” on page 381	Sets the current pack mode or retrieves the value and places it in a variable.
“PROCESS—Process Line Commands” on page 383	Controls when the line commands or data changes typed at the keyboard are to be processed.
“PROFILE—Set or Query the Current Profile” on page 384	Allows you to view or change the default modes for your edit session.
“RANGE_CMD—Query a Command That You Entered” on page 386	Identifies the name of a line command typed at the keyboard and processed by a macro.
“RCHANGE—Repeat a Change” on page 387	Repeats the most recently processed CHANGE command.
“RECFM—Query the Record Format” on page 387	Retrieves the record format of the data set being edited and places the value in variables.
“RECOVERY—Set or Query Recovery Mode” on page 389	Sets the recovery mode or retrieves the value and places it in a variable.
“RENUM—Renumber Data Set Lines” on page 390	Sets number mode on and renumbers all data lines.
“REPLACE—Replace a Data Set or Data Set Member” on page 391	Replaces the specified member in the library with the data specified in the member being edited.
“RESET—Reset the Data Display” on page 393	Restores the status of lines or deletes special temporary lines.
“RFIND—Repeat Find” on page 395	Locates the data string defined by the most recently processed SEEK, FIND, or CHANGE command, or excludes a line that contains the data string from the previous EXCLUDE command.
“RIGHT—Scroll Right” on page 395	Scrolls data to the right of the current panel position.
“RMACRO—Set or Query the Recovery Macro” on page 396	Sets or retrieves the name of the macro set in this edit session.
“SAVE—Save the Current Data” on page 397	Saves the data.
“SAVE_LENGTH—Set or Query Length for Variable-Length Data” on page 399	Sets or queries the length to be used to save each record in a variable-length file.
“SCAN—Set Command Scan Mode” on page 400	Sets the current value of scan mode (for variable substitution) or retrieves the value and places it in a variable.
“SEEK—Seek a Data String, Positioning the Cursor” on page 401	Finds one or more occurrences of a data string. SEEK is similar to FIND; however, when a string is found, the exclude status of the line is not affected.
“SEEK_COUNTS—Query Seek Counts” on page 403	Retrieves the values set by the most recently processed SEEK command and places them in variables.
“SESSION—Query Session Type” on page 404	Identifies the type of session in which the macro is running

Table 18. Summary of the macro commands (continued)	
Topic	**Description**
"SHIFT (—Shift Columns Left" on page 406	Moves columns of data to the left.
"SHIFT)—Shift Columns Right" on page 407	Moves columns of data to the right.
"SHIFT <—Shift Data Left" on page 408	Moves data to the left.
"SHIFT >—Shift Data Right" on page 409	Moves data to the right.
"SORT—Sort Data" on page 409	Puts data in a specified order.
"STATS—Set or Query Stats Mode" on page 412	Sets the current stats mode or retrieves the value and places it in a variable.
"SUBMIT—Submit Data for Batch Processing" on page 414	Submits data that is to be processed as a batch job.
"TABS—Set or Query Tabs Mode" on page 415	Sets the tabs mode or retrieves the mode and places it in a variable.
"TABSLINE—Set or Query Tabs Line" on page 416	Sets the tabs line or retrieves the tabs line and places it in a variable.
"TENTER—Set Up Panel for Text Entry" on page 418	Prepares the panel for power typing.
"TFLOW—Text Flow a Paragraph" on page 419	Restructures paragraphs.
"TSPLIT—Text Split a Line" on page 420	Divides a line so data can be added.
"UNNUMBER—Remove Sequence Numbers" on page 421	Removes the numbers from the data set and turns number mode off.
"UP—Scroll Up" on page 421	Scrolls data up from the current panel position.
"USER_STATE—Save or Restore User State" on page 423	Saves or restores the state of the edit profile values, FIND and CHANGE values, and panel and cursor values.
"VERSION—Set or Query Version Number" on page 424	Sets the version number or retrieves the value and places it in a variable.
"VIEW—View from within an Edit Session" on page 425	Views another member in the data set.
"VOLUME—Query Volume Information" on page 426	Retrieves the volume serial number (or serial numbers) and the number of volumes on which the data set resides.
"XSTATUS—Set or Query Exclude Status of a Line" on page 426	Sets the exclude status of the specified data line or retrieves the value and places it in a variable.

AUTOLIST—Set or Query Autolist Mode

The AUTOLIST macro command sets autolist mode, which controls the automatic printing of data to the ISPF list data set.

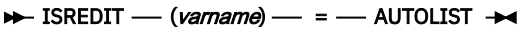
The AUTOLIST assignment statement either sets autolist mode or retrieves the current setting of autolist mode and places it in a variable.

Autolist mode is saved in the edit profile.

Syntax

ON
Specifies that when you end an edit session and save changed data, the editor generates a source listing in the ISPF list data set for eventual printing.

OFF
Does not generate a source listing.



varname
The name of a variable that contains the setting of autolist mode, either ON or OFF.

ON
Same as macro command syntax.

OFF
Same as macro command syntax.

Return codes

0
Normal completion

20
Severe error

Examples

To turn autolist mode on:

```
ISREDIT AUTOLIST ON
```

or

```
ISREDIT AUTOLIST = ON
```

To turn autolist mode off:

```
ISREDIT AUTOLIST OFF
```

or

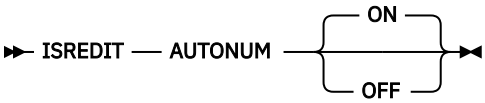
```
ISREDIT AUTOLIST = OFF
```

AUTONUM—Set or Query Autonum Mode

The AUTONUM macro command sets autonum mode, which controls the automatic renumbering of data when it is saved.

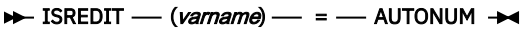
The AUTONUM assignment statement either sets autonum mode or retrieves the current setting of autonum mode and places it in a variable.

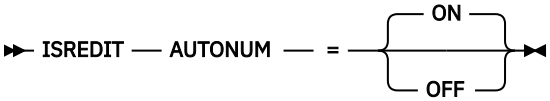
Syntax



ON
Turns on automatic renumbering. When number mode is also on, the data is automatically renumbered when it is saved.

OFF
Turns off automatic renumbering. Data is not renumbered.





varname
The name of a variable containing the setting of autonum mode, either ON or OFF.

ON
Same as macro command syntax.

OFF
Same as macro command syntax.

Description

When number mode is on, the first line of a data set or member is normally line number `000100`, the second number is `000200`, and so on. However, as lines are inserted and deleted, the increments between line numbers can change.

For example, you might think that when a line is inserted between `000100` and `000200`, line `000200` would be given the number `000300` and the new line would become `000200`. Instead, the existing lines retain their numbers and the new line is given line number `000110`.

Therefore, if the original line number increments are important to you, AUTONUM renumbers your lines automatically so that the original increments are maintained.

Autonum mode is saved in the edit profile.

Return codes

0
Normal completion

20
Severe error

Examples

To turn autonum mode on:

```
ISREDIT AUTONUM ON
```

or

```
ISREDIT AUTONUM = ON
```

To turn autonum mode off:

```
ISREDIT AUTONUM OFF
```

or

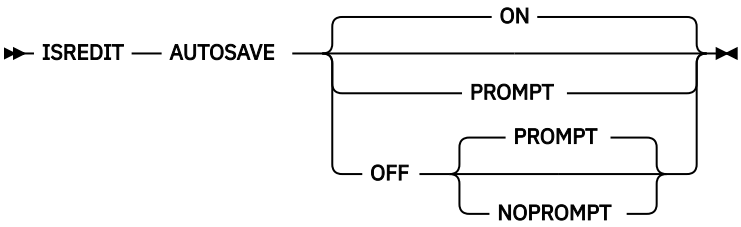
```
ISREDIT AUTONUM = OFF
```

AUTOSAVE—Set or Query Autosave Mode

The AUTOSAVE macro command sets autosave mode, which controls whether changed data is saved when you issue the END command.

The AUTOSAVE assignment statement either sets autosave mode, or retrieves the current setting of autosave mode and places it in variables.

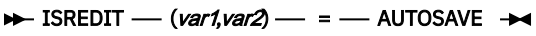
Syntax

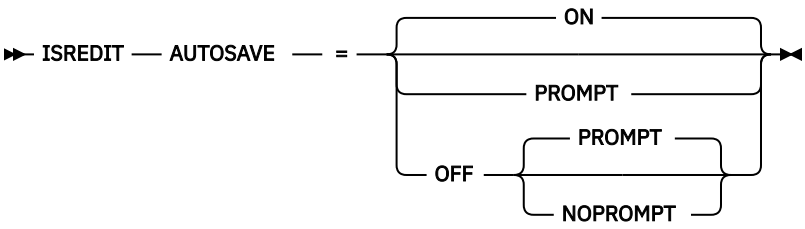


ON
Turns autosave mode on. When you enter END, any changed data is saved.

OFF PROMPT
Turns autosave mode off with the PROMPT operand. You are notified that changes have been made and to use either SAVE (followed by END) or CANCEL. If you specify only the PROMPT keyword, OFF is implied.

OFF NOPROMPT
Turns autosave mode off with the NOPROMPT operand. You are not notified and the data is not saved when you issue an END command. END becomes an equivalent to CANCEL. Use the NOPROMPT operand with caution.





var1
The name of a variable to contain the setting of autosave mode, either ON or OFF.

var2
The name of a variable to contain the prompt value, PROMPT or NOPROMPT.

ON
Same as macro command syntax.

OFF PROMPT
Same as macro command syntax.

OFF NOPROMPT
Same as macro command syntax.

Description

Data is considered changed if you have operated on it in any way that could cause a change. Shifting a blank line or changing a name to the same name does not actually alter the data, but the editor considers this data changed. When you enter SAVE, the editor resets the change status.

Autosave mode, along with the PROMPT operand, is saved in the edit profile.

See the DATA_CHANGED, CANCEL, and END macro commands, and the CANCEL and END primary commands for more information on saving data.

Return codes

0
Normal completion

4
OFF NOPROMPT specified

20
Severe error

Examples

To turn autosave mode on:

```
ISREDIT AUTOSAVE ON
```

or

```
ISREDIT AUTOSAVE = ON
```

To turn autosave mode off and have the editor prompt you to use the SAVE or CANCEL command:

```
ISREDIT AUTOSAVE OFF
```

or

```
ISREDIT AUTOSAVE = OFF
```

To turn autosave mode off and not have the editor prompt you to use SAVE or CANCEL:

```
ISREDIT AUTOSAVE OFF NOPROMPT
```

or

```
ISREDIT AUTOSAVE = OFF NOPROMPT
```

BLKSIZE—Query the Block Size

The BLKSIZE assignment statement returns the block size of the data being edited in a specified variable.

Syntax

varname
The name of a variable to contain the block size of the data being edited. The block size is a 6-digit value that is left-padded with zeros.

Return codes

0
Normal completion

12
Syntax Error

20
Severe error

Note: For a z/OS UNIX file, the BLKSIZE assignment statement returns a value of 0.

Examples

To find the block size and continue processing if the block size is greater than 800:

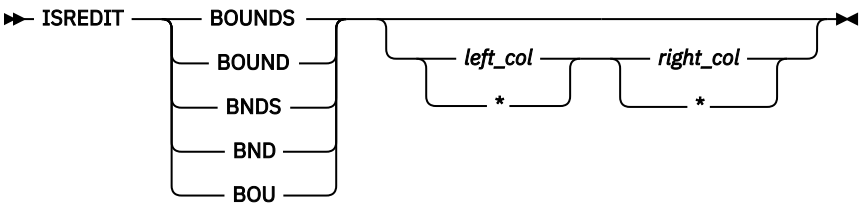
```
ISREDIT (BSIZE) = BLKSIZE
IF &BSIZE > 000800 THEN -
   …
```

BOUNDS—Set or Query the Edit Boundaries

The BOUNDS macro command sets the left and right boundaries and saves them in the edit profile.

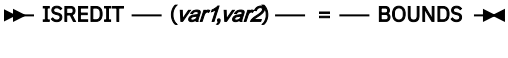
The BOUNDS assignment statement sets or retrieves the left and right boundaries and places the values in variables.

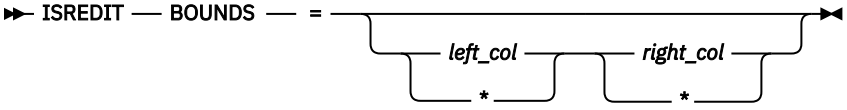
Syntax



left_col
The left boundary column to be set.

right_col
The right boundary column to be set.





var1
A variable containing the left boundary. If the variable is VDEFINEd in character format, it should be defined with a length of 5. The returned value is left padded with zeros. For compatibility with earlier releases of ISPF, a length of 3 or 4 is allowed if no data loss will result.

var2
A variable containing the right boundary. If the variable is VDEFINEd in character format, it should be defined with a length of 5. The returned value is left padded with zeros. For compatibility with earlier releases of ISPF, a length of 3 or 4 is allowed if no data loss will result.

left_col
Same as macro command syntax.

right_col
Same as macro command syntax.

Description

The BOUNDS macro command provides an alternative to setting the boundaries with the BOUNDS line command or primary command; the effect on the member or data set is the same.

The column numbers are always data column numbers (see "Referring to column positions" on page 106). Thus, for a variable format data set with number mode on, data column 1 is column 9 in the record.

See "Edit boundaries" on page 23 for more information, including tables that show commands affected by bounds settings and default bounds settings for various types of data sets.

Return codes

0
Normal completion

4
Right boundary greater than default, default right boundary used

12
Invalid boundaries specified

20
Severe error

Examples

To set the boundaries to their default values, type:

```
ISREDIT BOUNDS
```

To set one boundary while leaving the other value unchanged, type an asterisk (*) for the boundary to be unchanged. For example, to set the left boundary from the variable &LEFT, and leave the right boundary unchanged, type:

```
ISREDIT BOUNDS &LEFT *
```

To set the left boundary to 1, leaving the right boundary unchanged:

```
ISREDIT BOUNDS = 1 *
```

To save the value of the left boundary in the variable &LEFT:

```
ISREDIT (LEFT) = BOUNDS
```

To save the value of the right boundary in the variable &RIGHT:

```
ISREDIT (,RIGHT) = BOUNDS
```

To evaluate numbers for bounds when NUMBER COBOL is on, or NUMBER is on for a variable blocked data set:

```
/* REXX - Set physical bounds in a macro.  Input is 2 column       */
/*        numbers and result is bounds set on that physical column */
/*        regardless of number setting.  Bounds will not be set    */
/*        within line number areas.  This sample has minimal       */
/*        error checking.                                          */
Address isredit
'MACRO (LEFT,RIGHT)'                 /* Take left and right bounds*/
'(NUMBER,COBOL) = NUMBER'            /* Get number status         */
Parse Var cobol . cobol .            /* Get just left status      */
'(RECFM) = RECFM'                    /* Get record format         */
'(DW) = DATA_WIDTH'                  /* Get data width            */
If left='' Then left = 1             /* Assume col 1 for left     */
If right='' Then right = dw          /* Assume datawidth for right*/
shift = 0                            /* Assume no left seq numbers*/
If cobol='COBOL' Then                /* If numbered as cobol      */
  shift = 6                          /*   Account for sequence num*/
Else If number='ON' & recfm='V' Then  /* If numbered variable block*/
  shift = 8                          /*   Account for sequence num*/
right = max(1,right - shift)         /* Adjust right column       */
right = min(right,dw)                /* Adjust right column       */
left  = max(1,left  - shift)         /* Adjust left column        */
left  = min(left ,dw)                /* Adjust left column        */

'BOUNDS 'min(left,right) max(left,right) /* Issue bounds command   */
'PROFILE'
```

BROWSE—Browse from within an Edit Session

The BROWSE macro command allows you to browse a member of the same partitioned data set during your current edit session.

Syntax

member
A member of the library or other partitioned data set you are currently editing. You may enter a member pattern to generate a member list.

Description

Your initial edit session is suspended until the browse session is complete.

To exit from the browse session, END or CANCEL must be processed by a macro or entered by you. The current edit session resumes.

For more information on using the BROWSE service, refer to the *z/OS ISPF Services Guide*.

Return codes

0
Normal completion

12
Your error (invalid member name, recovery pending)

20
Severe error

Examples

To browse the member OLDMEM in your current ISPF library:

```
ISREDIT BROWSE OLDMEM
```

BUILTIN—Process a Built-In Command

The BUILTIN macro command is used within an edit macro to process a built-in edit command, even if a macro or macro statement with the same name has been defined.

Syntax

cmdname
The built-in command to be processed.

Description

If you create a macro named MACEND and enter a DEFINE END ALIAS MACEND command, your MACEND macro runs when you enter END. Within the MACEND macro you can perform logic and use a built-in END command to actually end the edit session.

Note that if END is issued in your MACEND macro without being preceded by BUILTIN, the MACEND macro would run again, resulting in an infinite loop.

Return codes

n
Return code from the built-in command

20
Severe error

Examples

To process the built-in END command:

```
ISREDIT BUILTIN END
```

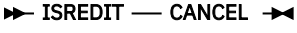
To process the built-in CHANGE command:

```
ISREDIT BUILTIN CHANGE ALL " " "-"
```

CANCEL—Cancel Edit Changes

The CANCEL macro command ends your edit session without saving any of the changes you have made.

Syntax



Description

CANCEL is especially useful if you have changed the wrong data, or if the changes themselves are incorrect. See the DATA_CHANGED, AUTOSAVE, and END commands for more information about saving data.

Note:

1. If you issue SAVE and later issue CANCEL, the changes you made before issuing SAVE are not canceled.
2. When CANCEL is entered in the macro field in the edit prompt panel (ISRUEDIT), the macro name is not saved in the profile for use in future sessions. This is to avoid having the editor appear to do nothing when it is invoked from the data set list.

CANCEL does not cause automatic recording in the ISPF list data set, regardless of the setting of the autolist mode.

Return codes

0
Normal completion

20
Severe error

Examples

To cancel the current edit session:

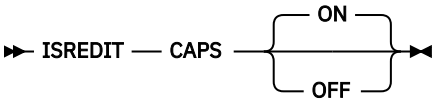
```
ISREDIT CANCEL
```

CAPS—Set or Query Caps Mode

The CAPS macro command sets caps mode, which controls whether alphabetic data that you type at the terminal is automatically converted to uppercase during edit operations.

The CAPS assignment statement either sets caps mode or retrieves the setting of caps mode and places it in a variable.

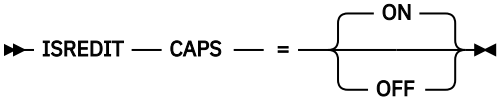
Syntax



ON
Turns caps mode on.

OFF
Turns caps mode off.

Assignment statement syntax



varname
The name of a variable containing the setting of caps mode, either ON or OFF.

ON
Same as macro command syntax.

OFF
Same as macro command syntax.

Description

When the editor retrieves data, it sets the caps mode on if the data contains all uppercase letters, or off if the data contains lowercase letters. The editor displays a message when the caps mode changes.

Caps mode is saved in the edit profile. To override the automatic setting of caps mode, you can include the CAPS command in an initial macro.

Caps mode is normally on for program development work. When caps mode is set to on, any alphabetic data that you type, plus any other alphabetic data that already exists on that line, is converted to uppercase when you press Enter or a function key.

Caps mode is normally off when you edit text documentation. When caps mode is set to off, any alphabetic data that you type remains just as you typed it. If you typed it in uppercase, it stays in uppercase; if you typed it in lowercase, it stays in lowercase. Also, alphabetic data that is already typed on that line is not affected.

CAPS does not apply to DBCS fields in formatted data or to DBCS fields in mixed fields. If you specify CAPS, the DBCS fields remain unchanged. See the LC (lowercase) and UC (uppercase) line commands and the CAPS primary command for more information about changing cases.

Return codes

0
Normal completion

20
Severe error

Examples

To save the value of caps mode in variable &CAPMODE:

```
ISREDIT (CAPMODE) = CAPS
```

To turn caps mode OFF:

```
ISREDIT CAPS = OFF
```

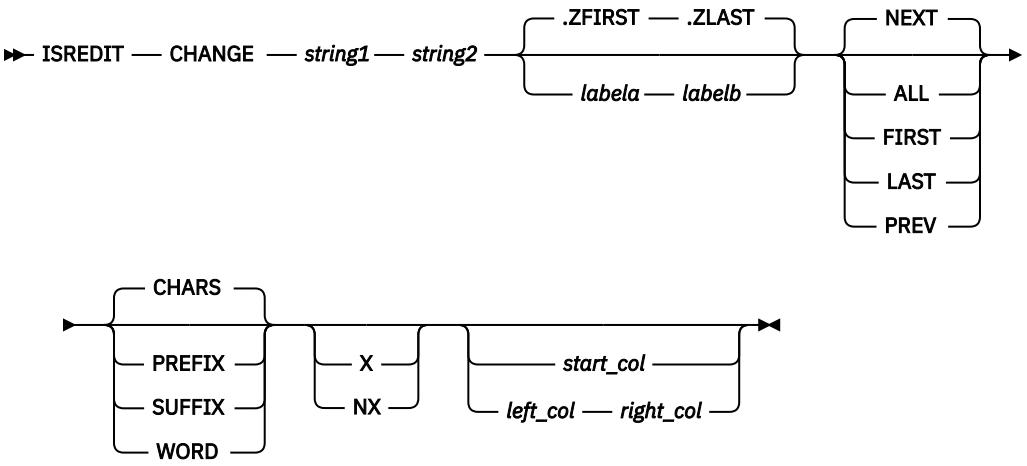
To set the value of caps mode from variable &CAPMODE:

```
ISREDIT CAPS &CAPMODE
```

CHANGE—Change a Search String

The CHANGE macro command changes one search string into another.

Syntax



string1
The search string you want to change. See "Finding, seeking, changing, and excluding data" on page 44.

Note: For edit macros written in CLIST, strings that contain an open comment delimiter (/*) must be placed within quotes within the &STR() such as &STR('/*XXX'). The maximum allowable length of the string is 256 bytes. If you are specifying a hex string, the maximum is 128 hexadecimal characters.

string2
The string you want to replace *string1*. The maximum allowable length of the string is 256 bytes. If you are specifying a hex string, the maximum is 128 hexadecimal characters. See "Finding, seeking, changing, and excluding data" on page 44.

labela*, *labelb
Labels identifying the start and end of the group of lines CHANGE searches.

If the cursor is currently placed above the start label and the PREV occurrence of a string is requested, or the cursor is currently placed below the end label and the NEXT occurrence of a string is requested, the process returns a return code of 4 and the string is not found, even if it exists within the label range.

For more information about using labels to identify a group of lines, see "Labels and line ranges" on page 59.

NEXT
Starts at the first position after the current cursor location and searches ahead to find the next occurrence of *string1*.

ALL
Starts at the top of the data and searches ahead to find all occurrences of *string1*.

FIRST
Starts at the top of the data and searches ahead to find the first occurrence of *string1*.

LAST
Starts at the bottom of the data and searches backward to find the last occurrence of *string1*.

PREV
Starts at the current cursor location and searches backward to find the previous occurrence of *string1*.

CHARS
Locates *string1* anywhere the characters match.

PREFIX
Locates *string1* at the beginning of a word.

SUFFIX
Locates *string1* at the end of a word.

WORD
Locates *string1* when it is delimited on both sides by blanks or other non-alphanumeric characters.

X
Scans only lines that are excluded from the display.

NX
Scans only lines that are not excluded from the display.

start_col
The first column to be included in the range of columns to be searched. When you specify only one column, the editor finds the string only if the string starts in the specified column.

left_col
The first column to be included in the range of columns CHANGE is to search.

right_col
The last column to be included in the range of columns CHANGE is to search.

Note: For more information about restricting the search to only a portion of each line, see "Limiting the search to specified columns" on page 54.

Description

CHANGE is often used with FIND, EXCLUDE, and SEEK, and the CHANGE_COUNTS assignment statement.

To change the next occurrence of "ME" to "YOU" without specifying any other qualifications, include this command in an edit macro:

```
ISREDIT CHANGE ME YOU
```

This command changes only the next occurrence of the letters "ME" to "YOU". Since no other qualifications were specified, the letters "ME" can be:

- Uppercase or a mixture of uppercase and lowercase
- At the beginning of a word (prefix), the end of a word (suffix), or the entire word (word)
- In an excluded line or a non-excluded line
- Anywhere within the current boundaries

To change the next occurrence of "ME" to "YOU", but only if the letters are uppercase, include this command in an edit macro:

```
ISREDIT CHANGE C'ME' YOU
```

This type of change is called a character string change (note the C that precedes the search string) because it changes the next occurrence of the letters "ME" to "YOU" only if the letters are found in uppercase. However, since no other qualifications were specified, the change occurs no matter where the letters are found, as outlined in the preceding list.

When you would like to issue CHANGE, but you are unsure of the exclude status of a line, you can use the XSTATUS assignment statement with SEEK. First, find the particular line with SEEK. Then, determine the exclude status with the XSTATUS assignment statement. Use CHANGE to change the string; and finally, reset the exclude status with another XSTATUS assignment statement. For example:

```
ISREDIT SEEK ABC
DO WHILE &LASTCC=0
  ISREDIT (X) = XSTATUS .ZCSR
  ISREDIT CHANGE ABC DEF .ZCSR .ZCSR
  ISREDIT XSTATUS .ZCSR = &X
  ISREDIT SEEK ABC
END
```

For more information, including other types of search strings, see “Finding, seeking, changing, and excluding data” on page 44.

Return codes

0
Normal completion

4
String not found

8
Change error. *string2* is longer than *string1* and substitution was not performed on at least one change.

12
Inconsistent parameters. The string to be found does not fit between the specified columns.

20
Severe error

Examples

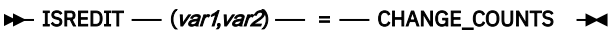
Before changing the current member name, put it into a variable name such as MEMNAME. To add an identifier to that name, if it is in columns 1 to 10 and lies within the first line and the line labeled .XLAB:

```
ISREDIT (MEMNAME) = MEMBER
ISREDIT CHANGE WORD &MEMNAME "MEMBER:&MEMNAME" 1 10 .ZFIRST .XLAB
```

CHANGE_COUNTS—Query Change Counts

The CHANGE_COUNTS assignment statement retrieves values set by the most recently processed CHANGE command and places these values in variables.

Syntax



var1
The name of a variable to contain the number of strings changed. It must be an 8-character value that is left-padded with zeros.

var2
The name of a variable to contain the number of strings that could not be changed. It also must be an 8-character value that is left-padded with zeros.

Return codes

0
Normal completion

20
Severe error

Examples

To put the number of changes resulting from the most recent CHANGE command into the variable &CHGED:

```
ISREDIT (CHGED) = CHANGE_COUNTS
```

To put the number of change errors into variable &ERRS:

```
ISREDIT (,ERRS) = CHANGE_COUNTS
```

To put the number of changes and change errors into variables &CHG and &ERR:

```
ISREDIT (CHG,ERR) = CHANGE_COUNTS
```

COMPARE—Edit Compare

The COMPARE command compares the file you are editing with an external sequential data set, member of a partitioned data set, or z/OS UNIX file. Lines that exist only in the file being edited are marked, and lines that exist only in the file being compared are inserted as information lines in the file being edited. The command operates as a primary command or an edit macro.

If you compare the file you are editing with a member of a PDSE version 2 data set that is configured for member generations, the current generation of the member is used for the comparison.

You can use the Delete and Make Data line commands to merge changes between files that are being compared.

The COMPARE function supports all line lengths, but some SuperC options are ignored for line lengths greater than 256 characters long.

When you are editing a cataloged data set, explicit data set names refer to cataloged data sets. However, if you are editing an uncataloged data set, explicit member names refer to cataloged data sets, but if you specify only a member name, COMPARE searches for the member in the current uncataloged data set. For example, if you are editing an uncataloged data set called "userid.TEMP", the command

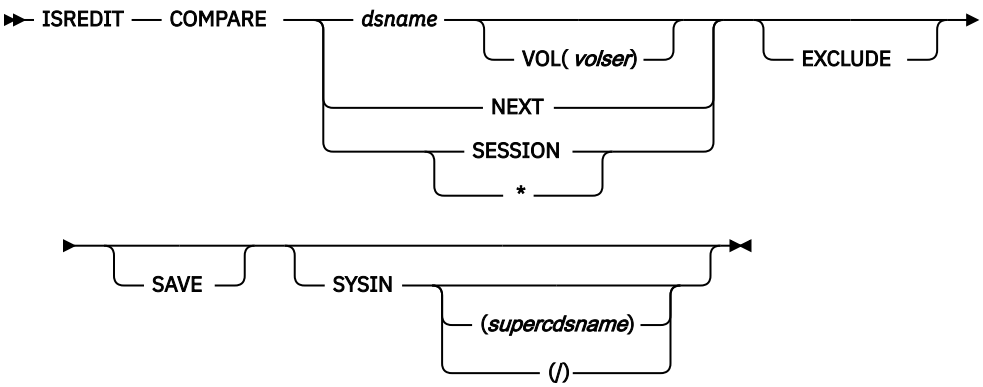
```
COMPARE TEMP
```

first looks for member TEMP in the current, uncataloged data set, then looks for a cataloged data set named TEMP (TSO prefix rules apply). If it finds data set TEMP, and the data set being edited is a PDS member, then the same named member is searched for in data set TEMP.

Use of COMPARE when editing concatenations that contain uncataloged data sets is not supported and can lead to unpredictable results.

If you have made changes to the data before issuing the COMPARE command, the COMPARE command uses the current contents of the edit session during the comparison. Because COMPARE does not require the data to be saved on disk, you can use the COMPARE command from EDIF, VIIF, or EDIREC sessions. However, COMPARE NEXT and COMPARE SESSION are *not* supported in EDIF, VIIF, or EDIREC sessions.

Syntax



dsname
The name of a member, data set, or z/OS UNIX file to which the current file is compared. This variable can be specified as a fully qualified data set name (in quotation marks), a partially qualified data set name, a member name, or a path name. (Also, see "Specifying z/OS UNIX pathnames with edit primary and macro commands" on page 16.)

If you specify only a member name, it can be preceded by a left parenthesis symbol. The right parenthesis is allowed but not required. The current edit session must be of a member of a partitioned data set. The current edit concatenation is searched for the member to compare.

If you specify only a data set name and the current file is a member of a PDS, then the specified data set is searched for a member of the same name as the member being edited.

VOL(*volser*)
Used when comparing against an uncataloged data set. Specifies the volser of the volume containing the uncataloged data set.

NEXT
Specifies to do a comparison between the currently edited member and the next member of the same name found at a higher level of the hierarchy (or next level of the edit concatenation) than the current member. For example, if the current member is found in the third level of the concatenation, and a like-named member exists at the fourth level, then the third and fourth level members are compared. After data is saved in the lowest level, compares are done from that level upward.

SESSION
Specifies that you want to compare the changes you have made during the edit session with the copy of the data saved on disk. Use COMPARE SESSION or COMPARE * to see the changes you have made to the edit data since the beginning of the edit session or since the last SAVE command.

*****
Same as SESSION.

EXCLUDE
Specifies that all matching lines in the compared data sets are excluded from the display *except* for a specified number of lines above and below the differences. The differences themselves are also shown in the display. The specified number of lines that are shown is set on the Edit Compare Settings and/or Command Parameters panel. If you do not specify a new number for this edit session, then whatever was the last number set is still valid. To change this number, issue the COMPARE command with no operand and change the EXCLUDE field on the Edit Compare Settings and/or Command Parameters panel. Valid numbers are 0 through 12, inclusive. You cannot display the Edit Compare Settings and/or Command Parameters panel from a macro.

You can also use the **COMPARE EXCLUDE** command at any time to exclude all lines in a file except lines with line labels and information lines, and the lines above and below those lines. When you specify EXCLUDE without a data set name or NEXT, no comparison is done. Instead the labels and information lines that already exist in the file are used to exclude functions. See "Examples" on page 316 for a macro that uses this technique.

SAVE
Specifies that SuperC (which performs the actual compare function) create a listing. The listing is saved in a data set with one of these names:

- *tsopref*.ISPFEDIT.COMPARE.LIST (where *tsopref* is your TSO prefix).
- *tsopref*.*userid*.ISPFEDIT.COMPARE.LIST (where *userid* is your TSO user ID and it does not match your TSO prefix).
- *userid*.ISPFEDIT.COMPARE.LIST (where no TSO prefix is defined in your TSO user profile).

Note: If the ISPF configuration table keyword USE_ADDITIONAL_QUAL_FOR_PDF_DATA_SETS is set to YES, an additional qualifier defined with the ISPF_TEMPORARY_DATA_SET_QUALIFIER keyword is included before the ISPFEDIT qualifier.

The save function is intended for debugging purposes, but it also provides a way to create a SuperC listing. The listing produced is a Change listing (option CHNGL). No notification is given regarding successful creation of the listing, and errors allocating the listing do not cause the comparison to end.

Note: Because of the way the SuperC comparison is done, the file currently being edited is shown in the SuperC listing as the *old* file, and the file to which the current file is being compared is listed as the *new* file. Therefore, insertions refer to lines that are *not* in the current file, and deletions refer to lines that are only in the current file.

SYSIN
Specifies not to free the ddname SYSIN before calling SuperC to compare files. This enables you to pass SuperC Process Statements to alter the comparison. No validation is done on the type of SYSIN allocation or the contents of the data set.

supercdsname
The name of a data set containing SuperC process statements.

/
Displays the Edit Compare SYSIN specification panel where you can specify the name of a data set containing SuperC Process statements that are used for the compare. The SYSIN data set is freed at the end of the compare.

Return codes

0
Normal completion

8
Member, data set, or z/OS UNIX file not found, or an error opening the member or data set occurred.

12
No parameters specified, or another parameter error such as not valid NEXT or member specification.

20
Severe error. SuperC, allocation, or delta file error occurred.

Examples

To compare the current file to another file called X.Y.Z and to save the SuperC output file in ISPFEDIT.COMPARE.LIST:

```
ISREDIT COMPARE X.Y.Z SAVE
```

To compare the current file to a member in the same partitioned data set, and exclude everything but the context in which changes exist:

```
ISREDIT COMPARE (memname) EXCLUDE
```

To find all of the occurrences of a string in a file and exclude lines to show the context in which the strings were found, you can use this macro:

```
/* REXX - Edit macro to find a string, show only lines with the   */
/*        string and a few lines above and below found strings.   */
/*        This uses the COMPARE EXCLUDE command to perform the    */
/*        line exclude function.                                  */
/* ---------------------------------------------------------------- */
Address isredit                 /*                                */
'MACRO (PARM)'                  /* Accept input string            */
If parm ^= '' Then              /* Do nothing if no parameters    */
  Do                            /*                                */
    'RESET LABEL'               /* Remove all existing labels     */
    'F FIRST 'parm              /* Find first string occurrence   */
    Do While(rc=0)              /*   For each occurrence          */
      'LABEL .ZCSR = 'label()' 0'/*      Assign a label to line    */
      'RFIND'                   /*      Find next occurrence      */
    End                         /*                                */
    'COMPARE X'                 /* Exclude everything except      */
                                /*  Labels and above/below lines  */
    'RESET LABEL'               /* Remove all labels              */
    '(XSTAT) = XSTATUS .ZFIRST' /* Save exclude status of line 1  */
```

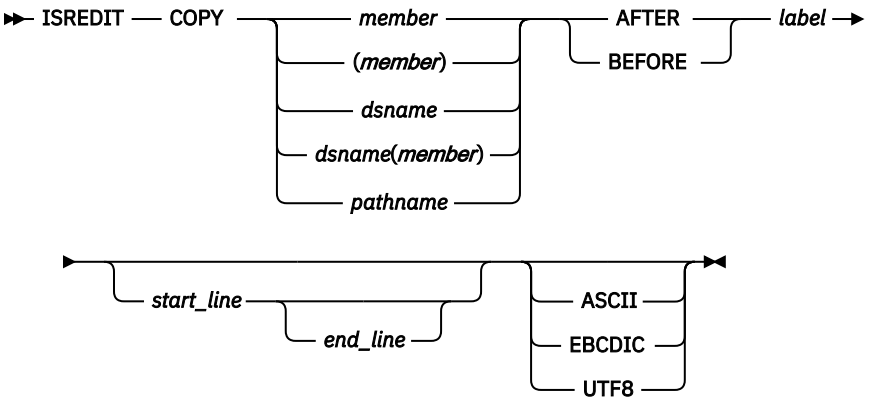
```
    'LOCATE .ZFIRST'             /* Move display to line 1        */
    'XSTATUS .ZFIRST = 'xstat    /* Restore line 1 exclude status */
  End                            /*                               */
Exit 0                           /* Always return a zero          */
/* ------------------------------------------------------------- */
label:Procedure Expose labelnum  /* Routine to generate a unique  */
If datatype(labelnum,'N')=0 Then /*   Edit line label             */
  labelnum=0                     /*                               */
Else                             /*                               */
  labelnum=labelnum+1            /*                               */
Return '.'translate(right(labelnum,4,'0'),'ABCDEFGHIJ','0123456789')
```

COPY—Copy Data

The COPY macro command copies a sequential data set, a member of a partitioned data set, or a z/OS UNIX file into the data are editing.

Syntax

Macro command syntax



member
A member of the ISPF library or partitioned data set that you are editing.

dsname
A partially or fully qualified data set name. If the data set is partitioned, you must include a member name in parentheses. If a name of eight or fewer characters is specified and it could be a member name or a data set name, COPY searches for a member name first. If no member is found, the name is used as a data set.

pathname
The path name for a z/OS UNIX regular file to be copied. (Also, see “Specifying z/OS UNIX pathnames with edit primary and macro commands” on page 16.)

AFTER
The data is copied after the line with the specified label.

BEFORE
The data is copied before the line with the specified label.

label
Label identifying the line where the data is to be copied. It can be either a label that you define or one of the editor-defined labels, such as .ZF or .ZL.

start_line
The number of the first line of the member to be copied. Must be greater than or equal to 1, and less than or equal to the number of lines in the member.

end_line
The number of the last line of the member to be copied. Must be greater than or equal to *start_line* and less than or equal to the number of lines in the member. If not specified, the last line of the member is used.

ASCII, EBCDIC, UTF8
When one of these keywords is supplied, if the data is using a different character set to that designated by the keyword, the data being copied in from the external file is converted from the character set designated by the keyword to the character set specified for the file being edited or to the terminal character set.

Note: If the member name or data set name is less than 8 characters and the data set you are editing is partitioned a like-named member is copied. If a like-named member does not exist the name is considered to be a partially qualified data set name.

Return codes

0
Normal completion

8
End of data reached before last record read

12
Invalid *label* or *linenum*; member not found or BLDL error

16
End of data reached before first record of specified range was reached

20
Syntax error (invalid name, incomplete range), or I/O error.

Examples

To copy all of the member MEM1 at the end of the data:

```
ISREDIT COPY MEM1 AFTER .ZLAST
```

To copy all of data set MOVECOPY.DATA before the first line of data:

```
ISREDIT COPY MOVECOPY.DATA BEFORE .ZFIRST
```

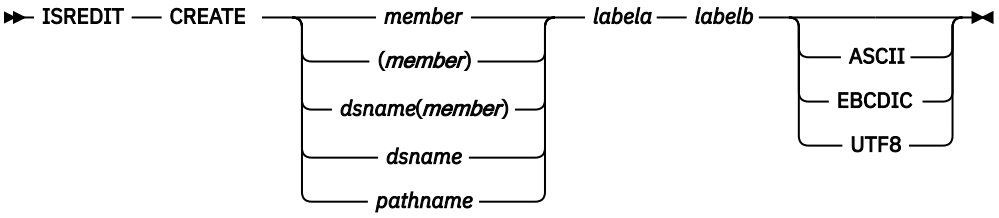
To copy the first three lines of the member MEM1 before the first line of data:

```
ISREDIT COPY MEM1 BEFORE .ZF 1 3
```

CREATE—Create a Data Set or a Data Set Member

The CREATE macro command creates a member of a partitioned data set or a z/OS UNIX file from the data you are editing. This command cannot be used to create a sequential data set. Use the Data Set Utility (option 3.2) to allocate a sequential data set.

Syntax



member
The name of the new member added to the partitioned data set currently being edited. If you are using a concatenated sequence of libraries, the member is always written to the first library in the sequence.

***dataset*(*member*)**
The name of a different partitioned data set and new member to be added to the partitioned data set. The data set name can be fully or partially qualified.

pathname
The path name for a z/OS UNIX regular file to be created. (Also, see specunix.dita#specunix.)

labela*, *labelb
Labels identifying the start and end of the group of lines used to create the new member.

linenum1
Relative line number identifying the start of a group of lines used to create the new member.

linenum2
Relative line number identifying the end of a group of lines used to create the new member. [XXX: Not sure why linenum1 and linenum2 are in this list - they don't appear in the syntax diagram.]

ASCII, EBCDIC, UTF8
When one of these keywords is supplied, if the data is using a different character set to that designated by the keyword, the data being saved in the external file is converted to the character set designated by the keyword.

Description

CREATE adds a member to a partitioned data set only if a member with the same name does not already exist. Use REPLACE if the member already exists.

Return codes

0
Normal completion

8
Member already exists, member not created

12
Invalid label or relative line number. The referenced line does not exist in the file.

20
Syntax error (invalid name or incomplete label or relative line number range), or I/O error.

Examples

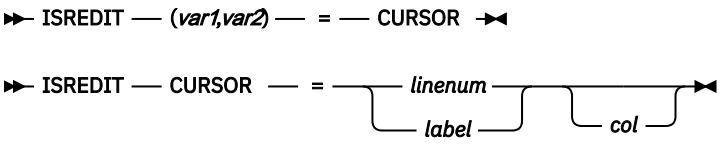
To create a new 10-line member from the first 10 lines of the member being edited:

```
ISREDIT CREATE MEM1 1 10
```

CURSOR—Set or Query the Cursor Position

The CURSOR assignment statement either sets the column number and relative line number of the cursor location within the data, or retrieves the column number and relative line number of the cursor location within the data and places them in variables.

Syntax



var1
The name of a variable containing the line number. The line number is a 6-digit value that is left-padded with zeros. It is the ordinal number (not the sequence number) of the line. If the variable is VDEFINEd in character format, it should be defined with a length of 8. The returned value is left-padded with zeros. For compatibility with previous releases of ISPF, a length of 6 or 7 is allowed in cases where no data loss will occur.

var2
The name of a variable containing the data column number. The data column number is a 3-digit number that is left-padded with zeros. If the variable is VDEFINEd in character format, it should be defined with a length of 5. The returned value is left padded with zeros. For compatibility with previous releases of ISPF, a length of 3 or 4 is allowed in cases where no data loss will occur.

The columns are numbered starting with 1 at the first data column. If the cursor is in the command line, the cursor value is column 0 of the first data line on the panel; the value is column 0 if the cursor is in the line command field. When you retrieve the cursor position in an empty member, the line number and column number are both set to 0.

linenum
The relative line number of the line on which the cursor is to be located. Make sure when you set the cursor to a line number that the line number exists.

label
The label of the line on which the cursor is to be located.

Note: If you try to use a label that has not been assigned, you receive a return code of 20. To avoid this, use the LINENUM assignment statement.

```
ISREDIT (X) = LINENUM .LABEL
```

When using the LINENUM statement, a return code of 8 is issued if the label does not exist.

col
The data column number where the cursor is to be located.

If the column number is beyond the end of the data area when setting the cursor, the cursor is positioned to the next line, which is equivalent to the first position of the line command field.

Description

The position of the cursor shows the starting or ending location for the SEEK, FIND, CHANGE, and EXCLUDE commands. It is also used as the text split point for TSPLIT. See “Referring to column positions” on page 106 for more information on how the column number is determined.

When you run a macro, the cursor value is the cursor position on the panel at run time.

Note: To position the cursor on the command line, issue a return code of 1 from the macro. For example, in CLIST code EXIT CODE(1) as the last statement in your EDIT MACRO to position the cursor on the command line.

These statements can change the cursor position:

```
CHANGE     CURSOR   EXCLUDE
FIND       SEEK     TSPLIT
USER_STATE
```

Table 19 on page 321 shows the line and column numbers returned, depending on the location of the cursor.

Table 19. Cursor position

If the CURSOR location is:	The LINE number is:	The COLUMN number is:
Command line	First display line	0
Line number field	Line by the cursor	0
Left sequence number (the sequence number is on the left of the data when number mode is on)	Line by the cursor	0
Right sequence number	Line by the cursor	Column by the cursor
Left or right of the bounds	Line by the cursor	Column by the cursor
Data within the bounds	Line by the cursor	Column by the cursor
Insert blank space	Line above the cursor. If the cursor is at the top of the panel, then the line number returned is the line below the cursor and the column number is column 0.	Column by the cursor
Non-data line and its line command field (above the last data line)	Line below the non-data line.	0
Non-data line (below the last data line)	Line number of the last line of data	Width of the last line of data plus 1

Return codes

0
Normal completion

4
Column number beyond data, line number incremented

12
Invalid line number

20
Severe error

Examples

To put the line number of the current cursor position into variable &LINE:

```
ISREDIT (LINE) = CURSOR
```

To set the cursor position to data line 1, column 1:

```
ISREDIT CURSOR = 1 1
```

To set the cursor position to column 1 of the last data line:

```
ISREDIT CURSOR = .ZLAST 1
```

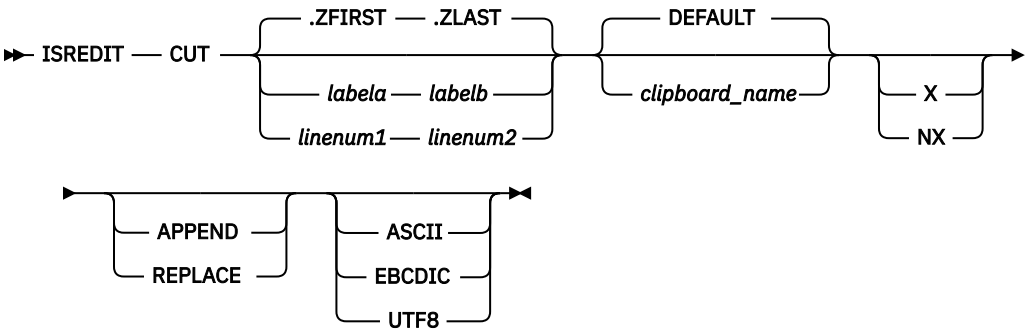
To set the cursor position to the line with the label .LAB, without changing the column position:

```
ISREDIT CURSOR = .LAB
```

CUT—Cut and Save Lines

The CUT macro command saves lines to one of eleven named clipboards for later retrieval by the PASTE command. The lines can be appended to lines already saved by a previous CUT command or the lines can replace the existing contents of a clipboard.

Syntax



linenum1
Relative line number identifying the start of a group of lines in the current member that are to be added to, or replace, data in the clipboard.

linenum2
Relative line number identifying the end of a group of lines in the current member that are to be added to, or replace, data in the clipboard.

labela, labelb
Labels identifying the start and end of the group of lines in the current member that are to be added to, or replace, data in the clipboard.

clipboard_name
The name of the clipboard to use. If you omit this parameter, the ISPF default clipboard (named DEFAULT) is used. You can define up to ten additional clipboards. The size of the clipboards and number of clipboards might be limited by installation defaults.

X|NX
Specify X to cut only lines that are excluded from the display. Specify NX to cut only lines that are not excluded from the display. The default is to cut all lines in the range (both excluded and nonexcluded lines) to the clipboard.

REPLACE|APPEND
Specify REPLACE to replace existing data in the clipboard. If you do not specify REPLACE, the lines in the current CUT are added to the end of the existing data within the clipboard.

If you specify APPEND, you add the data to the clipboard. This is the default.

ASCII, EBCDIC, UTF8
When one of these keywords is supplied, if the data is using a different character set to that designated by the keyword, the data being placed in the clipboard is converted to the character set designated by the keyword and tagged as being in the designated character set.

Description

CUT saves copies of lines from an edit session to a clipboard for later retrieval by the PASTE command. The lines are copied from the session to the named clipboard. Lines are specified by label names on the CUT command. The edit macro CUT command always copies lines to the clipboard and does not delete them from the edit session.

If you specify a clipboard name, lines are copied to that clipboard. If the specified clipboard does not yet exist, it is created. ISPF provides a default clipboard named DEFAULT. You can use up to 10 other clipboards that you define. The defined clipboards exist as long as you are logged on to TSO and are deleted when you log off.

Return codes

0
Normal completion

12
Parameter error. Insufficient storage, or no more clipboards available.

20
Severe error

Examples

To save all the lines in the current file to the default clipboard, appending them to lines already in the clipboard:

```
ISREDIT CUT .ZFIRST .ZLAST
```

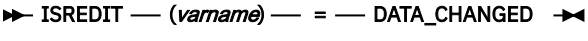
To save all the lines in the current file to a clipboard named USERC1, replacing any lines already in the clipboard:

```
ISREDIT CUT .ZFIRST .ZLAST USERC1 REPLACE
```

DATA_CHANGED—Query the Data Changed Status

The DATA_CHANGED assignment statement retrieves the current data-changed status and places it in a variable.

Syntax



varname
The name of a variable containing the data-changed status, either YES or NO. The data-changed status is initially set to NO at the beginning of an edit session, and is reset to NO whenever a save is

done. If you change data on your screen, but issue the END command, the data-changed status is still NO. When data is changed, or if a command is issued which might have changed the data, the changed status is set to YES.

Description

This command returns information about whether the data might have changed. However, it does not specify whether data is saved when the END command is issued. Data can be saved without being changed if there is a change to the version, number, stats, or pack mode. When DATA_CHANGED returns a value of NO, an 8 character variable called ZEDSAVE is set to indicate whether the data is saved. ZEDSAVE will contain either "SAVE " or "NOSAVE". See AUTOSAVE, CANCEL, SAVE and END for more information about saving data.

Return codes

0
Normal completion

20
Severe error

Examples

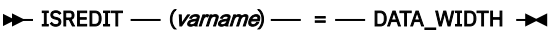
To determine whether data has been changed and, if it has, to issue the built-in SAVE command:

```
ISREDIT (CHGST) = DATA_CHANGED
IF &CHGST = YES THEN ISREDIT BUILTIN SAVE
```

DATA_WIDTH—Query Data Width

The DATA_WIDTH assignment statement retrieves the current logical data width and places it in a variable.

Syntax



varname
The name of a variable to contain the logical data width. The logical data width is a 3-digit value that is left-padded with zeros. If the variable is VDEFINEd in character format, it should be defined with a length of 5. The returned value is left padded with zeros. For compatibility with previous releases of ISPF, a length of 3 or 4 is allowed in cases where no data loss occurs.

Description

The logical data width is the maximum space, in bytes, that is available for data only. It does not include any COBOL or sequence number fields or, for variable-length records, the 4-byte record descriptor word (RDW).

The value returned by the DATA_WIDTH assignment statement depends on the record format (fixed or variable) and the setting of number mode, as shown in Table 20 on page 325. See “NUMBER—Generate Sequence Numbers” on page 260 if you need more information about number mode.

Table 20. Data width return value

Number mode setting	Logical data width for fixed-length records	Logical data width for variable-length records
OFF	LRECL	LRECL - 4
ON STD	LRECL - 8	LRECL - 12
ON COB	LRECL - 6	N/A “1” on page 325
ON STD COB	LRECL - 14	N/A “1” on page 325

Note:

1. COBOL numbering is invalid for variable-length records.

Use the LRECL assignment statement to get the maximum space, in bytes, that is available for data, COBOL number fields, and sequence number fields.

Return codes

0
Normal completion

12
Invalid command format

20
Severe error

Examples

To put the data width in variable &MAXCOL and override the boundary setting for SEEK:

```
ISREDIT (MAXCOL) = DATA_WIDTH
ISREDIT SEEK 1 &MAXCOL &ARGSTR
```

DATAID—Query Data ID

The DATAID assignment statement retrieves the data ID for the data set currently being edited and places it in a variable.

Syntax

varname
The name of a variable containing the data ID of the data set currently allocated for editing.

Description

The data ID is created by the LMINIT service to identify a data set.

If you begin an edit session with a data ID, the data ID is returned when you issue this command. If you begin an edit session without a data ID, then an LMINIT service obtains a data ID and returns it. On return from a top-level macro, the editor releases any data ID it has obtained.

For further information about the use of library access services, refer to the *z/OS ISPF Services Guide*.

Return codes

0
The data ID returned was passed to the editor

4
Data ID was generated by and is freed by the editor

8
A previously generated data ID was returned

20
Severe error

Examples

To store the data ID in variable &DID, and then find the member MEM1 of that data set by using the LMMFIND library access service:

```
ISREDIT (DID) = DATAID
ISPEXEC LMMFIND DATAID(DID) MEMBER(MEM1)
IF &LASTCC = 0 THEN ...
```

DATASET—Query the Current and Original Data Set Names

The DATASET assignment statement retrieves these items and places them in selected variables:

- The name of the data set into which the data currently being edited will be stored
- The name of the data set from which the data currently being edited originated
- The library concatenation number of the originating data set
- The path name of the file (when editing a z/OS UNIX file)

Syntax

var1
The name of a variable to contain the name of the data set currently being edited. The data set name is fully qualified without quotation marks (').

When editing a z/OS UNIX file, the path name of the file.

var2
The name of a variable to contain the name of the data set where the data currently being edited originated from. The data set name is fully qualified without quotation marks ('). If the data currently being edited is new, a blank is returned in this variable. If the original data is deleted, the name of the data set where the data currently being edited originated from is still returned in this variable.

var3
The library concatenation number of the original data set. If the data currently being edited is new, zeros are returned.

Return codes

0
Normal completion

20
Severe error

Examples

To place the name of the data set you are editing and the library concatenation number in the variables **&CURDSN** and **&LIBNUM**:

```
ISREDIT (CURDSN, ,LIBNUM) = DATASET
```

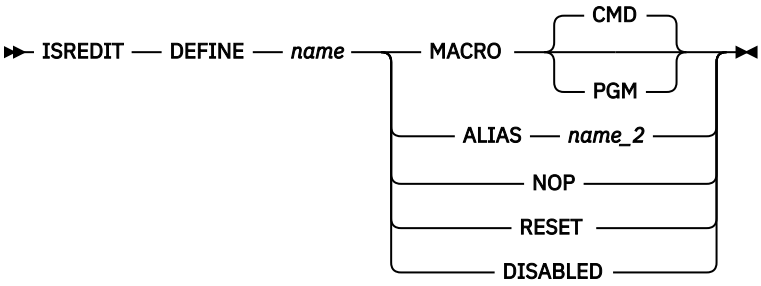
DEFINE—Define a Name

The DEFINE macro command is used to:

- Identify a macro that replaces a built-in command of the same name
- Identify programs that are edit macros
- Assign an alias to a macro or built-in command
- Make a macro or built-in command inoperable
- Reset an inoperable macro or built-in command
- Disable a macro or built-in command

DEFINE is often used with the BUILTIN command.

Syntax



name
The name with which you process the command.

MACRO CMD
Identifies the name that you are defining as a command language (CLIST or REXX exec) macro, which is called in the same way as using the SELECT service CMD keyword with a percent symbol (%) preceding the command. That means that you can specify only CLISTs or REXX EXECs.

MACRO PGM
Identifies the name that you are defining as a program (load module) macro, which is called by the SELECT PGM service.

ALIAS *name2*
Identifies the name that you are defining as an alias of another name, with the same characteristics. If *name2* is already an alias, the editor replaces it with the command it names. Therefore, it is not possible to have an alias of an alias.

NOP
Makes the name you are defining and all of its aliases inoperable until you reset them with the RESET operand. Therefore, when the name or an alias of the name is called, nothing is processed. NOP is similar to DISABLED, except that disabled names cannot be reset by the RESET operand.

RESET
Resets the most recent definition of the name that you are defining to the status in effect before that definition. For example, RESET makes inoperable names operable again.

DISABLED
Makes the name that you are defining and all of its aliases disabled until you end the edit session. Therefore, when the name or an alias of the name is called, nothing is processed. A disabled command or macro cannot be restored by RESET.

Description

The effects of the DEFINE macro command apply only to the edit session of the member or sequential data set being edited when the macro is run. This effect is different from the DEFINE primary command.

To temporarily override DEFINE, use BUILTIN.

Note: To define RESET as disabled, enclose it in quotes (' RESET '). If you do not use quotes, the editor interprets RESET as a keyword.

Return codes

0
Normal completion

8
RESET was attempted for a name not currently defined, or DEFINE name ALIAS *name2* requested and *name2* is an NOP

12
DEFINE was attempted for a name not currently defined

20
Severe error (unknown command)

Examples

To define the name IJKDOIT as a CLIST or REXX macro:

```
ISREDIT DEFINE IJKDOIT MACRO
```

To define the name SETITUP as a program macro:

```
ISREDIT DEFINE SETITUP MACRO PGM
```

To define the name DOIT as an alias of the macro IJKDOIT:

```
ISREDIT DEFINE DOIT ALIAS IJKDOIT
```

To define the name SAVE to have no effect:

```
ISREDIT DEFINE SAVE NOP
```

To reset the definition of the name SAVE:

```
ISREDIT DEFINE SAVE RESET
```

To define the name FINDIT as disabled:

```
ISREDIT DEFINE FINDIT DISABLED
```

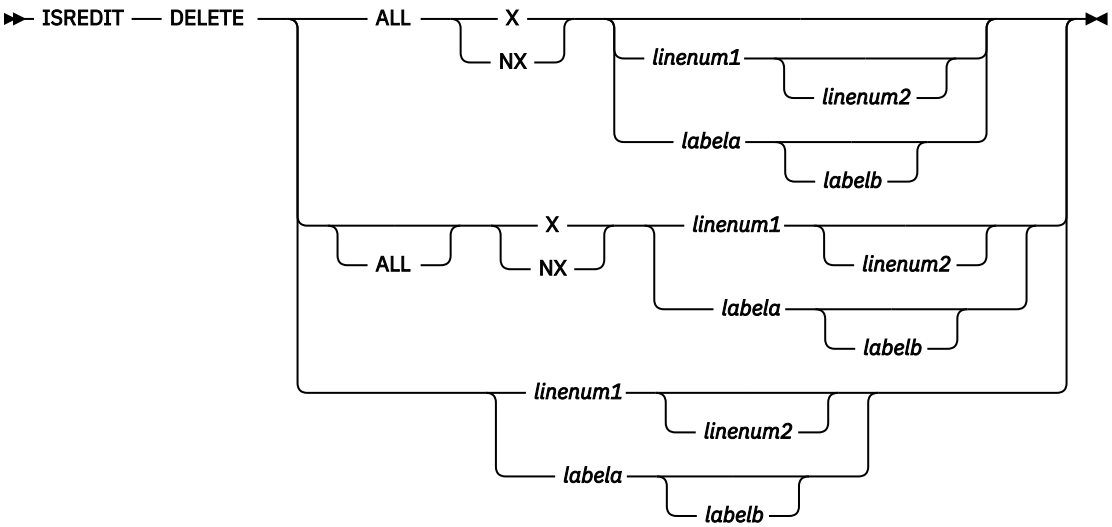
To create and update library statistics when data is saved, first set the stats mode on. Then make it impossible to turn off by defining it as disabled. Note that none of the commands that are defined as disabled can be called while you are editing a member.

```
ISREDIT MACRO
ISREDIT STATS ON
ISREDIT DEFINE STATS DISABLED
```

DELETE—Delete Lines

The DELETE macro command deletes lines from the data you are editing.

Syntax



ALL
Specifies that all selected lines are deleted. The DELETE command, unlike FIND, CHANGE, and EXCLUDE, does not use NEXT, FIRST, PREV, or LAST. ALL is required to emphasize that NEXT is not the default.

X
Restricts the lines deleted to those that are excluded.

NX
Restricts the lines deleted to those that are not excluded.

labela*, *labelb
Labels identifying the start and end of the group of lines to be deleted. To delete one line, enter one label.

linenum1
Relative line number identifying a line, or the start of a group of lines, to be deleted.

linenum2
Relative line number identifying the end of a group of lines to be deleted.

Description

DELETE can specify a single line or a range of lines. It can limit the lines to be deleted to all excluded or non-excluded lines in the data, or to all excluded or non-excluded lines within a line pointer range.

Return codes

0
Normal (lines deleted successfully)

4
No lines deleted

8
No standard records exist

12
Invalid line number

20
Severe error

Examples

To delete all non-excluded lines:

```
ISREDIT DELETE ALL NX
```

To delete all lines between labels .A and .B with a blank in column 1:

```
ISREDIT RESET X .A .B
ISREDIT EXCLUDE ALL " " 1 .A .B
ISREDIT DELETE ALL X .A .B
```

To delete the last line of data in the current data set:

```
ISREDIT DELETE .ZLAST
```

To delete the first 10 lines of data in the current data set:

```
ISREDIT DELETE 1 10
```

DISPLAY_COLS—Query Display Columns

The DISPLAY_COLS assignment statement retrieves the column numbers of the first and last data columns that you are seeing, and places them in variables.

Syntax

var1
The name of a variable containing the column number of the first data column visible to you. The column number is a 3-digit value that is left-padded with zeros. If the variable is VDEFINEd in character format, it should be defined with a length of 5. The returned value is left padded with zeros. For compatibility with previous releases of ISPF, a length of 3 or 4 is allowed in cases where no data loss will occur.

var2
The name of a variable containing the column number of the last data column visible to you. The column number is a 3-digit value that is left-padded with zeros. If the variable is VDEFINEd in character format, it should be defined with a length of 5. The returned value is left padded with zeros. For compatibility with previous releases of ISPF, a length of 3 or 4 is allowed in cases where no data loss will occur.

Description

Columns that contain sequence numbers are not considered data columns. Do not use this assignment statement in initial macros because the columns displayed are not known until the data first appears. See “Referring to column positions” on page 106 for more information.

Return codes

0
Normal completion

12
Invalid command format

20
Severe error

Examples

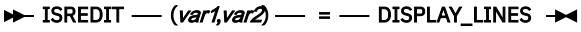
To put the leftmost and rightmost column values displayed to you in variables &LEFT and &RIGHT:

```
ISREDIT (LEFT,RIGHT) = DISPLAY_COLS
```

DISPLAY_LINES—Query Display Lines

The DISPLAY_LINES assignment statement retrieves the relative line numbers of the first and last data lines that would appear at this point if the macro ended, and places them in variables. Other non-data lines might be on the display. Do not use this assignment statement in an initial macro because the lines displayed are not known until the data is first displayed.

Syntax



var1
The name of a variable containing the relative line number of either the first visible data line or block of excluded lines if the macro ended at this point. The relative line number is a 6-digit value that is left-padded with zeros. If the variable is VDEFINEd in character format, it should be defined with a length of 8. The returned value is left-padded with zeros. For compatibility with previous releases of ISPF, a length of 6 or 7 is allowed in cases where no data loss will occur.

var2
The name of a variable containing the relative line number of either the last visible data line or block of excluded lines. The relative line number is a 6-digit value that is left-padded with zeros. If the variable is VDEFINEd in character format, it should be defined with a length of 8. The returned value is left-padded with zeros. For compatibility with previous releases of ISPF, a length of 6 or 7 is allowed in cases where no data loss will occur.

Return codes

0
Normal completion

4
No visible data lines

8
No existing data lines

12
Invalid command format

20
Severe error

Examples

To place the top and bottom line numbers in variables &TOP and &BOT:

```
ISREDIT (TOP,BOT) = DISPLAY_LINES
```

DOWN—Scroll Down

The DOWN macro command scrolls data down from the current panel position.

Syntax

amt
The number of lines (0-9999) to scroll, or one of these operands:

MAX
Scrolls to the end of data in the specified direction.

HALF
Displays the next sequential half panel of data.

PAGE
Displays the next sequential full panel of data.

CURSOR
Scrolls until the line on which the cursor is located becomes the first data line on the panel.

DATA
Scrolls until the last data line on the current panel of data becomes the first data line on the next panel of data.

Description

To scroll down using the panel position when the macro was first issued, use USER_STATE assignment statements to save and then restore the panel position operands.

When you issue DOWN, the non-data lines on the panel affect the number of lines scrolled. However, if you define a macro named DOWN, it only overrides the DOWN command when used from another macro. DOWN does not change the cursor position and cannot be used in an initial macro.

The actual number of lines appearing on the panel is determined by:

- The number of lines excluded from the display
- The terminal display size and split-panel line
- The number of special temporary lines appearing, such as the ==ERR>, ==CHG>, =COLS>, ======, =PROF>, ==MSG>, =NOTE=, =BNDS>, =TABS> or =MASK> lines

The first line appearing is determined in one of two ways: (1) a LOCATE command can set the line first on the panel, and (2) the first line to appear depends on whether the cursor was set explicitly by a CURSOR assignment statement or implicitly by a SEEK, FIND, CHANGE, or TSPLIT command. Since the cursor must be on the panel, the line that is the first line on the panel may be different from the line that was first when you called the macro.

Return codes

0
Normal completion

2
No more data DOWN

4
No visible lines

8
No data to display

12
Amount not specified

20
Severe error

Examples

To scroll down to the end of the data set:

```
ISREDIT DOWN MAX
```

To display the next half panel of data:

```
ISREDIT DOWN HALF
```

To display the next full panel of data:

```
ISREDIT DOWN PAGE
```

To make the line where the cursor is placed the first one on the display:

```
ISREDIT DOWN CURSOR
```

To display the next page less one line:

```
ISREDIT DOWN DATA
```

EDIT—Edit from within an Edit Session

The EDIT macro command allows you to edit a member of the same partitioned data set during your current edit session.

Syntax

member
A member of the library or other partitioned data set you are currently editing. You may enter a member pattern to generate a member list.

Description

Editing one data set or member while you are already editing another is called *recursive editing*. Your initial edit session is suspended until the second-level edit session is complete. Editing sessions can be nested until you run out of storage.

To exit from a nested edit session, END or CANCEL must be processed by a macro or entered by you. The current edit session resumes.

The EDIT service call, ISPEXEC EDIT, is an alternate method of recursively starting the editor. It offers the option of editing another data set and specifying an initial macro.

For more information on using the EDIT service for recursive editing, refer to the *z/OS ISPF Services Guide*.

Return codes

0
Normal completion, data was saved

4
Normal completion, data was *not* saved

12
Your error (invalid member name, recovery pending)

14
Member in use

20
Severe error

28
No ISREDIT MACRO statement preceded this call, or BROWSE was substituted because of the size of the member being edited.

Examples

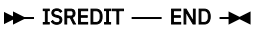
To recursively edit the member OLDMEM in your current ISPF library:

```
ISREDIT EDIT OLDMEM
```

END—End the Edit Session

The END macro command ends the editing of the current sequential data set or partitioned data set member.

Syntax



Description

If an edit macro contains an ISREDIT END statement, there can be no other ISREDIT or ISPEXEC statements following it. If one of these kinds of statements does follow an ISREDIT END, the edit macro ends with an error when that statement occurs. However, any other CLIST, REXX exec, or program statements can follow an ISREDIT END statement and process normally.

If no aliases have been defined for END, the response of the editor to the END command depends on:

- Whether changes were made to the data during your current edit session
- If changes were made, whether a SAVE command was entered after the last change
- The setting of number mode, autonum mode, stats mode, autolist mode, and autosave mode in the edit profile
- Whether you were editing a member that was an alias of another member

Note: When END is entered in the macro field in the edit prompt panel (ISRUEDIT), the macro name is not saved in the profile for use in future sessions. This is to avoid having the editor appear to do nothing when it is invoked from the data set list.

See “Ending an edit session” on page 12 for more information.

Return codes

0
Normal completion

4
New member saved

12
END not done, AUTOSAVE OFF PROMPT set, or Data not saved (insufficient space)

20
Severe error

Examples

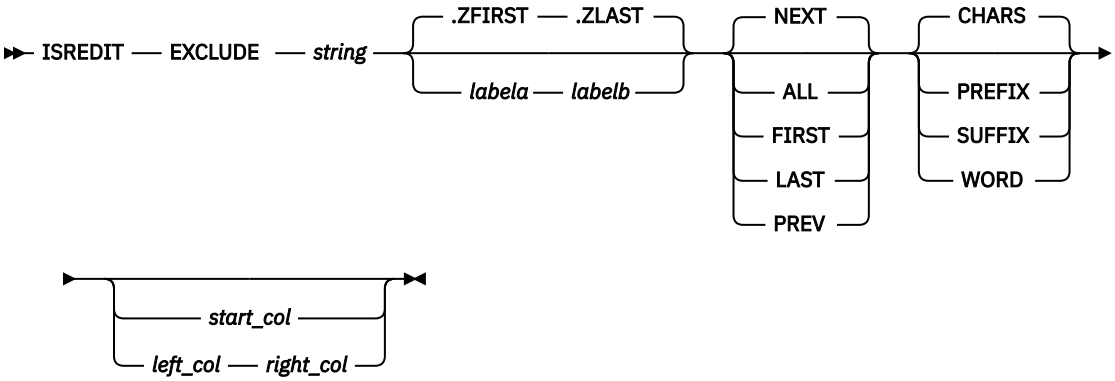
To end the current edit session:

```
ISREDIT END
```

EXCLUDE—Exclude Lines from the Display

The EXCLUDE macro command hides lines that contain a search string from view, and replaces them with a dashed line. To see the lines again, you enter either the RESET or RESET EXCLUDED command.

Syntax



string
The search string you want to exclude. See “Finding, seeking, changing, and excluding data” on page 44.

Note: For edit macros written in CLIST, strings that contain an open comment delimiter (/*) must be placed within quotes within the &STR() such as &STR('/*XXX'). The maximum allowable length of the string is 256 bytes. If you are specifying a hex string, the maximum is 128 hexadecimal characters.

labela*, *labelb
Labels identifying the start and end of the group of lines within which the EXCLUDE command is to search.

If the cursor is currently placed above the start label and the PREV occurrence of a string is requested, or the cursor is currently placed below the end label and the NEXT occurrence of a string is requested, the process returns a return code of 4 and the string is not found, even if it exists within the label range.

For more information about using labels to identify a group of lines, see "Labels and line ranges" on page 59.

NEXT
Starts at the first position after the current cursor location and searches ahead to find the next occurrence of *string*.

ALL
Starts at the top of the data and searches ahead to find all occurrences of *string*.

FIRST
Starts at the top of the data and searches ahead to find the first occurrence of *string*.

LAST
Starts at the bottom of the data and searches backward to find the last occurrence of *string*.

PREV
Starts at the current cursor location and searches backward to find the previous occurrence of *string*.

CHARS
Locates *string* anywhere the characters match.

PREFIX
Locates *string* at the beginning of a word.

SUFFIX
Locates *string* at the end of a word.

WORD
Locates *string* when it is delimited on both sides by blanks or other non-alphanumeric characters.

start_col
The first column to be included in the range of columns to be searched. When you specify only one column, the editor finds the string only if the string starts in the specified column.

left_col
The first column to be included in the range of columns to be searched.

right_col
The last column to be included in the range of columns to be searched.

Note: For more information about restricting the search to only a portion of each line, see "Limiting the search to specified columns" on page 54.

Description

You can use the EXCLUDE command with the FIND and CHANGE commands to find a search string, change it, and then exclude the line that contains the string from the panel.

To exclude the next non-excluded line that contains the letters ELSE without specifying any other qualifications, include this command in an edit macro:

```
ISREDIT EXCLUDE ELSE
```

Since no other qualifications were specified, the letters ELSE can be:

- Uppercase or a mixture of uppercase and lowercase
- At the beginning of a word (prefix), the end of a word (suffix), or the entire word (word)
- Anywhere within the current boundaries

To exclude the next line that contains the letters ELSE, but only if the letters are uppercase, include this command in an edit macro:

```
ISREDIT EXCLUDE C'ELSE'
```

This type of exclusion is called a character string exclusion (note the C that precedes the search string) because it excludes the next line that contains the letters ELSE only if the letters are found in uppercase.

However, since no other qualifications were specified, the exclusion occurs no matter where the letters are found on a non-excluded line, as outlined in the previous list.

For more information, including other types of search strings, see "Finding, seeking, changing, and excluding data" on page 44.

Return codes

0
Normal completion

4
String not found

8
Lines not excluded

12
Inconsistent parameters

20
Severe error

Examples

This example excludes the first non-excluded line in the data set that contains the letters ELSE. However, the letters must occur on or between lines labeled .E and .S and they must be the first four letters of a word:

```
ISREDIT EXCLUDE ELSE .E .S FIRST PREFIX
```

This example excludes the last non-excluded line in the data set that contains the letters ELSE. However, the letters must occur on or between lines labeled .E and .S and they must be the last four letters of a word.

```
ISREDIT EXCLUDE ELSE .E
.S LAST SUFFIX
```

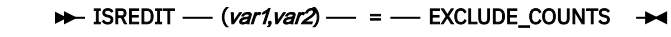
This example excludes the first non-excluded line that immediately precedes the cursor position and that contains the letters ELSE. However, the cursor must not be positioned ahead of the lines labeled .E and .S. Also, the letters must occur on or between the labeled lines; they must be standalone characters (not part of any other word); and they must exist within columns 1 and 5:

```
ISREDIT EXCLUDE ELSE .E .S PREV WORD 1 5
```

EXCLUDE_COUNTS—Query Exclude Counts

The EXCLUDE_COUNTS assignment statement retrieves values set by the most recently processed EXCLUDE command and places them in variables.

Syntax



var1
The name of a variable to contain the number of strings found. The number of strings is an 8-digit value that is left-padded with zeros.

var2
The name of a variable to contain the number of lines excluded. The number of lines excluded is an 8-digit value that is left-padded with zeros.

Return codes

0
Normal completion

12
Invalid command format

20
Severe error

Examples

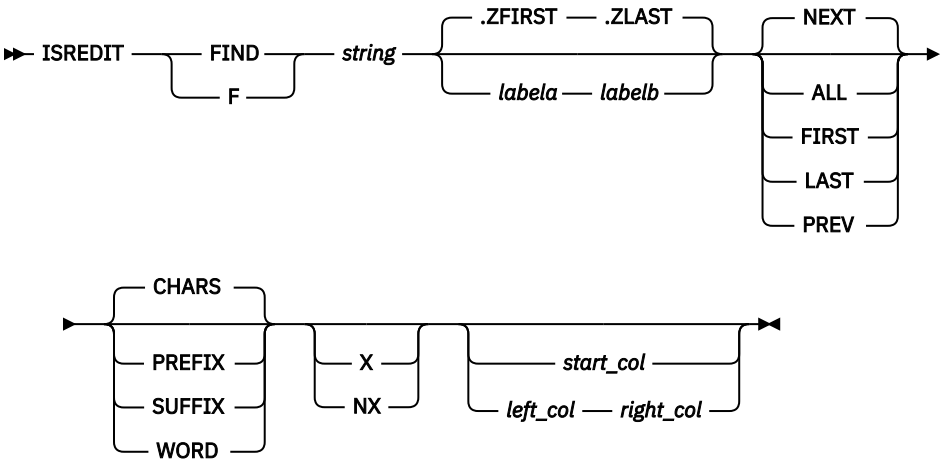
To determine the number of lines that contain the word BOX:

```
ISREDIT EXCLUDE ALL BOX
ISREDIT (,BOXLINES) = EXCLUDE_COUNTS
```

FIND—Find a Search String

The FIND macro command locates one or more occurrences of a search string.

Syntax



string
The search string you want to find. See “Finding, seeking, changing, and excluding data” on page 44.

Note: For edit macros written in CLIST, strings that contain an open comment delimiter (/*) must be placed within quotes within the &STR() such as &STR('/*XXX'). The maximum allowable length of the string is 256 bytes. If you are specifying a hex string, the maximum is 128 hexadecimal characters.

labela*, *labelb
Labels identifying the start and end of the group of lines within which the FIND command is to search.

If the cursor is currently placed above the start label and the PREV occurrence of a string is requested, or the cursor is currently placed below the end label and the NEXT occurrence of a string is requested, the process returns a return code of 4 and the string is not found, even if it exists within the label range.

For more information about using labels to identify a group of lines, see "Labels and line ranges" on page 59.

NEXT
Starts at the first position after the current cursor location and searches ahead to find the next occurrence of *string*.

ALL
Starts at the top of the data and searches ahead to find all occurrences of *string*.

FIRST
Starts at the top of the data and searches ahead to find the first occurrence of *string*.

LAST
Starts at the bottom of the data and searches backward to find the last occurrence of *string*.

PREV
Starts at the current cursor location and searches backward to find the previous occurrence of *string*.

CHARS
Locates *string* anywhere the characters match.

PREFIX
Locates *string* at the beginning of a word.

SUFFIX
Locates *string* at the end of a word.

WORD
Locates *string* when it is delimited on both sides by blanks or other non-alphanumeric characters.

X
Scans only lines that are excluded from the display.

NX
Scans only lines that are not excluded from the display.

start_col
The first column to be included in the range of columns to be searched. When you specify only one column, the editor finds the string only if the string starts in the specified column.

left_col
The first column to be included in the range of columns to be searched.

right_col
The last column to be included in the range of columns to be searched.

Note: For more information about restricting the search to only a portion of each line, see "Limiting the search to specified columns" on page 54.

Description

Use the SEEK macro command instead of FIND if you want to locate a string without changing the exclude status of the line that contains the string.

You can use FIND with the EXCLUDE and CHANGE commands to find a search string, change it, and then exclude the line that contains the string from the panel.

To find the next occurrence of the letters ELSE without specifying any other qualifications, include this line in an edit macro:

```
ISREDIT FIND ELSE
```

Since no other qualifications were specified, the letters ELSE can be:

- Uppercase or a mixture of uppercase and lowercase
- At the beginning of a word (prefix), the end of a word (suffix), or the entire word (word)
- In either an excluded or a non-excluded line

- Anywhere within the current boundaries

To find the next occurrence of the letters ELSE, but only if the letters are uppercase:

```
ISREDIT FIND C'ELSE'
```

This type of search is called a character string search (note the C that precedes the search string) because it finds the next occurrence of the letters ELSE only if the letters are in uppercase. However, since no other qualifications were specified, the letters can be found anywhere in the data set or member, as outlined in the preceding list.

For more information, including other types of search strings, see "Finding, seeking, changing, and excluding data" on page 44.

Return codes

0
Normal completion

4
String not found

12
Syntax error

20
Severe error

Examples

The example shown here finds the first occurrence in the data set of the letters ELSE. However, the letters must occur on or between lines labeled .E and .S and they must be the first four letters of a word:

```
ISREDIT FIND ELSE .E .S FIRST PREFIX
```

The example shown here finds the last occurrence in the data set of the letters ELSE. However, the letters must occur on or between lines labeled .E and .S; they must be the last four letters of a word; and they must be found in an excluded line.

```
ISREDIT FIND ELSE .E .S LAST SUFFIX X
```

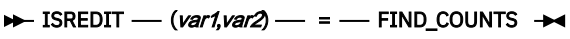
The example shown here finds the first occurrence of the letters ELSE that immediately precedes the cursor position. However, the cursor must not be positioned ahead of the lines labeled .E and .S. Also, the letters must occur on or between lines labeled .E and .S; they must be standalone characters (not part of any other word); they must be found in a non-excluded line; and they must exist within columns 1 and 5:

```
ISREDIT FIND ELSE .E .S PREV WORD NX 1 5
```

FIND_COUNTS—Query Find Counts

The FIND_COUNTS assignment statement retrieves values that were set by the most recently entered FIND or RFIND command, and places these values in variables.

Syntax



var1
The name of a variable to contain the number of strings found. The number of strings is an 8-digit value that is left-padded with zeros.

var2
The name of a variable to contain the number of lines on which strings were found. The number of lines on which strings were found is an 8-digit value that is left-padded with zeros.

Return codes

0
Normal completion

12
Invalid command format

20
Severe error

Examples

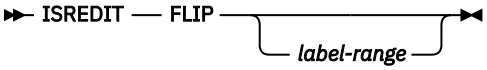
To find all occurrences of && in the line labeled .A and loop through and process them:

```
ISREDIT FIND .A .A && ALL
ISREDIT (FINDS) = FIND_COUNTS
DO WHILE &FINDS > 0
  ...
END
```

FLIP—Reverse Exclude Status of Lines

The FLIP macro command lets you reverse the exclude status of a specified range of lines or of all the lines in a file, including data, information, message, and note lines.

Syntax



labela*, *labelb
Labels identifying the start and end of the group of lines within which the FLIP command is to reverse the exclude status. If one label is specified, only that labeled line is reversed.

For more information about using labels to identify a group of lines, see "Labels and line ranges" on page 59.

Return codes

0
Successful completion. The excluded status of the requested lines was reversed.

20
Severe error

Examples

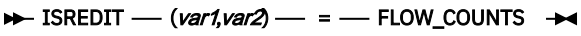
These are examples of statements using the FLIP commands from an Edit macro. The actual values for .a and .b can be defined by edit macro or by the user:

```
ISREDIT FLIP          /* Flip all lines                              */
ISREDIT FLIP .ZL .ZF  /* Flip all lines                              */
ISREDIT FLIP .ZF      /* Flip first line in file                     */
ISREDIT FLIP .a .b    /* Flip lines between and including .a and .b  */
ISREDIT FLIP .a       /* Flip line labeled .a                        */
```

FLOW_COUNTS—Query Flow Counts

The FLOW_COUNTS assignment statement retrieves values that were set by the most recently entered TFLOW command, and places these values in variables.

Syntax



var1
The name of a variable to contain the number of original lines that participated in the text flow operation. The number of original lines is an 8-digit value that is left-padded with zeros.

var2
The name of a variable to contain the number of lines that were generated by the text flow operation. The number of lines is an 8-digit value that is left-padded with zeros.

If the value in *var1* is larger than the value in *var2*, the difference is the number of lines that were deleted from the current data because of the text flow operation. If the value in *var1* is less than the value in *var2*, the difference is the number of lines that were added to the current data because of the text flow operation.

Return codes

0
Normal completion

20
Severe error

Examples

To retrieve the value of the rightmost column displayed, allow a margin of 8 for the text flow, and then take action if lines were added because of the text flow operation:

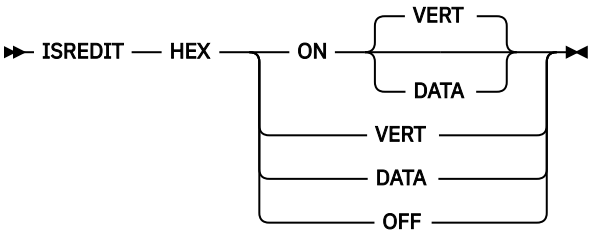
```
ISREDIT (,MAXCOL) = DISPLAY_COLS
ISREDIT TFLOW .ZCSR &EVAL(MAXCOL - 8)
ISREDIT (INLINE,OUTLIN) = FLOW_COUNTS
IF &OUTLIN > &INLINE THEN DO
  ...
```

HEX—Set or Query Hexadecimal Mode

The HEX macro command sets hexadecimal mode, which determines whether data appears in hexadecimal format.

The HEX assignment statement either sets hexadecimal mode or retrieves the current values of hexadecimal mode, and places them in variables.

Syntax

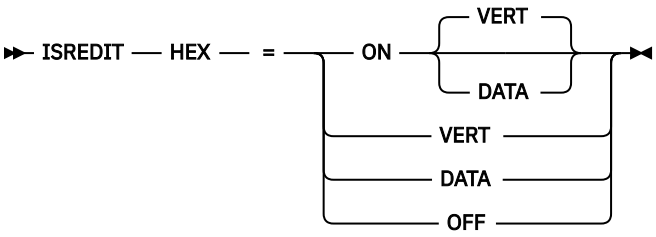


ON DATA
Displays the hexadecimal representation of the data as a string of hexadecimal characters (two per byte) under the characters.

ON VERT
Displays the hexadecimal representation of the data vertically (two rows per byte) under each character.

OFF
Does not display hexadecimal representation of the data.

Note: The command, HEX OFF, cancels the effect of any previous HX or HXX commands.



var1
The name of a variable to contain ON or OFF.

var2
The name of a variable to contain DATA, VERT, or blanks.

ON DATA
Same as macro command syntax.

ON VERT
Same as macro command syntax.

OFF
Same as macro command syntax.

Description

The HEX macro command and assignment statement determines whether the editor displays hexadecimal representation in a vertical or data string format.

When the editor is operating in hexadecimal mode, three lines are displayed for each source line. The first line shows the data in standard character form, while the next two lines show the same data in hexadecimal representation.

Besides normal editing on the first of the three lines, you can change any characters by typing over the hexadecimal representations.

You can also use the FIND, CHANGE, and EXCLUDE commands to find, change, or exclude invalid characters or any specific hexadecimal character, regardless of the setting of hexadecimal mode. See the discussion of picture strings and hexadecimal strings under “Finding, seeking, changing, and excluding data” on page 44.

Return codes

0
Normal completion

20
Severe error

Examples

To put the value of hexadecimal mode (on or off) in variable &HEXMODE and to process if hexadecimal mode is on:

```
ISREDIT (HEXMODE) = HEX
IF &HEXMODE = ON THEN -
   ...
```

To turn hexadecimal mode off:

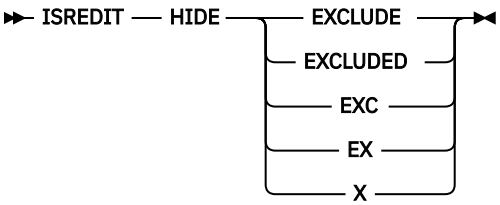
```
ISREDIT HEX OFF
```

HIDE—Hide Excluded Lines Message

The HIDE command removes the "*n* Line(s) not Displayed" messages from the display where lines have been excluded by the EXCLUDE command.

The HIDE function has dependencies on the value of the ISPF variable ZHIDEX, panel attributes, and extended highlighting support of the terminal. These dependencies are described in the section "Providing customized Browse and Edit panels" in the *ISPF Planning and Customisation Guide*.

Syntax



X
Removes each "*n* Line(s) not Displayed" message from the display and underscores the line number field of the preceding line.

Return codes

0
Successful completion. Any "*n* Line(s) not Displayed" messages were removed from the display.

4
HIDE X not supported

20
Severe error

Examples

These statements show how to use the HIDE command from an Edit macro to hide excluded lines, then the RESET HIDE command to display the excluded lines again:

```
ISREDIT HIDE X        /* Hide excluded lines                        */
ISREDIT RESET HIDE    /* Redisplay excluded lines                   */
```

HILITE—Enhanced Edit Coloring

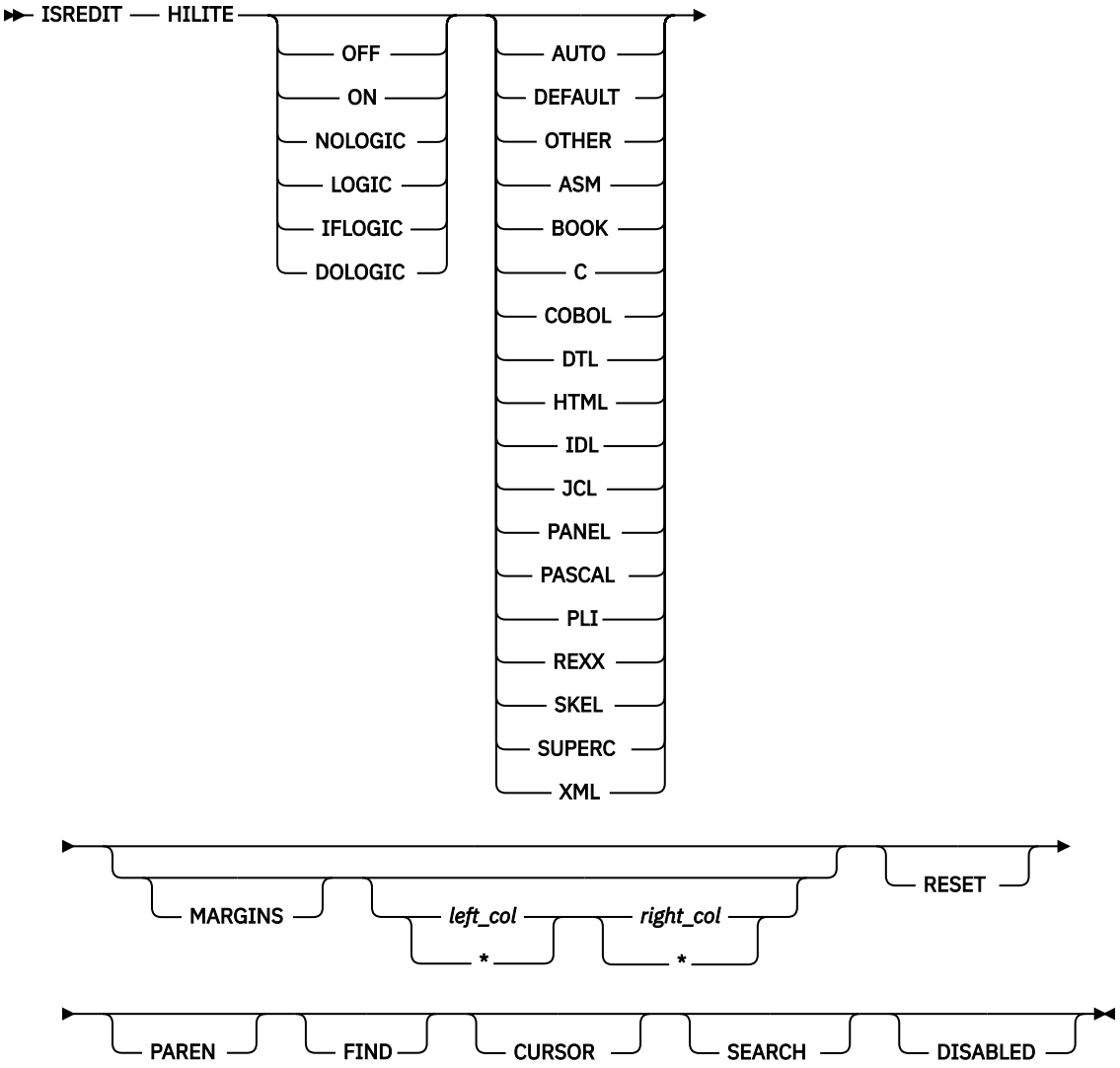
HILITE is used to control the use of color in the editor by changing the settings for the enhanced color and language-sensitive editing features.

The HILITE dialog is not available in the Edit Macro environment.

Note: Language sensitive and enhanced coloring of the edit session is only available if it is enabled by the installer or person who maintains the ISPF product. For information on enabling the enhanced color functions, see *z/OS ISPF Planning and Customizing*.

Language and logic hiliting is not supported for ASCII or UTF-8 editing sessions and the HILITE command is not available during these edit sessions.

Syntax



ON
Sets program coloring ON and turns LOGIC coloring off.

OFF
Sets coloring OFF, with the exception of cursor highlighting.

LOGIC
LOGIC highlighting matches logical language-specific keywords in the same color. If an unmatched *closing* keyword is found, such as END for PL/I or :eul. for BookMaster, it is highlighted in reverse video pink *only* if HILITE LOGIC is active. When logic is being highlighted, only comments are highlighted along with it.

Logic highlighting is available only for PL/I, PL/X, REXX, OTHER, C, SKELS, Pascal, and BookMaster. HILITE LOGIC turns on both IFLOGIC and DOLOGIC.

Note: LOGIC highlighting can be turned off by issuing HILITE ON, HILITE NOLOGIC, or HILITE RESET commands. Changing the HILITE language does not change the LOGIC setting.

IFLOGIC
Turns on IF/ELSE logic matching. IFLOGIC matches IF and ELSE statements. When IFLOGIC is enabled, unmatched ELSE keywords are highlighted in reverse video pink.

DOLOGIC
Turns on DO/END logic matching. DOLOGIC matches logical blocks such as DO/END in PL/I or :ol/:eol in BookMaster. For the C language, DOLOGIC matches curly braces ({ and }). C trigraphs for curly

braces are not recognized and are not supported by DOLOGIC highlighting. When DOLOGIC is enabled, unmatched logical block terminators (such as END keywords in PL/I, :e tags in BookMaster or right braces (}) in C) are highlighted in reverse video pink.

NOLOGIC
Same as ON.

AUTO
Allows ISPF to determine the language.

DEFAULT
Highlights the data in a single color.

OTHER
Highlight the data as a pseudo-PL/I language.

ASM
Highlights the data as Assembler.

BOOK
Highlights the data as BookMaster.

C
Highlights the data as C.

COBOL
Highlights the data as COBOL.

DTL
Highlights the data as Dialog Tag Language.

HTML
Highlights the data as HTML.

IDL
Highlights the data as IDL.

JCL
Highlights the data as MVS Job Control Language.

PANEL
Highlights the data as ISPF Panel Language.

PASCAL
Highlights the data as Pascal.

PLI
Highlights the data as PL/I.

REXX
Highlights the data as REXX.

SKEL
Highlights the data as ISPF Skeleton Language.

SUPERC
Highlights the data as a SuperC Listing.

XML
Highlights the data as XML.

MARGINS [*left-margin* | * [*right-margin* | *]]
Specifies either or both of the left-margin or right-margin parameters for languages C, PL/I, and PL/X. The MARGINS keyword can be included on the same command that includes one of these languages. It cannot be specified when the language AUTO is specified, even if the language would subsequently be determined to be C, PL/I, or PL/X.

left-margin
The left hand margin for processing the language source. The value must be within the range as defined by the language. The maximum value is 254 for C, 100 for PL/I, and 65 for PL/X. If *left-margin* exceeds the last input column or if an asterisk (*) is specified, the default left margin is

obtained from the ISPF configuration table keyword for this language (HILITE_MARGIN_C, HILITE_MARGIN_PLI, or HILITE_MARGIN_PLX).

right-margin
The right hand margin for processing the language source. The value must be within the range as defined by the language. The maximum value is 255 for C, 200 for PL/I, and 80 for PL/X. If *right-margin* exceeds the last input column or if an asterisk (*) is specified, the default right margin is obtained from the ISPF configuration table keyword for this language (HILITE_MARGIN_C, HILITE_MARGIN_PLI, or HILITE_MARGIN_PLX).

RESET
Resets defaults (LANG AUTO, COLOR ON, LOGIC OFF, FIND ON and CURSOR ON).

PAREN
Toggles parenthesis matching. When parenthesis matching is active, only comments and quoted strings are specially colored. All other code appears in the default color. Note that extra parenthesis highlighting is always active when highlighting is active.

Parentheses within quoted strings and comments are not checked or highlighted by the parenthesis matching function.

FIND
The HILITE FIND command toggles the highlighting color of any string that would be found by an RFIND. The user can select the highlight color. The default is reverse video white.

Only non-picture strings are supported, and the only additional qualifiers recognized are hex strings (X'...'), character strings (C'...'), text strings (T'...'), WORD, PREFIX and SUFFIX, and boundaries specified in the FIND command. Hex strings may be highlighted. but non-displayable characters are not highlighted. Default bounds and labels are ignored when FIND strings are highlighted.

Because FIND highlighting is not quite as robust at the FIND command itself, the editor may highlight more occurrences of the FIND string than FIND would actually locate.

The RESET edit command has been enhanced, through the addition of a FIND operand, to temporarily disable the highlighting of FIND strings until the next FIND, RFIND, CHANGE, or RCHANGE command is issued. RESET with the FIND operand (or no operands at all), temporarily disables the highlighting of FIND strings.

CURSOR
The CURSOR operand toggles the highlighting of the phrase that contains the cursor in a user-selectable color. The default is white.

Cursor highlighting in Edit is performed in a manner similar to the way it is done in Browse. The entire phrase from the previous blank to the next blank is highlighted.

SEARCH
HILITE SEARCH finds the first unmatched END, ELSE, }, or) above the last displayed line on the panel. If a mismatched item is found, the file is scrolled so that the mismatch is at the top of the panel. The search for mismatches only occurs for lines above the last displayed line, so you may need to scroll to the bottom of the file before issuing the HI SEARCH command.

Search is not available for the when the DEFAULT language operand is used.

DISABLED
Turns off all HILITE features and removes all action bars. This benefits performance at the expense of function. Since DISABLED status is not stored in the edit profile, you need to reenter this operand each time you enter the editor. If ISREDIT HILITE DISABLED is issued by a macro, any attempts to restore highlighting within the same macro invocation are ignored.

Description

The HILITE macro command can be used to highlight, in user-specified colors, many language-specific constructs, program logic features, the phrase containing the cursor, and any strings that match the previous FIND operation or those that would be found by an RFIND or RCHANGE request. In addition,

when HILITE is entered with no operands, a dialog appears that allows you to set default colors for the data area in non-program files, for any characters typed since the previous Enter or function key entry, and for strings located by the FIND command.

Both HI and HILIGHT are valid synonyms for HILITE.

When the code page being used is not the English codepage, the HILITE primary command does not detect key sequences if the control character within the key sequence has a different binary representation in the code page being used from the binary representation in the codepage used for English. For example, in the C language, a '\' is used as an escape sequence character to influence the interpretation of the next character; however, the '\' has a different binary representation in different codepages.

Note: Highlighting is *not* available for edit sessions that involve:

- Data sets with record lengths greater than 255
- Mixed mode edit sessions (normally used when editing DBCS data)
- Formatted data

If a macro issues HILITE in any of these situations, a return code of 12 is set.

Return codes

0
Normal completion.

8
Logic or search not supported in the current environment. Invalid language.

12
Hilite dialog is invalid from an edit macro or Hilite not available because of the installation defaults or because the edit panel in use is not enabled for enhanced color.

20
Severe error. Possibly extra parameters.

IMACRO—Set or Query an Initial Macro

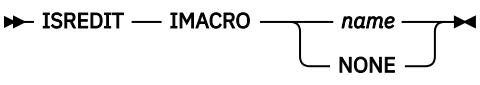
The IMACRO macro command saves the name of an initial macro in the current edit profile.

The IMACRO assignment statement sets or retrieves the value for the initial macro in the current profile, and places it in a variable.

See “Initial macros” on page 24 for more information on creating and using initial macros.

Syntax

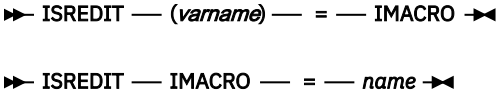
Macro command syntax



name
Identifies the initial macro to be run when editing the data set type that matches this profile. This macro is run before any data is displayed.

NONE
Shows that no macro is to be run at the beginning of each edit session. The editor returns a value of NONE when no initial macro has been specified.

Assignment statement syntax



varname
The name of a variable to contain the name of the initial macro.

name
Same as macro command syntax.

Return codes

0
Normal completion

4
IMACRO set not accepted; profile is locked

12
Invalid name specified

20
Severe error

Examples

To set the initial macro name to ISCRIPT:

```
ISREDIT IMACRO ISCRIPT
```

To set no initial macro:

```
ISREDIT IMACRO NONE
```

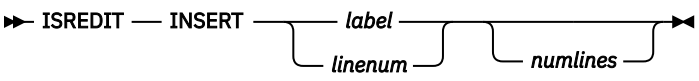
To store the name of the initial macro in the variable &IMACNAM:

```
ISREDIT (IMACNAM) = IMACRO
```

INSERT—Prepare Display for Data Insertion

The INSERT macro command inserts one or more blank lines, and allows you to fill them with data.

Syntax



label
A label that shows which line you want the inserted line or lines to follow.

linenum
A relative line number that shows which line you want the inserted line or lines to follow.

numlines
The number of lines to appear for data input; these lines are not saved until they contain data. If you do not type a number or if the number you type is 1, only one data input line appears.

Description

Use the INSERT macro command for data input. Inserted lines are initialized with data from the mask line. However, they are not data lines and cannot be referred to by any macro. Inserted lines are deleted if they do not contain data.

You must specify that the line referenced on INSERT should be displayed; otherwise, you will not see the inserted line. Use LOCATE to position a line at the top of the display.

Do not use this command for adding lines with specific data; instead, use the LINE_BEFORE and LINE_AFTER assignment statements.

Return codes

0
Normal completion

12
Invalid line number

20
Severe error

Examples

To open a 5-line area for data input after the line with the label .POINT, locate .POINT to position it to the top of the display. Then issue INSERT:

```
ISREDIT LOCATE .POINT
ISREDIT INSERT .POINT 5
```

LF—realign the data based on the ASCII linefeed character

The LF macro command allows you to realign the data being edited by interpreting the ASCII linefeed character X'0A'.

The LF macro command is not available when editing a z/OS UNIX file. Instead, use the ASCII edit facility to automatically realign the data in a z/OS UNIX file based on the ASCII linefeed and carriage return characters. See “Working with ASCII data” on page 50.

Note: If the data is saved, it is saved in the realigned state. There is no command to reverse the alignment. The command should not be executed twice against the data, as the blanks following the linefeed character will be interpreted as part of the data for the next line.

Syntax

See “Restructuring data based on the linefeed character” on page 51 for more information.

Examples

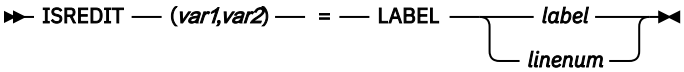
To realign the data being edited by interpreting the ASCII linefeed character X'0A':

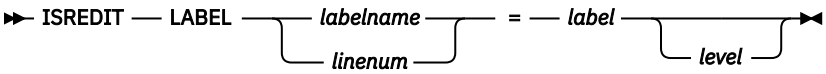
```
LF
```

LABEL—Set or Query a Line Label

The LABEL assignment statement sets or retrieves the values for the label on the specified line and places the values in variables.

Syntax





var1
The name of a variable to contain the name of the label.

var2
The name of a variable to contain the nesting level of the label. It must be a 3-character value that is left-padded with zeros.

label
A label identifying the line for which a label must be set or retrieved.

See the LOCATE and RESET command descriptions, which use labels to specify line ranges.

linenum
A relative line number identifying the line for which a label must be set or retrieved.

Use the LINENUM assignment statement to obtain the current relative line number of a line with a label.

labelname
The name of the label.

For more information about using labels, see “Labels and line ranges” on page 59.

The LINENUM assignment statement can be used to determine whether a label exists. For more information, see “LINENUM—Query the Line Number of a Labeled Line” on page 362.

level
The highest nesting level at which this label is visible to you or to a macro. Level 0 is the highest level. Labels at this level are visible to you and to all levels of nested macros. Level 1 is not visible to you, but it is visible to all macros, and so on. The level can never exceed the current nesting level. The maximum nesting level is 255. The level number defaults to the current nesting level.

Description

A range of labels is particularly useful for commands that operate on a range of lines, such as those in this list:

```
CHANGE   EXCLUDE   LOCATE    SEEK
CREATE   FIND      REPLACE   SORT
DELETE   FLIP      RESET     SUBMIT
```

Return codes

0
Normal completion

4
Label name not returned, specified line has no label

8
Label set, but an existing label at the same level was deleted

12
Line number specified is beyond the end of data

20
Severe error

Examples

To get the line of data at the cursor, look for the next occurrence of the string in the variable &ARG, and then label the line if it is found and currently unlabeled:

```
ISREDIT (NAME) = LINE .ZCSR
ISREDIT FIND &ARG
IF &LASTCC = 0 THEN -
  ISREDIT (LBL,NEST) = LABEL .ZCSR
IF &LBL=&STR() THEN -
  ISREDIT LABEL .ZCSR = .POINT 0
```

LEFT—Scroll Left

The LEFT macro command scrolls data to the left of the current panel position.

Syntax

amt
The scroll amount, the number of columns (0-9999) to scroll, or one of these operands:

MAX
Displays the first page of data to the left.

HALF
Displays the next half-panel of data to the left.

PAGE
Displays the next full panel of data to the left.

CURSOR
Scrolls until the column on which the cursor is located becomes the first data column on the panel.

DATA
Scrolls until the first column on the current panel of data becomes the last column on the next panel.

Description

The editor stops scrolling when it reaches the current BOUNDS setting. For example, if the left bound is position 9 and positions 21 to 92 are displayed, issuing ISREDIT LEFT 20 leaves positions 9 to 80 displayed, not 1 to 72.

To scroll to the left using the panel position when the macro was issued, use USER_STATE assignment statements to save and then restore the panel position operands.

If you define a macro named LEFT, it overrides the LEFT command when used from another macro. LEFT does not change the cursor position and cannot be used in an initial macro. For further information, see the BOUNDS and DISPLAY_COLUMNS descriptions.

Return codes

0
Normal completion

4
No visible lines

8
No data to display

12
Amount not specified

20
Severe error

Examples

To scroll the display to the left by the number of columns specified in variable &COL:

```
ISREDIT LEFT &COL
```

LEVEL—Set or Query the Modification Level Number

The LEVEL macro command allows you to control the modification level that is assigned to a member of an ISPF library.

The LEVEL assignment statement either sets the modification level or retrieves the current modification level and places it in a variable.

See “Version and modification level numbers” on page 26 for more information about level numbers.

Syntax

num
The modification level. It can be any number from 0 to 99.

varname
The name of a variable to contain the modification level. The modification level is a 2-digit value that is left-padded with zeros.

num
The modification level. A 2-digit value, left-padded with zeros.

Return codes

0
Normal completion

4
Statistics mode is off; the command is ignored

12
Invalid value specified

20
Severe error

Examples

To reset the modification level to 1:

```
ISREDIT LEVEL = 1
```

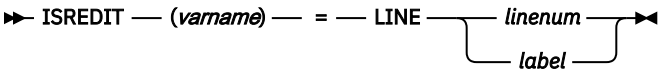
To save the value of the modification level in variable &MODLVL:

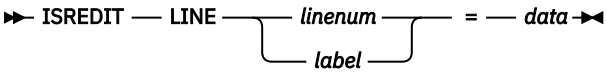
```
ISREDIT (MODLVL) = LEVEL
```

LINE—Set or Query a Line from the Data Set

The LINE assignment statement either sets or retrieves the data from the data line specified by a relative line number or label, and places it in a variable.

Syntax





varname
Specifies the name of a variable to hold the contents of the specified data line.

linenum
A relative line number identifying the data line.

label
A label identifying the data line.

data
Specifies that these forms can be used:

- Simple string
- Delimited string
- Variable
- Template (< *col*,*string* >)
- Merge format (*string1* + *string2*, operand + *string2*, *string1* + operand)
- Operand:

  LINE
  Data from this line is used.

  LINE *linenum*
  Data from the line with the given relative line number.

LINE *label*
Data from the line with the given label.

MASKLINE
Data from the mask line.

TABSLINE
Data from the tabs line.

Description

The logical data width of the line determines how many characters are retrieved or set. See the description of the DATA_WIDTH command for information on determining the current logical data width.

You must specify the line pointer to set or retrieve a line. To set data on a line, you can use a variety of data formats: (variable), templates, or merging a line with other data. The data on the line is completely overlaid with the data specified on this command.

Return codes

0
Normal completion

4
Data truncated (line shorter than data supplied)

8
Variable not found

12
Invalid line number

16
Variable data truncated

20
Severe error

Examples

To replace the data on line 7 with data from a variable named NEWDAT:

```
ISREDIT LINE 7 = (NEWDAT)
```

Note: This syntax is preferred to:

```
ISREDIT LINE 7 = &NEWDAT
```

because the variable is not rescanned by either the language processor or ISPF.

To set comment delimiters in columns 40 and 70, blanking the rest of the line:

```
ISREDIT LINE 1 = < 40 '&STR(/*)' 70 '&STR(*/)' >
```

To overlay the first 2 columns of line 2 with //:

```
ISREDIT LINE 2 = LINE + //
```

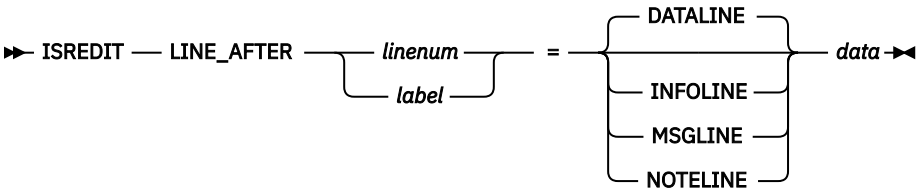
To merge mask line data with data from variable &VAR:

```
ISREDIT LINE 3 = MASKLINE + (VAR)
```

LINE_AFTER—Add a Line to the Current Data Set

The LINE_AFTER assignment statement adds a line after a specified line in the current data set.

Syntax



linenum
A relative line number identifying the data line after which the new line is to be inserted. A line pointer of 0 causes the new line to be inserted at the beginning of the current data set.

label
A label identifying the data line after which the new line is to be inserted.

DATALINE
The line inserted is a data line.

INFOLINE
The line inserted is a temporary, non-data line. The line command field shows `======` in high intensity and the data on the line is in high intensity, also. The line can be scrolled left and right and can be as long as the current record length. An information line is protected. Once it has been added to the data, it cannot be referenced.

MSGLINE
The line inserted is a temporary, non-data line. The line command field contains `==MSG>` in high intensity and the data on the line is also in high intensity. A message line has a data length of 72 characters, regardless of the data width. Once it has been added to the data, it cannot be referenced.

NOTELINE
The line inserted is a temporary, non-data line. The line command field shows `=NOTE=` in high intensity and the data on the line is in low intensity. A note line has a data length of 72 characters, regardless of the data width. It cannot be referenced after it is added to the data.

data
Specifies that these data formats can be used:

- Simple string
- Delimited string
- Variable
- Template (< *col*,*string* >)
- Merge format (*string1* + *string2*, operand + *string2*, *string1* + operand)
- Operand:

  LINE
  Data from the line preceding this line.

  LINE *linenum*
  Data from the line with the given relative line number.

  LINE *label*
  Data from the line with the given label.

  MASKLINE
  Data from the mask line.

TABSLINE
Data from the tabs line.

Description

This statement is used for adding lines with specific data. Use the INSERT command for data input.

Result: If the cursor is located within the data when the LINE_AFTER assignment statement is processed, the cursor is repositioned, if necessary, so that it remains on the same data after the statement is processed.

Return codes

0
Normal completion

4
Data truncated

12
Invalid line number

20
Severe error

Examples

To add data after line 4 with data from a variable named NEWDAT:

```
ISREDIT LINE_AFTER 4 = (NEWDAT)
```

Note: This syntax is preferred to `ISREDIT LINE_AFTER 4 = &NEWDAT` because the variable is not rescanned by either the language processor or ISPF.

To put a new line that contains the string:

```
This is the new top line of the data
```

as the first line of the data set:

```
ISREDIT LINE_AFTER 0 = "This is the new top line of the data"
```

To put the contents of the line labeled `.START` on a new line following the line labeled `.END`:

```
ISREDIT LINE_AFTER .END = LINE .START
```

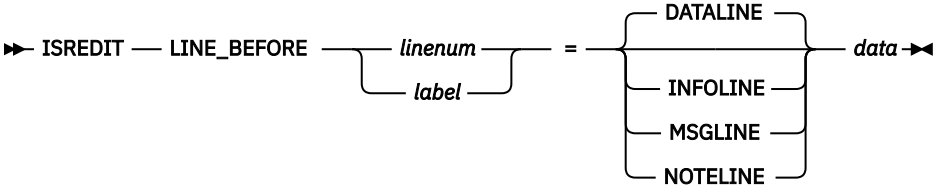
To put the contents of the mask line modified by the variable &DATA after the line whose number is in variable &N:

```
ISREDIT LINE_AFTER &N = MASKLINE + &DATA
```

LINE_BEFORE—Add a Line to the Current Data Set

The LINE_BEFORE assignment statement adds a line before a specified line in the current data set.

Syntax



linenum
A relative line number identifying the data line before which the new line is to be inserted. A line pointer of 0 is invalid.

label
A label identifying the data line before which the new line is to be inserted.

DATALINE
The line inserted is a data line.

INFOLINE
The line inserted is a temporary, non-data line. The line command field shows ====== in high intensity. The data on the line is shown in high intensity also. The line can be scrolled left and right and can be as long as the current record length. An information line is protected. Once it has been added to the data, it cannot be referenced.

MSGLINE
The line inserted is a temporary, non-data line. The line command field contains ==MSG> in high intensity. The data on the line is shown in high intensity also. A message line has a data length of 72 characters, regardless of the data width. Once it has been added to the data, it cannot be referenced.

NOTELINE
The line inserted is a temporary, non-data line. The line command field shows =NOTE= in high intensity. The data on the line is shown in low intensity. A note line has a data length of 72 characters, regardless of the data width. It cannot be referenced once it has been added to the data.

data
Specifies that these data formats can be used:

- Simple string
- Delimited string
- Variable
- Template (< *col*,*string* >)
- Merge format (*string1* + *string2*, operand + *string2*, *string1* + operand)
- Operand (those allowed follow):

  LINE
  Data from the line following this line.

  LINE *linenum*
  Data from the line with the given relative line number.

  LINE *label*
  Data from the line with the given label.

  MASKLINE
  Data from the mask line.

  TABSLINE
  Data from the tabs line.

Description

The LINE_BEFORE statement is used for adding lines with specific data. Use INSERT for data input.

Result: If the cursor is located within the data when the LINE_BEFORE assignment statement is processed, the cursor is repositioned, if necessary, so that it remains on the same data after the statement is processed.

Return codes

0
Normal completion

4
Data truncated

12
Invalid line number

20
Severe error

Examples

To add data before line 4 with data from a variable named NEWDAT:

```
ISREDIT LINE_BEFORE 4 = (NEWDAT)
```

Note: This syntax is preferred to `ISREDIT LINE_BEFORE 4 = &NEWDAT` because the variable is not rescanned by either the language processor or ISPF.

To put the contents of the line labeled `.START` on a new line preceding the line labeled `.END`:

```
ISREDIT LINE_BEFORE .END = LINE .START
```

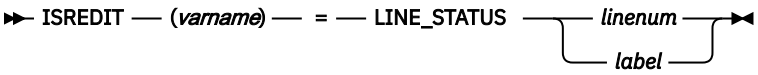
To put the contents of the mask line modified by the variable &DATA before the line whose number is in variable &N:

```
ISREDIT LINE_BEFORE &N = MASKLINE + &DATA
```

LINE_STATUS—Query Source and Change Information for a Line in a Data Set

The LINE_STATUS assignment statement retrieves the source and change information for the data line specified by a line pointer, and places it in a variable. This information indicates how the line was originally added to the data, and how it has been changed during the edit session.

Syntax



varname
The name of a variable to contain the status string for the specified line. This is a 32-character variable containing character 1s and 0s:

Characters 1-7 are "source" information.

Character 1
Line is an original record (it existed when the edit session started)

Character 2
Line was created by the Move line command

Character 3
Line was created by the Copy or Repeat line command

Character 4
Line was created by the MOVE primary or macro command

Character 5
Line was created by the COPY primary or macro command

Character 6
Line was created by the TE line command

Character 7
Line was created by the Insert line command

Characters 8-14 are "change" information.

Character 8
Line was changed (one of these characters will also be set to show HOW the line was changed)

Character 9
Data on the line was typed over

Character 10
Data was changed by the CHANGE primary command or the Overlay line command

Character 11
Data was changed by the Column Shift line command [used the (, ((,), or)) command]

Character 12
Data was changed by the Data Shift line command [used the <, <<, >, or >> command]

Character 13
Data was changed by the TE, TF, or TS line command

Character 14
The line was renumbered

Characters 15-32 are reserved for future use.

linenum
A relative line number identifying the data line.

label
A label identifying the data line.

Return codes

0
Normal completion

12
Line number not valid

20
Severe error

Examples

To determine if line number one of your data has changed and to display a message informing you of its status:

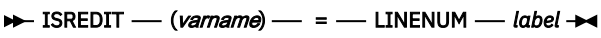
```
ISREDIT (LINESTAT) = LINE_STATUS 1
If linestat(1) = '1' Then
   Say 'Line is an ORIGINAL record'
Else
   Say 'Line was created during this edit session'
If linestat(8) = '1' Then
   Say 'Line has been changed'
Else
```

```
   Say 'Line has not been changed'
```

LINENUM—Query the Line Number of a Labeled Line

The LINENUM assignment statement retrieves the current relative line number of a specified label, and places it in a variable.

Syntax



varname
The name of a variable to contain the line number of the line with the specified label. The line number is a 6-digit value that is left-padded with zeros. If the variable is VDEFINEd in character format, it should be defined with a length of 8. The returned value is left-padded with zeros. For compatibility with previous releases of ISPF, a length of 6 or 7 is allowed in cases where no data loss will occur.

label
The name of the label for the line whose line number is needed.

Return codes

0
Normal completion

4
Line 0 specified

8
Label specified, but not found (variable set to 0)

12
Invalid line number

20
Severe error

Description

Once the line number is retrieved and placed in a variable, it can be used in arithmetic operations. Note that line numbers are relative to the position of the line: first=1, second=2, and so on. Therefore, the value returned by the LINENUM assignment statement is not always be correct if lines are added or deleted before the line number is obtained.

Examples

To determine the number of lines in the data set and set variable &VAR to the last line number:

```
ISREDIT (VAR) = LINENUM .ZLAST
```

That number is 0 if there are no lines.

To set variable &NUM to the line number containing the label `.MYLAB`:

```
ISREDIT (NUM) = LINENUM .MYLAB
```

LOCATE—Locate a Line

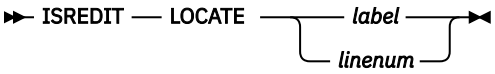
The LOCATE macro command scrolls up or down to a specified line. The line is then displayed as the first line on the panel. There are two forms of LOCATE, specific and generic.

The specific form of LOCATE positions a particular line at the top of the panel. You must specify either a line number or a label.

The generic LOCATE command positions the panel to the first, last, next, or previous occurrence of a particular kind of line.

Syntax

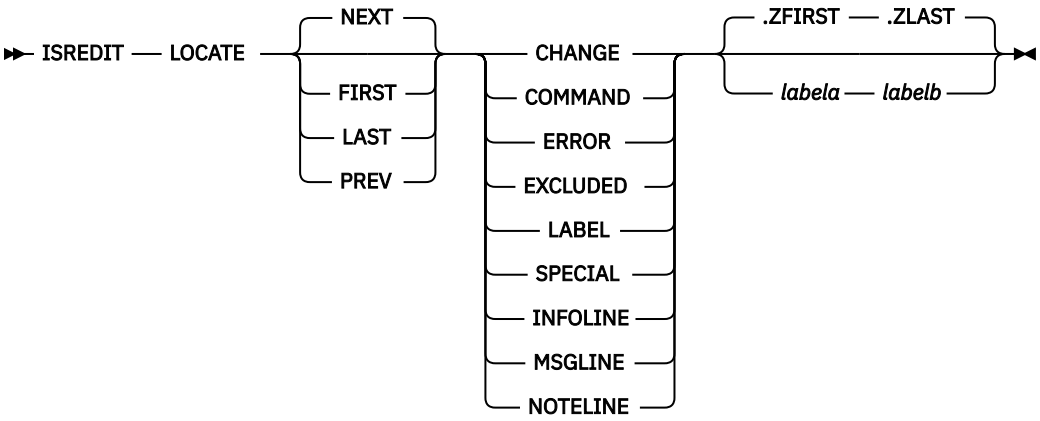
Specific LOCATE macro command syntax



linenum
A relative line number identifying the data line.

label
A label identifying the data line. It must be a label that you have previously defined or an editor-defined label, such as .ZFIRST or .ZLAST.

Generic LOCATE macro command syntax



FIRST
Searches from the first line, proceeding forward.

LAST
Searches from the last line, proceeding backward.

NEXT
Searches from the first line of the page displayed, proceeding forward.

PREV
Searches from the first line of the page displayed, proceeding backward.

CHANGE
Searches for a line with a change flag (==CHG>).

COMMAND
Searches for a line with a pending line command.

ERROR
Searches for a line with an error flag (==ERR>).

EXCLUDED
Searches for an excluded line.

LABEL
Searches for a line with a label.

SPECIAL
Searches for any special non-data (temporary) line:

- Bounds line flagged as =BNDS>
- Column identification lines flagged as =COLS>
- Information lines flagged as ======
- Mask lines flagged as =MASK>
- Message lines flagged as ==MSG>
- Note lines flagged as =NOTE=
- Profile lines flagged as =PROF>
- Tabs line flagged as =TABS>

INFOLINE
Searches for information lines flagged with ======

MSGLINE
Searches for message lines flagged with ==MSG>

NOTELINE
Searches for note lines flagged with =NOTE=

labela*, *labelb
Labels identifying the start and end of the group of lines in which to search.

Note: If you try to locate a line using a label that has not been assigned, you will receive a return code of 20. To avoid this, use the LINENUM assignment statement. When using the LINENUM statement, a return code of 8 is issued if the label does not exist.

```
ISREDIT (X) = LINENUM .LABEL
```

linenum1
Relative line number identifying the start of a group of lines in which to search.

linenum2
Relative line number identifying the end of a group of lines in which to search.

Return codes

0
Normal completion

4
Line not located

8
Empty member or data set

20
Severe error

Examples

To locate the next occurrence of a line with a label:

```
ISREDIT LOCATE NEXT LABEL
```

To locate the first occurrence of a special (non-data) line:

```
ISREDIT LOCATE FIRST SPECIAL
```

To locate the last excluded line:

```
ISREDIT LOCATE LAST X
```

To locate the previous line that contains an unprocessed line command:

```
ISREDIT LOCATE PREV CMD
```

To locate the first message line:

```
ISREDIT LOCATE FIRST MSGLINE
```

LRECL—Query the Logical Record Length

The LRECL assignment statement returns the maximum space, in bytes, available for data, COBOL number fields, and sequence number fields.

Syntax

varname
The name of a variable to contain the logical record length of the data being edited. The logical record length is a 3-digit value that is left-padded with zeros. If the variable is VDEFINEd in character format, it should be defined with a length of 5. The returned value is left padded with zeros. For compatibility with previous releases of ISPF/PDF, a length of 3 or 4 is allowed in cases where no data loss occurs.

Description

The value returned by the LRECL assignment statement includes the sequence number field and, for fixed-length records, the COBOL number field, if these number fields are used. For variable-length records, the value returned by LRECL does not include the 4-byte record descriptor word (RDW).

Use the DATA_WIDTH assignment statement to get the maximum space, in bytes, available for data.

Return codes

0
Normal completion

12
Invalid command format

20
Severe error

Examples

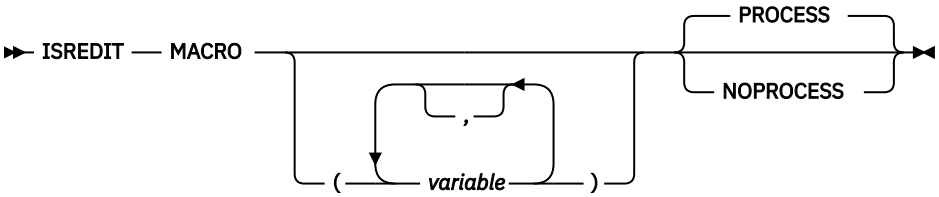
To check the logical record length of the data and process the data if the logical record length (LRECL) is 80:

```
ISREDIT (RECLEN) = LRECL
IF &RECLEN = 80 THEN -
   ...
```

MACRO—Identify an Edit Macro

The MACRO macro command identifies a command as a macro.

Syntax



variable
The names of the variables that contain parameters, if a macro allows parameters to be specified. Parameters are parsed and placed into the named variables in the order in which they are typed. The last variable contains any remaining parameters. Variables that do not receive a parameter are set to a null string. A parameter is a simple or quoted string, separated by blanks or commas. Quotes can be single (') or double ("), but must be matched at the beginning and end of the string.

PROCESS
Immediately processes all changes and line commands typed at the keyboard.

For edit line macros, see note under **NOPROCESS**.

NOPROCESS
Processes changes and line commands typed at the keyboard when the macro completes processing or a PROCESS statement is found. NOPROCESS must be used if the macro is to use line commands as input to its processing.

See “PROCESS—Process Line Commands” on page 383 for more information.

Note: For edit line macros, the NOPROCESS keyword must be used. The PROCESS macro statement is used within the macro to set the labels relating to the line command.

For more information, see “Working with an edit line command table” on page 84.

Description

The MACRO macro command is required in all macros. It must be the first command in a CLIST or REXX macro that is not a CLIST or REXX statement. Similarly, it also must be the first edit command in a program macro.

Return codes

0
Normal completion

8
No parameters are permitted for this processing

12
Syntax Error

20
Severe error

Examples

To begin a macro, first accepting a member name and optionally a line number range to be placed in the variable &PARM:

```
ISREDIT MACRO (PARM)
ISREDIT COPY AFTER .ZCSR &PARM
```

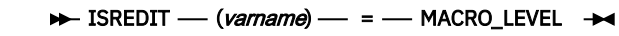
To begin a macro, checking parameters before processing panel information, testing for missing input, excess input, and nonnumeric input:

```
ISREDIT MACRO NOPROCESS (COL,X)
IF &STR(&COL) = &STR() THEN -
  ISREDIT (,COL) = DISPLAY_COLS
ELSE -
  IF &DATATYPE(&COL) = CHAR THEN -
    GOTO MSG
  IF &STR(&X) ¬= &STR() THEN -
    GOTO MSG
ISREDIT PROCESS
```

MACRO_LEVEL—Query the Macro Nesting Level

The MACRO_LEVEL assignment statement retrieves the current nesting level of the macro being run, and places the nesting level in a variable.

Syntax



varname
The name of a variable to contain the macro nesting level. The nesting level is a 3-digit value that is left-padded with zeros.

Description

The nesting level can be any number between 1 (a macro that you start) and 255. MACRO_LEVEL is used to adjust processing based on whether the macro is started by you or called by another macro. It is required if labels are to be set for the starter of this macro. See “LABEL—Set or Query a Line Label” on page 352 for more information.

Return codes

0
Normal completion

12
Invalid command format

20
Severe error

Examples

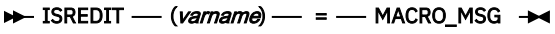
To set the label for the caller of the macro at 1 less than the current level:

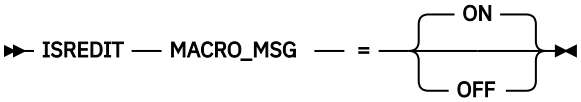
```
ISREDIT (NESTLEV) = MACRO_LEVEL
ISREDIT LABEL .ZCSR = .XSTR &EVAL(&NESTLEV -1)
```

MACRO_MSG—Set or Query the Macro Message switch

The MACRO_MSG assignment statement sets or retrieves the value of the macro_msg switch, which controls whether macro processing delivers ISPF messages to the macro.

Syntax





varname
The name of a variable containing the setting of MACRO_MSG.

ON
ISPF messages generated by macro commands are formatted.

OFF
ISPF messages are not formatted.

Description

The MACRO_MSG assignment statement sets a switch for subsequent macro processing. When set ON, any message that is generated by a macro command is formatted and made available in variables in ZEDILMSG, ZEDISMSG, and ZEDMSGNO.

This is a diagnostic switch and should only be used to extract messages as required. Macros that perform operations on many edit lines may experience a performance degradation if this switch is ON.

Return codes

0
Normal completion

20
Severe error

Examples

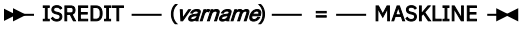
To set macro_MSG:

```
ISREDIT MACRO_MSG = ON
```

MASKLINE—Set or Query the Mask Line

The MASKLINE assignment statement sets or retrieves the value of the mask line, which controls the display formatting of your input.

Syntax



varname
The name of a variable containing maskline contents.

data
Specifies that these forms can be used:

- Simple string
- Delimited string
- Variable
- Template (< *col*,*string* >)
- Merge format (*string1* + *string2*, operand + *string2*, *string1* + operand)
- Operand:

  LINE *linenum*
  Data from the line with the given relative line number.

  LINE *label*
  Data from the line with the given label.

  MASKLINE
  Data from the mask line.

  TABSLINE
  Data from the tabs line.

Description

The MASKLINE assignment statement places the mask line contents in a variable or sets the mask line from a variable. The mask line can contain any characters and serves to initialize inserted lines to the value of the mask line. See the description of templates in "Overlays and templates" on page 97 for more information on the setting of a mask line.

Be careful not to destroy a DBCS string in the mask line. If shift-out (SO) or shift-in (SI) characters in a mask line are overlaid through the MASKLINE statement, the result is unpredictable.

Return codes

0
Normal completion

4
Data truncated

16
Variable data truncated

20
Severe error

Examples

To set the mask line to place comment delimiters starting at lines 40 and 70:

```
ISREDIT MASKLINE = <40 '&STR(/*)' 70 '&STR(/*)'>
```

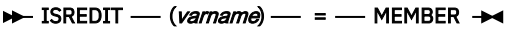
To set the mask line to blanks:

```
ISREDIT MASKLINE = " "
```

MEMBER—Query the Current Member Name

The MEMBER assignment statement retrieves the name of the library member currently being edited, and places it in a variable. If a sequential data set is being edited, the variable is set to blanks.

Syntax



varname
The name of a variable to contain the name of the library member currently being edited.

Return codes

0
Normal completion

12
Invalid command format

20
Severe error

Examples

To determine if you are editing a library member with a prefix of MIN:

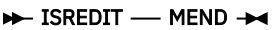
```
ISREDIT (MEMNAME) = MEMBER
IF &SUBSTR(1:3,&MEMNAME  ) = MIN THEN -
   ...
```

MEND—End a Macro in the Batch Environment

Note: The MEND command is obsolete.

The MEND macro command ends a macro that is running in the batch environment. It was required for CLISTs that ran in the batch environment using the MVS/370 operating system. It is not required for z/OS, but can be used.

Syntax



Return codes

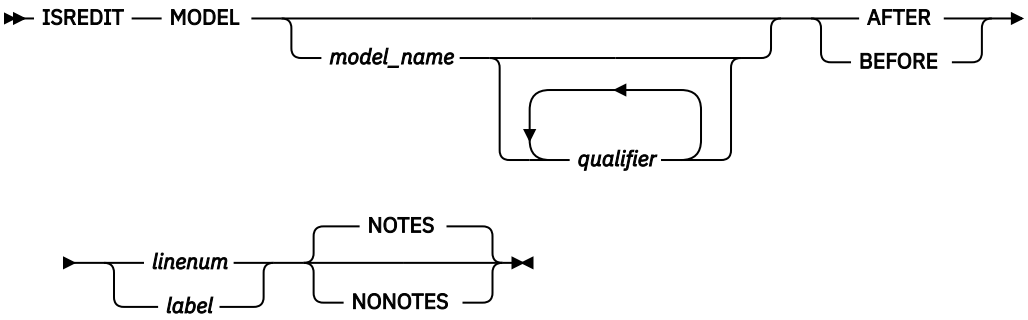
0
Normal completion

MODEL—Copy a Model into the Current Data Set

The model name form of the MODEL macro command copies a specified dialog development model before or after a specified line.

The class name form of the MODEL macro command changes the model class that the editor uses to determine the model you want. For more information on edit models, see Chapter 4, "Using edit models," on page 69.

Syntax



model_name
The name of the model to be copied, such as VGET for the VGET service model. This operand can also be one of the options listed on a model selection panel, such as V1 for the VGET service model. However, to use these options with the MODEL macro command, you must already know what they are or else display a model selection panel by using the MODEL primary command. The MODEL macro command does not display model selection panels. See *z/OS ISPF Planning and Customizing* for a list of models and model names.

qualifier
The name of a model on a secondary model selection panel, such as TBCREATE for the TBCREATE service model. This operand can also be one of the options listed on a model selection panel, such as G1 for the TBCREATE service model.

For example, a model selection panel allows you to enter T1 to choose table models. It then displays another model selection panel for choosing table models, such as G1 for the TBCREATE service model. Therefore, your MODEL macro command could use either TABLES or T1 as the model-name operand and either TBCREATE or G1 as the qualifier operand. The simplest way would be to use TBCREATE or G1 as the model-name operand and omit the qualifier operand.

To use options with the MODEL macro command, you must already know what they are or else display a model selection panel by using the MODEL primary command. The MODEL macro command does not display model selection panels. See *z/OS ISPF Planning and Customizing* for a list of models and model names.

AFTER
Specifies that the model is to be copied after the line specified by *linenum* or *label*.

BEFORE
Specifies that the model is to be copied before the line specified by *linenum* or *label*.

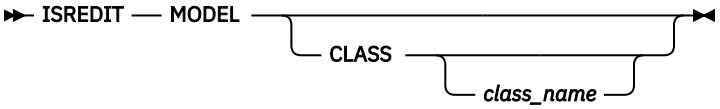
linenum
A relative line number identifying where the model should be copied.

label
A label identifying where the model should be copied.

NOTES
Explanatory notes appear when a model is copied.

NONOTES
No explanatory notes appear.

Macro command class name syntax



CLASS
Specifies that the current model class is to be replaced by class-name. The new class name is used for all models from that point on, until you change the model class again or end the edit session.

class_name
Specifies the model class for the current edit session. It must be a name on the Model Classes panel or an allowable abbreviation. The model class coincides with the type of model, such as REXX, COBOL, or FORTRAN.

Return codes

0
Normal completion

4
Data truncated (the model exceeded the right-hand margin of the data being edited)

12
Invalid line number (*linenum*) or label (*label*)

20
Severe error

Examples

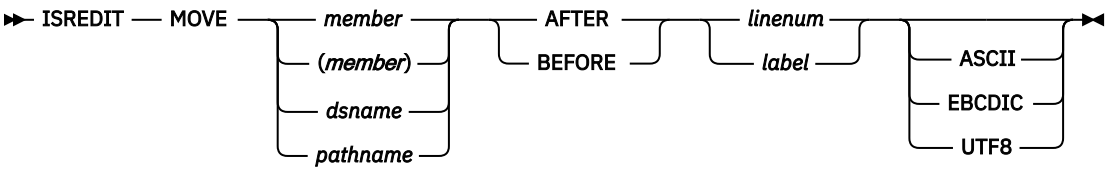
To copy the VGET model at the end of the current data:

```
ISREDIT MODEL VGET AFTER .ZL
```

MOVE— Move a Data Set or a Data Set Member

The MOVE macro command moves a sequential data set, member of a partitioned data set, or z/OS UNIX file into the data you are editing.

Syntax



member
A member of the ISPF library or partitioned data set you are editing.

dsname
A partially or fully qualified data set name. If the data set is partitioned you must include a member name in parentheses.

pathname
The pathname for a z/OS UNIX regular file to be moved. (Also, see specunix.dita#specunix.)

AFTER
Specifies that the member is to be moved after the target specified by *linenum* or *label*.

BEFORE
Specifies that the member is to be moved before the target specified by the label.

linenum
A relative line number identifying the target of the move.

label
A label identifying the target of the move. It can be either a label that you define, or one of the editor-defined labels, such as .ZF and .ZL.

ASCII, EBCDIC, UTF8
When one of these keywords is supplied, if the data is using a different character set to that designated by the keyword, the data being moved in from the external file is converted from the character set designated by the keyword to the character set specified for the file being edited or to the terminal character set.

Note: If *member* or *dsname* is less than 8 characters and the data set you are editing is partitioned, a like-named member is copied. If a like-named member does not exist, the name is considered to be a partially qualified data set name.

Description

The member, data set, or z/OS UNIX file is deleted after the move. For a concatenated sequence of ISPF libraries, the deletion occurs only if the member was in the first library of the concatenation sequence.

See camd.dita#camd if you need more information.

Return codes

0
Normal completion

8
End of data before last record read or the specified data set is in use

12
Invalid line pointer (*linenum* or *label*); member not found or BLDL error

16
End of data before first record read

20
Syntax error (invalid name, incomplete range), or I/O error

Examples

To move the contents of member ABC after the first line in the current data:

```
ISREDIT MOVE ABC AFTER .ZF
```

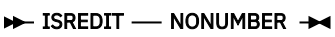
To move all of data set MOVECOPY.DATA before the line where the cursor is currently positioned:

```
ISREDIT MOVE MOVECOPY.DATA BEFORE .ZCSR
```

NONUMBER—Turn Off Number Mode

The NONUMBER macro command turns off number mode, which controls the numbering of lines in the current data.

Syntax



Description

You can also use the NUMBER OFF macro command to turn off number mode.

When number mode is off, NONUMBER prevents any verification of valid line numbers, generation of sequence numbers, and the renumbering of lines that normally occurs when autonum mode is on.

Return codes

0
Normal completion

20
Severe error

Examples

To turn number mode off by using the NONUMBER command:

```
ISREDIT NONUMBER
```

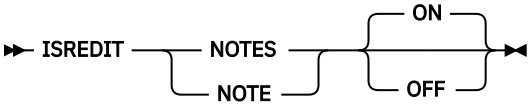
NOTES—Set or Query Note Mode

The NOTES macro command sets note mode, which controls whether notes are to appear when a dialog development model is inserted into the data.

The NOTES assignment statement either sets note mode, or retrieves the setting of note mode and places it in a variable.

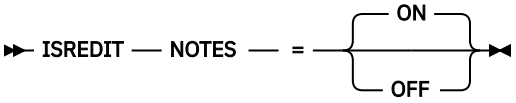
See “MODEL—Copy a Model into the Current Data Set” on page 251 for information about copying dialog development models.

Syntax



ON
Displays explanatory notes when a model is copied into the data being edited.

OFF
Does not display explanatory notes.



varname
The name of a variable to contain the value of note mode, either ON or OFF.

ON
Same as macro command syntax.

OFF
Same as macro command syntax.

Return codes

0
Normal completion

20
Severe error

Examples

To set note mode off:

```
ISREDIT NOTES = OFF
```

To store the value of note mode in variable &NOTEMODE:

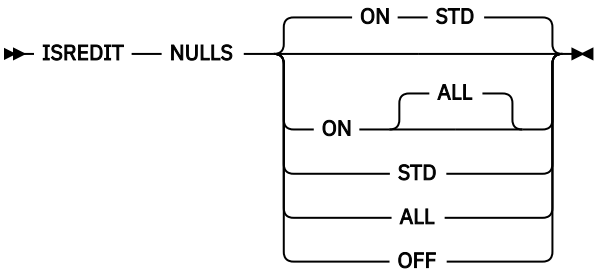
```
ISREDIT (NOTEMODE) = NOTES
```

NULLS—Set or Query Nulls Mode

The NULLS macro command sets nulls mode, which determines whether trailing blanks in each data field are written to the panel as blanks or nulls.

The NULLS assignment statement either sets nulls mode or retrieves the setting of nulls mode and places it in a variable.

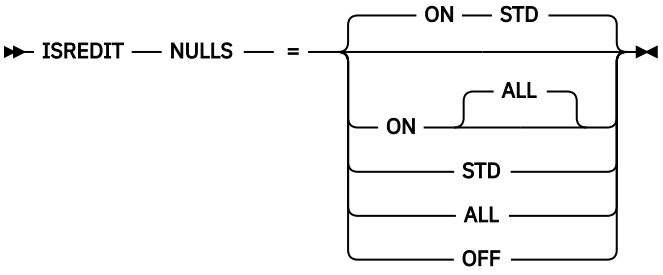
Syntax



ON STD
Specifies that in fields that contain any blank trailing space, the space is to be written as one blank followed by nulls. If the field is entirely empty, it is written as all blanks.

ON ALL
Specifies that all trailing blanks and all-blank fields are written as nulls.

OFF
Specifies that trailing blanks in each data field are written as blanks.



var1
The name of a variable to contain either ON or OFF.

var2
The name of a variable to contain ALL, STD, or blanks.

ON STD
Same as macro command syntax.

ON ALL
Same as macro command syntax.

OFF
Same as macro command syntax.

Description

The term *data field* normally refers to the 72 characters of data on each line. Using hardware tabs, however, you can split each line into multiple fields. See "TABS—Define Tabs" on page 286 for more details.

Blank characters (X'40') and null characters (X'00') both appear as blanks. When you use the I (insert) line command, the data entry area appears as blanks for NULLS ON STD and as nulls for NULLS ON ALL.

Trailing nulls simplify use of the Ins (insert) key on the IBM 3270 keyboard. You can use this key to insert characters on a line if the line contains trailing nulls.

Besides using NULLS, you can create nulls at the end of a line by using the Erase EOF or Del (delete) key. Null characters are never stored in the data; they are always converted to blanks.

Return codes

0
Normal completion

20
Severe error

Examples

To set nulls mode on with blank trailing space written as one blank followed by nulls and empty fields written as all blanks:

```
ISREDIT NULLS = ON STD
```

To set nulls mode off and thus have trailing blanks in each data field:

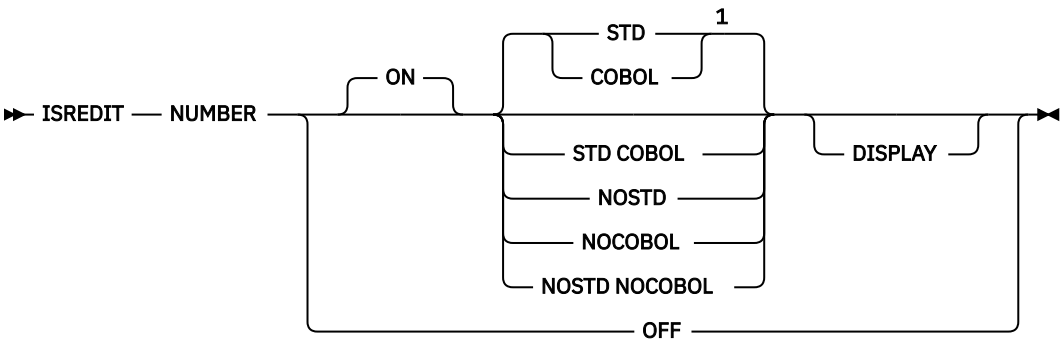
```
ISREDIT NULLS = OFF
```

NUMBER—Set or Query Number Mode

The NUMBER macro command sets number mode, which controls the numbering of lines in the current data.

The NUMBER assignment statement either sets number mode, or retrieves the setting of number mode and places it in variables.

Syntax



Notes:

[1] STD is the default for non-COBOL data set types. COBOL is the default for COBOL data set types.

ON
Automatically verifies that all lines have valid numbers in ascending sequence and renumbers any lines that are either unnumbered or out of sequence. You can also use the RENUM command to turn number mode on and renumber lines.

The editor interprets the STD, COBOL, and DISPLAY operands only when number mode is turned on.

OFF
Turns number mode off. You can also use the NONUMBER command to turn number mode off.

STD
Numbers the data in the standard sequence field.

COBOL
Numbers the data in the COBOL field.

Note: The NUMBER ON COBOL mode is not supported for formatted data sets.

Attention: If number mode is off, make sure the first 6 columns of your data set are blank before using either the NUMBER ON COBOL or NUMBER ON STD COBOL command. Otherwise, the data in these columns is replaced by the COBOL sequence numbers. If that happens and if edit recovery or SETUNDO is on, you can use the UNDO command to recover the data. You can also use CANCEL at any time to end the edit session without saving the data.

STD COBOL
Numbers the data in both fields.

If both STD and COBOL numbers are generated, the STD number is determined and then used as the COBOL number. The COBOL numbers can be out of sequence if the COBOL and STD fields were not synchronized. Use RENUM to force synchronization.

NOSTD
Turns standard number mode off.

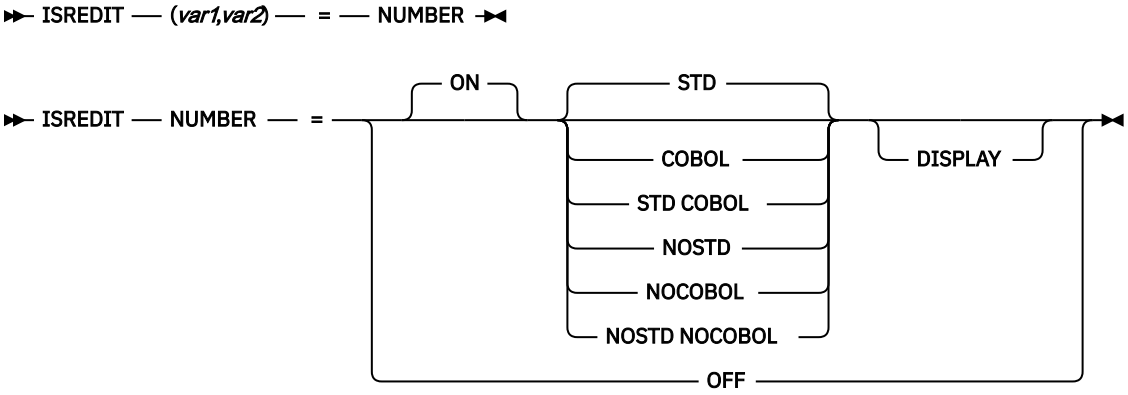
NOCOBOL
Turns COBOL number mode off.

NOSTD NOCOBOL
Turns both the standard number mode and COBOL number mode off.

DISPLAY
Causes the width of the data window to include the sequence number fields. Otherwise, the width of the window does not include the sequence number fields. When you display a data set with a logical

record length of 80 and STD numbering, the sequence numbers are not shown unless you are using a 3278 Model 5 terminal, which displays 132 characters. Automatic left or right scrolling is performed, if required, so that the leftmost column of the data window is the first column displayed.

Assignment statement syntax



var1
The name of a variable to contain either ON or OFF.

var2
The name of a variable to contain one of the eight combinations in this list:

```
NOSTD   NOCOBOL   DISPLAY
STD     NOCOBOL   DISPLAY
NOSTD   COBOL     DISPLAY
STD     COBOL     DISPLAY

NOSTD   NOCOBOL   NODISPL
STD     NOCOBOL   NODISPL
NOSTD   COBOL     NODISPL
STD     COBOL     NODISPL
```

The value STD, COBOL, or DISPLAY can be placed in *var2*, even when *var1* is set to off. This allows the macro to save and restore number mode. It also allows the macro to set number mode off, while specifying defaults to be used when number mode is changed to on.

ON
Same as for macro command syntax.

OFF
Same as for macro command syntax.

STD
Same as for macro command syntax.

COBOL
Same as for macro command syntax.

NOSTD
Turns standard number mode off.

NOCOBOL
Turns COBOL number mode off.

NOSTD NOCOBOL
Turns both the standard number mode and COBOL number mode off.

STD COBOL
Same as for macro command syntax.

DISPLAY
Same as for macro command syntax.

Description

When number mode is on, NUMBER verifies that all lines have valid numbers in ascending sequence. It renumbers any lines that are either unnumbered or out of sequence, but it does not otherwise change existing numbers.

In number mode, the editor automatically generates sequence numbers in the data for new lines that are created when data is copied or inserted. The editor also automatically renumbers the data when it is saved if autonum mode is in effect.

If the number overlays the shift-in (SI) or shift-out (SO) characters, the double-byte characters are displayed incorrectly and results are unpredictable.

Return codes

0
Normal completion

20
Severe error

Examples

To save the current value of number mode, set number mode off for processing, and then restore the value of number mode:

```
ISREDIT (STAT,VALUE) = NUMBER
ISREDIT NUMBER OFF
   ...
ISREDIT NUMBER = (STAT VALUE)
```

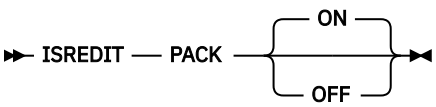
PACK—Set or Query Pack Mode

The PACK macro command sets pack mode, which controls whether the data is stored in packed format.

The PACK assignment statement either sets pack mode, or retrieves the setting of pack mode and places it in a variable.

The PACK command saves the pack mode setting in the edit profile. See “Packing data” on page 16 for more information about packing data.

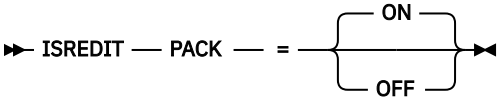
Syntax



ON
Saves data in packed format.

OFF
Saves data in unpacked (standard) format.

If you change pack mode, data is written when an END command is issued.



varname
The name of a variable to contain the setting of pack mode, either ON or OFF.

ON
Same as macro command syntax.

OFF
Same as macro command syntax.

Return codes

0
Normal completion

20
Severe error

Examples

To set pack mode off:

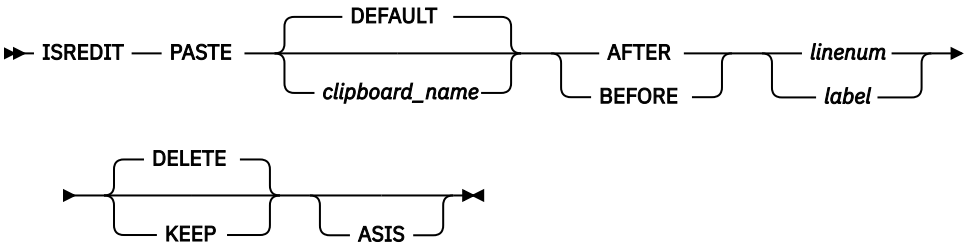
```
ISREDIT PACK OFF
```

PASTE—Move or Copy Lines from Clipboard

The PASTE macro command moves or copies lines from a clipboard into an edit session.

Syntax

Macro command syntax



clipboardname
The name of the clipboard to use. If you omit this parameter, the ISPF default clipboard (named DEFAULT) is used. You can define up to ten additional clipboards. The size of the clipboards and number of clipboards might be limited by installation defaults.

BEFORE
The destination of the data that is being transferred from the clipboard. BEFORE copies the data *before* the specified label *linenum* or *label*.

AFTER
The destination of the data that is being transferred from the clipboard. AFTER copies the data *after* the specified label *linenum* or *label*.

linenum
A relative line number identifying the line after, or before, which the lines from the clipboard are copied or moved.

label
A label identifying the line after, or before, which the lines from the clipboard are copied or moved.

KEEP
Records are copied and not removed from the clipboard.

DELETE
Records are moved and deleted from the clipboard.

ASIS
The PASTE command determines the character set of the data in the clipboard. If this is different to the character set being used for the file being edited an automatic conversion occurs for the data being pasted into the file.

If ASIS is specified, then the automatic conversion does not take place.

Description

PASTE copies or moves lines from a specified clipboard to the current edit session. If lines in the clipboard are longer than the lines in the edit session, they are truncated.

The portion of the line that is saved in the clipboard is only the data portion of the line. Line numbers are *not* saved. If the data was CUT from a data set that had sequence numbers and is PASTEd into an edit session without sequence numbers, or if it was CUT from a data set without sequence numbers and PASTEd into a session with sequence numbers, some shifting of data is likely to occur.

Return codes

0
Normal completion

12
Parameter error. Clipboard is empty or does not exist.

20
Severe error

Examples

To paste data from the default clipboard to the line after the last line in the edit session:

```
ISREDIT PASTE AFTER .ZLAST DELETE
```

To paste data from the default clipboard to the line after the first line in the edit session, without clearing the contents of the clipboard:

```
ISREDIT PASTE AFTER .ZFIRST KEEP
```

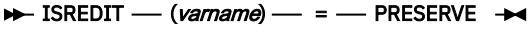
PRESERVE—Enable Saving of Trailing Blanks

The PRESERVE macro command enables or disables the saving of trailing blanks in the editor. This enables you to override the setting for the field on the edit entry panel called "Preserve VB record length".

Syntax

ON
The editor saves all trailing blanks in the record.

OFF
Turns truncation on. ISPF removes trailing blanks when saving variable-length files. If a line is empty ISPF saves 1 blank.



varname
The name of a variable to contain the setting of PRESERVE mode, either ON or OFF.

ON
Same as macro command syntax.

OFF
Same as macro command syntax.

Description

PRESERVE ON causes the editor to save trailing blanks for variable length files. The number of blanks saved for a particular record is determined by one of these:

- The original record length of the record when it was read in to the editor
- The number of blanks required to pad the record length specified by the SAVE_LENGTH edit macro command
- The length of the record that was saved on disk during a previous SAVE request in the same edit session

PRESERVE OFF causes the editor to truncate trailing blanks. If a line is empty ISPF saves 1 blank.

Use of the PRESERVE command does not prevent the editor from working on data past the specified record length. The length set and returned by the PRESERVE command is only used when the data is written and does not affect the operation of other edit functions.

Return codes

0
Normal completion

6
Record format is not variable.

16
Error setting variable.

20
Severe error

Examples

To save the value of the PRESERVE mode in variable &TRMODE:

```
ISREDIT (TRMODE) = PRESERVE
```

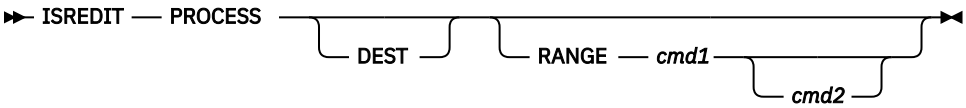
To enable the editor to remove trailing blanks when the data is saved:

```
ISREDIT PRESERVE OFF
```

PROCESS—Process Line Commands

The PROCESS macro command allows the macro to control when line commands or data changes typed at the keyboard are processed.

Syntax



DEST
Specifies that the macro can capture an A (after) or a B (before) line command that you enter. The .ZDEST label is set to the line preceding the insertion point. If A or B is not entered, .ZDEST points to the last line in the data.

Note: If the PROCESS macro command is issued within a line macro, see separate note.

RANGE
Must be followed by the names of one or two line commands, either of which you can enter. Use the RANGE_CMD assignment statement to return the value of the line command entered. This allows the macro to define and then capture a line command that you enter. It can also modify its processing based on which of the two commands was entered.

Note: If the PROCESS macro command is issued within a line macro, see separate note.

cmd1* and *cmd2
Specifies one or two line command names, which can be 1 to 6 characters; however, if the name is 6 characters long it cannot be used as a block format command (to specify multiple lines) by doubling the last character. The name can contain any alphabetic or special character except blank, hyphen (-), or apostrophe ('). It cannot contain any numeric characters.

The .ZFRANGE label is set to the first line identified by the line command that you have entered, and .ZLRANGE is set to the last line. They can refer to the same line. If the expected RANGE line command was not entered, .ZFRANGE points to the first line in the data and .ZLRANGE points to the last line in the data.

Note:
Sequence of processing when PROCESS command issued within a line macro

If the PROCESS command is issued within a line macro, it sets the DEST and RANGE labels, but does not influence the normal processing order of line commands. Line commands that appear before the user line command will have already been executed, and line commands that occur after the user line command are not executed until the user line command macro has completed.

For more information, see “Working with an edit line command table” on page 84.

Description

If a line is retrieved before the PROCESS macro command is called, changes made to this line will not be seen. The DEST and RANGE operands allow the macro to identify the line commands that you can enter as additional input to the macro.

This command cannot be specified without first coding the MACRO command with a NOPROCESS operand.

For more information about using the PROCESS command, see "Using the PROCESS command and operand" on page 107.

Return codes

0
Normal completion.

4
A RANGE was expected by the macro, but one was not specified; default values set.

8
A DEST (destination) was expected by the macro, but one was not specified; default values set.

12
Both a RANGE and a DEST (destination) were expected by the macro, but were not specified; default values set.

16
You entered incomplete or conflicting line commands.

20
Severe error

Note: ISPF does not consider a return code of 12 from the PROCESS edit macro command an error and does not terminate a macro that receives a return code of 12 from the PROCESS edit macro.

Examples

To set up the macro to process the line commands * and # (defined by the macro writer):

```
ISREDIT MACRO NOPROCESS
ISPEXEC CONTROL ERRORS RETURN
ISREDIT PROCESS RANGE * #
IF &LASTCC >= 16 THEN EXIT CODE(&LASTCC)
ISREDIT (CMD) = RANGE_CMD
ISREDIT (FIRST) = LINENUM .ZFRANGE
ISREDIT (LAST)  = LINENUM .ZLRANGE
IF &STR(&CMD) = &STR(*) THEN -
   ...
```

To place data depending on the location of the A (after) or B (before) line command:

```
ISREDIT MACRO NOPROCESS
ISREDIT PROCESS DEST
ISREDIT LINE_AFTER .ZDEST = "&DATA"
```

To allow processing of the A and B destination line commands and the specification of a range by using the * line command (defined by the macro writer):

```
ISREDIT MACRO NOPROCESS
ISREDIT PROCESS DEST RANGE *
```

See "Using the PROCESS command and operand" on page 107.

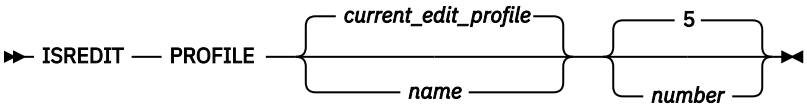
PROFILE—Set or Query the Current Profile

The control form of the PROFILE macro command displays your current edit profile, defines a new edit profile, or switches to a different edit profile.

The lock form of the PROFILE macro command locks or unlocks the current edit profile.

The PROFILE assignment statement retrieves the name and lock status of the current edit profile and stores those values in variables.

Syntax



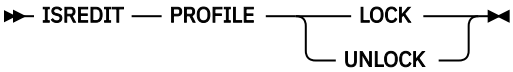
name
The profile name. It can consist of up to 8 alphanumeric characters, the first of which must be alphabetic. The edit profile table is searched for an existing entry with the same name. That profile is then read and used. If one is not found, a new entry is created in the profile table.

If you omit this operand, the current edit profile is used.

number
The number of lines, from 0 through 8, of profile data to be displayed. When you type 0 as the number, no profile data is displayed. When you omit the number operand, the profile modes appear; the =MASK> and =TABS> lines are displayed if they contain data, followed by the =COLS> line.

The =BNDS> line does not appear if it contains the default boundary positions. It does appear when the bounds are set to something other than the default, and no 'number' parameter is entered into the PROFILE command.

For more information about displaying and defining a profile, see "Displaying or defining an edit profile" on page 17.



LOCK
Specifies that the current values in the profile are saved in the edit profile table and are not modified until the profile is unlocked. The current copy of the profile can be changed, either because of commands you enter that modify profile values (BOUNDS and NUMBER, for example) or because of differences in the data from the current profile settings. However, unless you unlock the edit profile, the saved values replace the changes when you end the edit session.

Caps, number, stats, and pack mode are automatically changed to fit the data. These changes occur when the data is first read or when data is copied into the data set. Message lines (==MSG>) are inserted in the data set to show you which changes occurred.

Note: To force caps, number, stats, or pack mode to a particular setting, use an initial macro. Be aware, however, that if you set number mode on, data may be overlaid.

UNLOCK
Specifies that the editor saves changes to profile values.

See "Locking an edit profile" on page 19 for more information about locking and unlocking the profile.

RESET
Specifies that the ZEDFAULT profile is to be removed and the site-wide configuration for new edit profiles is to be used.

See "Locking an edit profile" on page 19 for more information about locking and unlocking the profile.

var1
The name of a variable to contain the name of the current edit profile.

var2
The name of a variable to contain the profile status, LOCK or UNLOCK.

Description

Profile names cannot be set by an assignment statement. Instead, use PROFILE to change a profile name, thereby changing the current edit profile and the edit profile values.

Return codes

0
Normal completion

20
Severe error

Examples

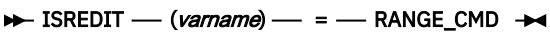
To check the lock status of the profile and perform processing if the profile is locked:

```
ISREDIT (,STATUS) = PROFILE
IF &STATUS = LOCK THEN -
     ...
```

RANGE_CMD—Query a Command That You Entered

The RANGE_CMD assignment statement identifies the name of a line command entered from the keyboard and processed by a macro.

Syntax



varname
The name of a variable to contain the line command that you entered.

Description

The macro must first issue a PROCESS command to identify all line commands to be processed by this macro. A particular line command within a range can be found by using the RANGE_CMD. For instance, if this PROCESS command is issued by a macro:

```
PROCESS RANGE Q $
```

The RANGE_CMD statement returns either a Q or a $. If a range such as Q5 is entered, only Q is returned.

Return codes

0
Normal completion

4
Line command not set

8
Line command setting not acceptable

20
Severe error

Examples

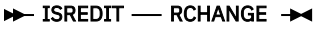
To determine which line command (* or #) you entered and to process the line command (defined by the macro writer):

```
ISREDIT MACRO NOPROCESS
ISREDIT PROCESS RANGE * #
ISREDIT (CMD) = RANGE_CMD
IF &STR(&CMD) = &STR(*) THEN -
        ...
ELSE IF &STR(&CMD) = &STR(#) THEN -
        ...
```

RCHANGE—Repeat a Change

The RCHANGE command repeats the change requested by the most recent CHANGE command.

Syntax



Description

You can use this command to repeatedly change other occurrences of the search string. After a *string* NOT FOUND message appears, the next RCHANGE issued starts at the first line of the current range for a forward search (FIRST or NEXT specified) or the last line of the current range for a backward search (LAST or PREV specified).

Return codes

0
Normal completion

4
String not found

8
Change error (*string2* longer than *string1* and substitution was not performed on at least one change)

12
Syntax error

20
Severe error

Examples

To perform a single-line change and then repeat the change from the top if the string was not found:

```
ISREDIT CHANGE C'.  the' C'.  The' 1 8
IF &LASTCC = 4 THEN—
  ISREDIT RCHANGE
```

RECFM—Query the Record Format

The RECFM assignment statement retrieves the record format of the data set being edited, and places the value in a variable.

Syntax

var1
The name of a variable to contain the type of record format of the data being edited, either F or V:

F
Fixed-length records.

V
Variable-length records.

var2
The name of a variable to contain the remaining record format information of the data being edited, in the combination of M, A, S, BM, BA, BS, BSM, or BSA:

B
Blocked records.

S
Standard or spanned records.

M
Machine print control character records.

A
ASA print control character records.

When editing a z/OS UNIX file, *var2* is set to blanks.

Return codes

0
Normal completion

20
Severe error

Examples

To place the type of record format in variable RECFM1 and then use either the logical data width (for a fixed data set) or the right display column (for a variable data set):

```
ISREDIT (RECFM1) = RECFM
IF &RECFM1 = F THEN -
  ISREDIT (WIDTH) = DATA_WIDTH
ELSE -
  ISREDIT (,WIDTH) = DISPLAY_COLS
```

To place the remaining record format information in variable RECFM2:

```
ISREDIT (,RECFM2) = RECFM
```

To place the type of record format information in variable RECFM1, and the remaining record format information in variable RECFM2:

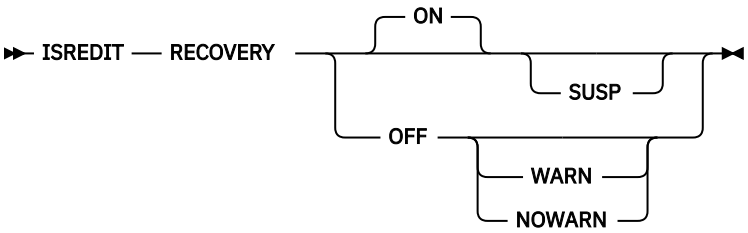
```
ISREDIT (RECFM1,RECFM2) = RECFM
```

RECOVERY—Set or Query Recovery Mode

The RECOVERY macro command sets edit recovery mode, which allows you to recover data after a system failure or power outage.

The RECOVERY assignment statement either sets edit recovery mode, or retrieves the edit recovery mode setting and places it in a variable.

Syntax



ON
The system creates and updates a recovery data set for each change thereafter.

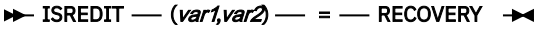
OFF
The system does not create and update a recovery set.

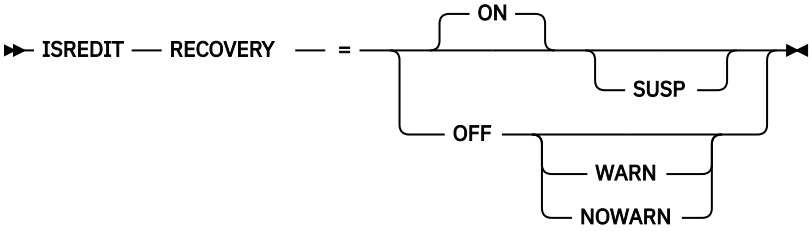
WARN
This operand no longer has a practical function, due to a software change. However, the primary command continues to accept the operand for compatibility reasons.

NOWARN
This operand no longer has a practical function, due to a software change. However, the primary command continues to accept the operand for compatibility reasons.

SUSP
This operand, when specified with the ON operand has no function. It allows existing macros which save and restore the recovery state to continue working. When SUSP is specified by itself, it functions like the ON operand.

See “Edit recovery” on page 38 for more information about edit recovery.





var1
The name of a variable to contain the setting of recovery mode, either ON or OFF.

var2
The name of a variable that contains the warning setting, either WARN, NOWARN (when RECOVERY is OFF), or blank or SUSP (when RECOVERY is ON).

ON
The system creates and updates a recovery data set for each change thereafter.

OFF
The system does not create and update a recovery set.

WARN
This operand no longer has a practical function, due to a software change. However, the primary command continues to accept the operand for compatibility reasons.

NOWARN
This operand no longer has a practical function, due to a software change. However, the primary command continues to accept the operand for compatibility reasons.

SUSP
This value indicates that recovery is ON, but that it is suspended due to a previous error.

Return codes

0
Normal completion

20
Severe error

Examples

To save the value of recovery mode in variable &RECOV:

```
ISREDIT (RECOV) = RECOVERY
```

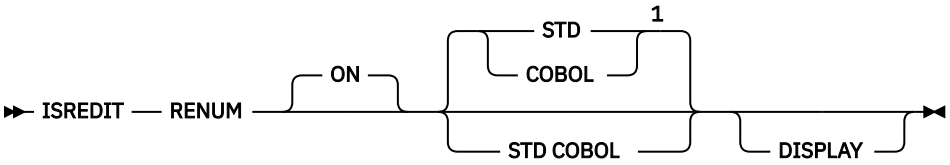
To set recovery mode OFF:

```
ISREDIT RECOVERY = OFF
```

RENUM—Renumber Data Set Lines

The RENUM macro command immediately turns on number mode and renumbers all lines, starting with number 100 and incrementing by 100. For any members exceeding 10 000 lines, the increment would be less than 100.

Syntax



Macro command syntax
Notes:

[1] STD is the default for non-COBOL data set types. COBOL is the default for COBOL data set types.

ON
Automatically verifies that all lines have valid numbers in ascending sequence and renumbers any lines that are either unnumbered or out of sequence. It also turns number mode on and renumbers lines.

The STD, COBOL, and DISPLAY operands are interpreted only when number mode is turned on.

STD
Numbers the data in the standard sequence field.

COBOL
Numbers the data in the COBOL field.

STD COBOL
Numbers the data in both fields.

If both STD and COBOL numbers are being generated, the STD number is determined and then used as the COBOL number. This can result in COBOL numbers that are out of sequence if the COBOL and STD fields were not synchronized. Use RENUM to force synchronization.

DISPLAY
Causes the width of the data window to include the sequence number fields. Otherwise, the width of the window does not include the sequence number fields. When you display a data set with a logical record length of 80 and STD numbering, the sequence numbers are not shown unless you are using a 3278 Model 5 terminal, which displays 132 characters. The editor automatically scrolls left or right, if required, so that the leftmost column of the data window is the first column displayed.

Return codes

0
Normal completion

20
Severe error

Examples

To renumber all data lines with standard numbering:

```
ISREDIT RENUM
```

To renumber all data lines with standard and COBOL numbering:

```
ISREDIT RENUM STD COBOL
```

To renumber all data lines with COBOL numbering, bringing the sequence numbers within the data window:

```
ISREDIT RENUM COBOL DISPLAY
```

To turn sequence numbers off:

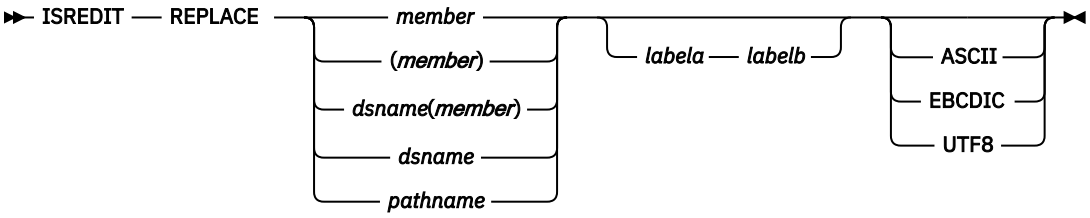
```
ISREDIT RENUM OFF
```

REPLACE—Replace a Data Set or Data Set Member

The REPLACE macro command adds or replaces data in a member of the partitioned data set that you are editing, in a member of another partitioned data set, in a sequential data set, or in a z/OS UNIX file. If a member you want to replace exists and the member is in a PDSE version 2 data set that is configured for member generations, the editor creates a new generation of the member. This new generation becomes the current generation (also known as generation zero).

Syntax

Macro command syntax



member
The name of the member to be replaced in the partitioned data set currently being edited. If a name of eight or fewer characters is specified and it could be a member name or a data set name, REPLACE searches for a member name first. If no member name is found, then the name is used as a data set. If the member does not exist, the editor creates it. If you are using a concatenated sequence of libraries, the member is always written to the first library in the sequence.

dsname
The name of a sequential data set that is to be replaced. The data set name can be fully or partially qualified.

***dsname*(*member*)**
The name of a different partitioned data set and member name to be replaced in the partitioned data set. The data set name can be fully or partially qualified.

pathname
The pathname for a z/OS UNIX regular file to be replaced. If the file does not exist, it is created. (Also, see specunix.dita#specunix.)

linenum1
Relative line number identifying the start of a group of lines in the current member that replace data in the other member.

linenum2
Relative line number identifying the end of a group of lines in the current member that replace data in the other member.

labela*, *labelb
Labels identifying the start and end of the group of lines in the current member that replace data in the other member.

For more information about using labels to identify a group of lines, see linergn.dita#linergn.

ASCII, EBCDIC, UTF8
When one of these keywords is supplied, if the data is using a different character set to that designated by the keyword, the data being replaced in the external file is converted to the character set designated by the keyword.

Return codes

0
Normal completion

8
Member in use

12
Invalid line pointer

20
Syntax error (invalid name, incomplete line pointer value), or I/O error

Examples

To replace member MEM1 with the first 10 lines of the current data:

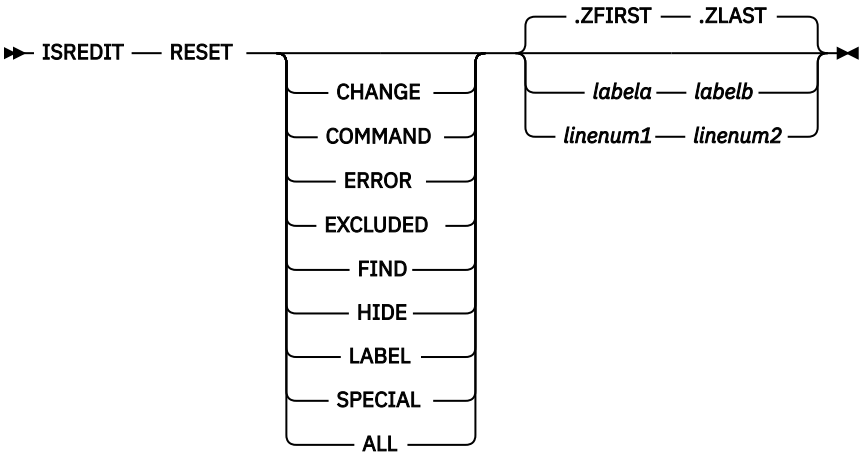
```
ISREDIT REPLACE MEM1 1 10
```

RESET—Reset the Data Display

The RESET macro command can restore line numbers in the line command field when those line numbers have been replaced by labels, pending line commands, error flags, and change flags. However, to reset any pending line commands, you must have specified the NOPROCESS operand in the MACRO command. RESET can also delete special lines from the display, redisplay excluded lines, and temporarily disable the highlighting of FIND strings.

Syntax

Macro command syntax



You can type the operands in any order. If you do not specify any operands, RESET processes all operands except LABEL.

CHANGE
Removes ==CHG> flags from the line command field.

COMMAND
Removes any pending line commands from the line command field.

ERROR
Removes ==ERR> flags from the line command field.

EXCLUDED
Redisplays any excluded line.

FIND
Turns off highlighting of FIND strings until the next FIND, RFIND, CHANGE, or RCHANGE command. However, SEEK and EXCLUDE do not return the highlighting of FIND strings in this manner.

RESET with no operands has the same effect on highlighted FIND strings as RESET FIND.

HIDE
Redisplays all "*n* Line(s) not Displayed" messages for excluded lines that were hidden through the HIDE command.

LABEL
Removes labels from the line command field.

SPECIAL
Deletes any temporary line from the panel:

- Bounds line flagged as =BNDS>
- Column identification lines flagged with =COLS>
- Information lines flagged with ======
- Mask lines flagged as =MASK>
- Message lines flagged as ==MSG>
- Note lines flagged with =NOTE=
- Profile lines flagged as =PROF>
- Tabs line flagged as =TABS>

ALL
Removes all changes to the line number field.

linenum1
Relative line number identifying the start of a group of lines to be reset.

linenum2
Relative line number identifying the end of a group of lines to be reset.

labela, labelb
Labels identifying the start and end of the group of lines to be reset.

For more information about using labels to identify a group of lines, see linergn.dita#linergn.

Description

RESET scans every line of data for conditions to be reset. If you want to delete a small number of special lines, you can get faster response time if you use the D (delete) line command.

Return codes

0
Normal completion

20
Severe error

Examples

To remove all change flags from the current data:

```
ISREDIT RESET CHANGE
```

To remove all error flags from the current data:

```
ISREDIT RESET ERROR
```

To redisplay all excluded lines messages that are hidden:

```
ISREDIT RESET HIDE
```

To redisplay all excluded lines between the .START and .STOP labels:

```
ISREDIT RESET EXCLUDED .START .STOP
```

To remove all labels from the current data between and including the .START and .STOP labels:

```
ISREDIT RESET LABEL .START .STOP
```

To remove all special lines from the current data between lines 100 and 200:

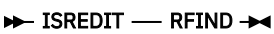
```
ISREDIT RESET SPECIAL 100 200
```

RFIND—Repeat Find

The RFIND macro command locates the search string defined by the most recent SEEK, FIND, or CHANGE command, or excludes a line containing the search string defined by the previous EXCLUDE command.

The RFIND command can be used repeatedly to find other occurrences of the search string. After a *string* NOT FOUND message appears, the next RFIND issued starts at the first line of the current range for a forward search (FIRST or NEXT specified), or the last line of the current range for a backward search (LAST or PREV specified).

Syntax



Return codes

0
Normal completion

4
String not found

12
Syntax error

20
Severe error (string not defined)

Examples

To find a character string, process it, and then repeat the operation for the rest of the data:

```
ISREDIT FIND FIRST C'. the'
SET RETCODE = &LASTCC;
DO WHILE &RETCODE = 0

    ...

  ISREDIT RFIND
  SET RETCODE = &LASTCC;
END
```

RIGHT—Scroll Right

The RIGHT macro command scrolls data to the right of the current panel position.

Syntax

amount
The scroll amount. The number of columns (0-9999) to scroll,

MAX
Displays the last panel of data to the right.

HALF
Displays the next half-panel of data to the right.

PAGE
Displays the next full panel of data to the right.

CURSOR
Scrolls until the column on which the cursor is located becomes the first data column on the panel.

DATA
Scrolls until the last column on the current panel of data becomes the first column on the next panel of data.

Description

The editor stops scrolling when it reaches the current BOUNDS setting. For example, if the right bound is position 100, and positions 9 to 80 are displayed, issuing ISREDIT RIGHT 100 leaves positions 29 to 100 being displayed, not positions 109 to 180.

To scroll to the right using the panel position when the macro was issued, use USER_STATE assignment statements to save and then restore the panel position operands.

If you define a macro named RIGHT, it overrides RIGHT when used from another macro, but has no effect for you. RIGHT does not change the cursor position and cannot be used in an initial macro. See “BOUNDS—Set or Query the Edit Boundaries” on page 305 and “DISPLAY_COLS—Query Display Columns” on page 330 for further information.

Return codes

0
Normal completion

4
No visible lines

8
No data to display

12
Amount not specified

20
Severe error

Examples

To scroll the display to the right by the number of columns specified in variable &RCOL:

```
ISREDIT RIGHT &RCOL
```

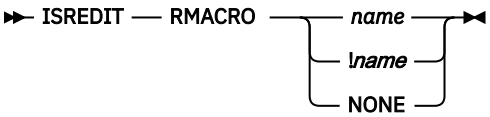
RMACRO—Set or Query the Recovery Macro

The RMACRO macro command sets the name of the recovery macro.

The RMACRO assignment statement sets or retrieves the name of the recovery macro set in this edit session.

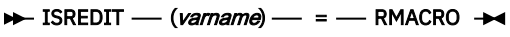
See "Recovery macros" on page 109 for more information.

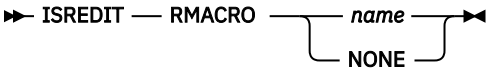
Syntax



name
The name of the recovery macro to be run. The name can be preceded by an exclamation point (!) to show that it is a program macro.

NONE
The name to prevent a recovery macro from being run; conversely, a value of NONE is returned when no recovery macro has been specified.





varname
The name of a variable to contain the name of the recovery macro.

name
Same as macro command syntax.

NONE
Same as macro command syntax.

Return codes

0
Normal completion

12
Invalid name specified

20
Severe error

Examples

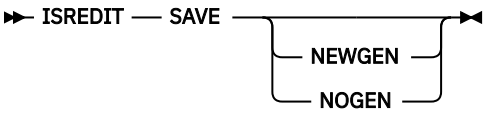
To set the RMACRO name from the variable &RMAC:

```
ISREDIT RMACRO = &RMAC
```

SAVE—Save the Current Data

The SAVE macro command stores the current data on disk. Generally, you do not need to use SAVE if recovery mode is on. See the **DATA_CHANGED**, **AUTOSAVE**, **CANCEL**, and **END** commands for more information about saving data.

Syntax



NEWGEN
Applicable only when editing a member in a PDSE version 2 data set that is configured for member generations. Saves the member in a new generation. This new generation becomes the current generation, also known as generation zero. The generation being edited is left unchanged. This is the default behavior when editing the current generation.

NOGEN
Applicable only when editing a member in a PDSE version 2 data set that is configured for member generations. Saves the member to the same generation that is being edited. This is the default behavior when editing a non-current generation.

Note: The default SAVE behavior when editing a non-current member generation can be changed in the ISPF site configuration table.

Description

The SAVE command writes the data to the same data set from which it was retrieved unless you specified a concatenated sequence of partitioned data sets on the Edit - Entry panel. In that case, the data is saved in the first library in the concatenation sequence, regardless of which library it came from. For a sequential data set, the complete data set is rewritten. For a partitioned data set, the member is rewritten with the same member name.

For a member in a PDSE version 2 data set that is configured for member generations, the behavior depends on the member generation being edited:

- When editing the current generation, also known as generation zero, the default behavior is to write the member to a new generation.
- When editing a non-current generation, the default behavior is to write the member to the same generation that is being edited. The default behavior when editing a non-current generation can be changed in the ISPF site configuration table.
- These default behaviors for member generations can be overridden using the NEWGEN and NOGEN keywords.

If stats mode is on, the library statistics for the member are automatically updated.

If both number mode and autonum mode are on, the data is automatically renumbered before it is saved.

Return codes

0
Normal completion

4
New member saved

8
Data is not saved; the SAVE command was entered while in View.

12
Data is not saved; possible reasons are:

- There is not enough PDS space or directory space available to satisfy the command.
- The NEWGEN or NOGEN parameter was specified while editing a file that is not a member in a PDSE version 2 data set that is configured for member generations.

20
Severe error

Examples

To check autosave mode and, if it is set to OFF, ensure that changes are saved:

```
ISREDIT (VAR) = AUTOSAVE
IF &VAR = OFF THEN -
   ISREDIT SAVE
```

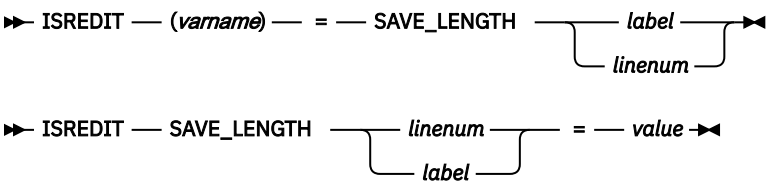
When you are editing generation zero of a member in a PDSE version 2 data set and you want the data to be saved to the same generation (rather than create a new generation) use:

```
ISREDIT SAVE NOGEN
```

SAVE_LENGTH—Set or Query Length for Variable-Length Data

The SAVE_LENGTH macro command sets or queries the length to be used to save each record in a variable-length file. It does not enable you to truncate the nonblank portion of a record, but it does enable you to extend a record. When records are written to disk, they are padded on the end with blanks as needed.

Syntax



Description

You can use the SAVE_LENGTH macro command to set or query the minimum length that is used to store an individual record in a variable-length data set.

When setting a length, the length is automatically adjusted to include the nonblank portion of the line.

When retrieving the length, the number returned reflects the line length that would be used if the line were saved immediately. This is the greater of these two values:

- The length of the nonblank portion of the line *and* the length set by a previous SAVE_LENGTH request.
- The length of the nonblank portion of the line *and* the original line length.

You can use the SAVE_LENGTH command in edit macros to define line commands to prompt the user for final record lengths or to check the record length. You might also use it to substitute a visible character for trailing blanks to make editing easier.

Use of the SAVE_LENGTH command does not prevent the editor from working on data past the specified record length. The length set and returned by the SAVE_LENGTH command is only used when the data is written and does not affect the operation of any other edit functions.

Return codes

0
Normal completion

4
Value supplied on set call was out of range. If the supplied length was too great, it is adjusted to equal the maximum record length. Otherwise, the length was adjusted to the length of the nonblank data portion of the record.

6
Record format is not variable. Any value on an assignment request is ignored.

16
Error setting variable.

20
Severe error

Examples

To save the number of characters that are saved for the last line in the file when PRESERVE OFF is active:

```
ISREDIT (NCHARS) = SAVE_LENGTH .ZLAST
```

To set the minimum line length for the last line in the file and to set PRESERVE ON active:

```
ISREDIT SAVE_LENGTH .ZLAST = 74
```

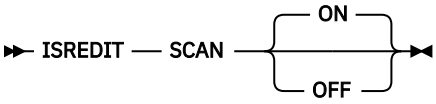
Another edit macro sample using the SAVE_LENGTH command can be found in the ISRSETLN member of the ISPF EXEC library.

SCAN—Set Command Scan Mode

The SCAN macro command sets scan mode, which controls the automatic replacement of variables in command lines passed to the editor.

The SCAN assignment statement either sets the value of scan mode (for variable substitution), or retrieves the value of scan mode and places it in a variable.

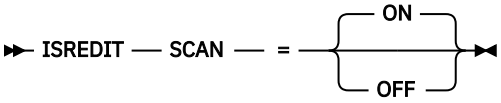
Syntax



ON
Specifies that the editor automatically replaces variables in command lines.

OFF
Specifies that the editor does not automatically replace variables.

Scan mode is initialized to ON when a macro is started.



varname
The name of a variable to contain the setting of scan mode, either ON or OFF.

ON
Same as macro command syntax.

OFF
Same as macro command syntax.

Return codes

0
Normal completion

20
Severe error

Examples

To set a line whose number is in variable &LNUM to:

```
&SYSDATE is a CLIST built-in function
```

set scan mode off and issue the LINE command with &&SYSDATE as the CLIST function name. The CLIST processor strips off the first &, but, because scan mode is off, the editor does not remove the second &:;

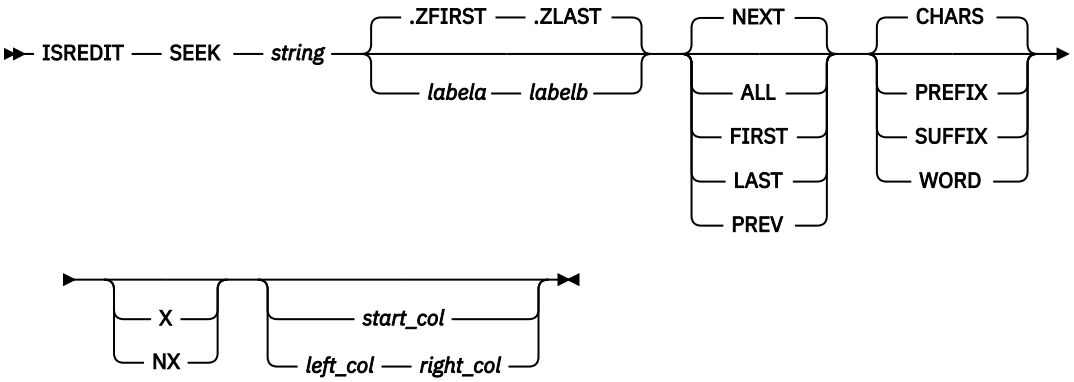
```
ISREDIT SCAN OFF
ISREDIT LINE &LNUM = "&&SYSDATE is a CLIST built-in function"
ISREDIT SCAN ON
```

Because the ISPEXEC call interface for REXX EXECs allows you to specify parameters as symbolic variables, a single scan always takes place before the syntax check of a statement. Therefore, the rule of using two ampersands (&) before variable names to avoid substitution of variable names also applies to REXX EXECs.

SEEK—Seek a Data String, Positioning the Cursor

The SEEK macro command finds one or more occurrences of a search string without changing the exclude status of the line.

Syntax



string
The search string you want to find. The maximum allowable length of the string is 256 bytes. If you are specifying a hex string, the maximum is 128 hexadecimal characters. See “Finding, seeking, changing, and excluding data” on page 44.

labela*, *labelb
Labels identifying the start and end of the group of lines SEEK is to search.

If the cursor is currently placed above the start label and the PREV occurrence of a string is requested, or the cursor is currently placed below the end label and the NEXT occurrence of a string is requested, the process returns a return code of 4 and the string is not found, even if it exists within the label range.

For more information about using labels to identify a group of lines, see "Labels and line ranges" on page 59.

NEXT
Starts at the first position after the current cursor location and searches ahead to find the next occurrence of string.

ALL
Starts at the top of the data and searches ahead to find all occurrences of string.

FIRST
Starts at the top of the data and searches ahead to find the first occurrence of string.

LAST
Starts at the bottom of the data and searches backward to find the last occurrence of string.

PREV
Starts at the current cursor location and searches backward to find the previous occurrence of string.

CHARS
Locates string anywhere the characters match.

PREFIX
Locates string at the beginning of a word.

SUFFIX
Locates string at the end of a word.

WORD
Locates string when it is delimited on both sides by blanks or other non-alphanumeric characters.

X
Scans only lines that are excluded from the display.

NX
Scans only lines that are not excluded from the display.

left_col
The first column to be included in the range of columns SEEK is to search.

right_col
The last column to be included in the range of columns SEEK is to search.

Description

Use the FIND macro command instead of SEEK if you want to locate a string and change the exclude status of the line that contains that string at the same time.

You can use SEEK to find a search string, change it with CHANGE, and then exclude it from the display with EXCLUDE.

To find the next occurrence of the letters ELSE without specifying any other qualifications, include this line in an edit macro:

```
ISREDIT SEEK ELSE
```

Since no other qualifications were specified, the letters ELSE can be:

- Uppercase or a mixture of uppercase and lowercase
- At the beginning of a word (prefix), the end of a word (suffix), or the entire word (word)
- In either an excluded or a non-excluded line
- Anywhere within the current boundaries

To find the next occurrence of the letters ELSE, but only if the letters are uppercase:

```
ISREDIT SEEK C'ELSE'
```

This type of search is called a character string search (note the C that precedes the search string) because it finds the next occurrence of the letters ELSE only if the letters are in uppercase. However, since no other qualifications were specified, the letters can be found anywhere in the data set or member, as outlined in the preceding list.

For more information, including other types of search strings, see "Finding, seeking, changing, and excluding data" on page 44.

Return codes

0
Normal completion

4
String not found

12
Syntax error

20
Severe error

Examples

The example shown here finds the last occurrence in the data set of the letters ELSE. However, the letters must occur on or between lines labeled .E and .S; they must be the last four letters of a word; and they must be found in an excluded line.

```
ISREDIT SEEK ELSE .E .S LAST SUFFIX X
```

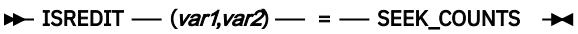
The example shown here finds the first occurrence of the letters ELSE that immediately precedes the cursor position. However, the cursor must not be positioned ahead of the lines that are labeled .E and .S. Also, the letters must occur on or between lines labeled .E and .S; they must be stand-alone characters (not part of any other word); they must be found in a non-excluded line; and they must exist within columns 1 and 5:

```
ISREDIT SEEK ELSE .E .S PREV WORD NX 1 5
```

SEEK_COUNTS—Query Seek Counts

The SEEK_COUNTS assignment statement retrieves the values set by the most recently entered SEEK command and places them in variables.

Syntax



var1
The name of a variable to contain the number of strings found. It must be an 8-character value that is left-padded with zeros.

var2
The name of a variable to contain the number of lines on which strings were found. It must be an 8-character value that is left-padded with zeros.

Return codes

0
Normal completion

20
Severe error

Examples

To seek all lines with a blank in column 1 and store the number of such lines in variable &BLNKS:

```
ISREDIT SEEK ALL " " 1
ISREDIT (BLNKS) = SEEK_COUNTS
```

SESSION—Query Session Type

The SESSION assignment statement identifies the type of session in which the macro is running, Edit, View, EDIF, or VIIF. It also identifies if SCLM is active or not.

Syntax

var1
This variable contains either EDIF, EDIT, VIEW, or VIIF to identify the type of session.

var2
This variable contains SCLM if the SCLM edit environment is active, or four asterisks (****) if not. Until SCLM edit is initialized and is active, edit commands such as SAVE will not update SCLM correctly.

Note: SCLM edit is not available during execution of the site-wide initial edit macro.

Return codes

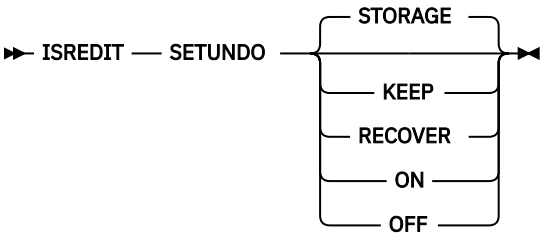
0
Normal completion

20
Severe error

SETUNDO—Set UNDO Mode

The SETUNDO macro command allows the UNDO function to be turned on or off and retrieves the current UNDO status.

Syntax



STORAGE
Enables edit changes to be saved in storage.

KEEP
Has the same effect as STORAGE except the UNDO buffers are not cleared when a SAVE is issued.

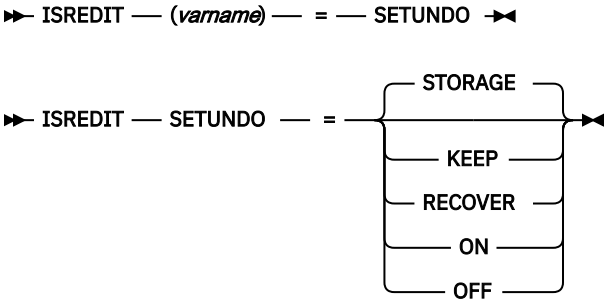
Note: The effect of KEEP (UNDO buffers not cleared when a SAVE is issued) ceases if SETUNDO is subsequently issued without the KEEP keyword.

RECOVER
Enables edit changes to be saved through the recovery file only. If edit recovery is off, SETUNDO RECOVER turns recovery on.

ON
The same as STORAGE.

OFF
Disables the saving of edit changes in storage. If edit recovery is available, the undo command uses the edit recovery file.

Assignment statement syntax



varname
The name of a variable containing the setting of the UNDO mode, either OFF, RECOVER, STORAGE, or KEEP.

STORAGE
Enables edit changes to be saved in storage.

KEEP
Has the same effect as STORAGE except the UNDO buffers are not cleared when a SAVE is issued.

Note: The effect of KEEP (UNDO buffers not cleared when a SAVE is issued) ceases if SETUNDO is subsequently issued without the KEEP keyword.

RECOVER
Enables edit changes to be saved through the recovery file only. If edit recovery is off, SETUNDO RECOVER turns recovery on.

ON
Enables edit changes to be saved in storage.

OFF
Disables the saving of edit changes in storage. If edit recovery is available, the undo command uses the edit recovery file.

Description

The SETUNDO macro command enables undo processing. It does not perform the undo function itself. Valid operands are STORAGE, KEEP, RECOVER, ON, or OFF.

If SETUNDO is set on by a macro and was not on already, the UNDO function is enabled for all interactions started from the point SETUNDO was turned on.

Note:

1. Changes are saved on the undo chain after:
   - SETUNDO STORAGE or SETUNDO KEEP is specified in a macro, and it was previously OFF or REC, or
   - SETUNDO REC is specified in a macro, and it was previously OFF

   It is possible to undo back to a particular point in a macro. This is helpful in debugging edit macros.
2. If SETUNDO is disabled through the configuration table, the SETUNDO macro command is accepted and returns a zero return code. It does not turn recovery on.
3. The SETUNDO command is ignored if UNDO from storage is not enabled by the installer or person who maintains the ISPF product. For information on enabling UNDO from storage, see *z/OS ISPF Planning and Customizing*.

Return codes

0
Successful completion. SETUNDO was turned on or off, or status remains unchanged because UNDO was already on or off.

20
Severe error. Probably a parameter error (something other than STG, KEEP, REC, or OFF was specified).

Examples

To disable the saving of edit changes in storage:

```
ISREDIT SETUNDO OFF
```

To enable the saving of edit changes in storage:

```
ISREDIT SETUNDO = STORAGE
```

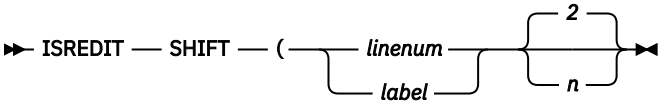
To store the value of SETUNDO in the variable &SET:

```
ISREDIT (SET) = SETUNDO
```

SHIFT (—Shift Columns Left

The SHIFT (macro command moves characters on a line to the left without altering their relative spacing. Characters shifted past the current BOUNDS setting are deleted. See “Shifting data” on page 42 for more information.

Syntax



linenum
A relative line number identifying the line on which characters are to be moved to the left.

label
A label identifying the line on which characters are to be moved to the left.

n
Specifies the number of columns to shift.

Description

The SHIFT (command is limited to shifting columns of data on a single line. If you want to shift columns of data on several lines, each line of data columns must be moved individually.

Return codes

0
Normal completion

12
Invalid line number

20
Severe error

Examples

To shift columns of data 10 columns to the left on the line that contains the cursor:

```
ISREDIT SHIFT ( .ZCSR 10
```

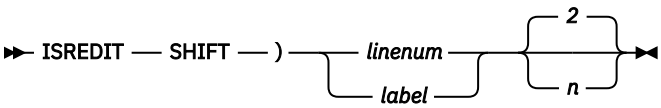
To shift columns of data 2 columns to the left on the line with the label .LAB:

```
ISREDIT SHIFT ( .LAB
```

SHIFT)—Shift Columns Right

The SHIFT) macro command moves characters on a line to the right without altering their relative spacing. Characters shifted past the current BOUNDS setting are deleted. See “Shifting data” on page 42 for more information.

Syntax



linenum
A relative line number identifying the line on which characters are to be moved to the right.

label
A label identifying the line on which characters are to be moved to the right.

n
Specifies the number of columns to shift.

Description

The SHIFT) command is limited to shifting columns of data on a single line. If you want to shift columns of data on several lines, each line of data columns must be moved individually.

Return codes

0
Normal completion

12
Invalid line number

20
Severe error

Examples

To shift columns of data 4 columns to the right on the line that contains the cursor:

```
ISREDIT SHIFT ) .ZCSR 4
```

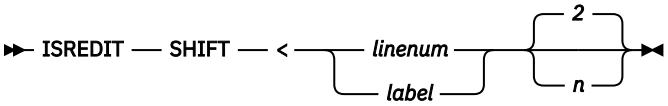
To shift columns of data 2 columns to the right on the line with the label .LAB:

```
ISREDIT SHIFT ) .LAB
```

SHIFT <—Shift Data Left

The SHIFT < macro command moves the body of a program statement to the left without shifting the label or comments. This command prevents loss of nonblank characters by stopping before shifting nonblank characters past the bound. See “Shifting data” on page 42 for more information.

Syntax



linenum
A relative line number identifying the line on which the body of a program statement is to be moved to the left.

label
A label identifying the line on which the body of a program statement is to be moved to the left.

n
Specifies the number of columns to shift.

Description

The SHIFT < command is limited to shifting data on a single line. To shift data on several lines, you must shift data on each line individually.

Return codes

0
Normal completion

12
Invalid line number

20
Severe error

Examples

To shift data 4 columns to the left on the line that contains the cursor:

```
ISREDIT SHIFT < .ZCSR 4
```

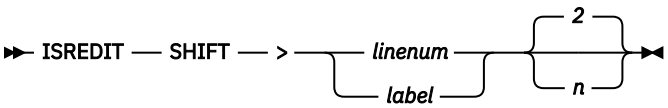
To shift data 2 columns to the left on the line with the label .LAB:

```
ISREDIT SHIFT < .LAB
```

SHIFT >—Shift Data Right

The SHIFT > macro command moves the body of a program statement to the right without shifting the label or comments. This command prevents loss of nonblank characters by stopping before shifting nonblank characters past the bound. See “Shifting data” on page 42 for more information.

Syntax



linenum
A relative line number identifying the line on which the body of a program statement is to be moved to the right.

label
A label identifying the line on which the body of a program statement is to be moved to the right.

n
Specifies the number of columns to shift.

Description

The SHIFT > command is limited to shifting data on a single line. To shift data on several lines, you must shift data on each line individually.

Return codes

0
Normal completion

12
Invalid line number

20
Severe error

Examples

To shift data 4 columns to the right on the line that contains the cursor:

```
ISREDIT SHIFT > .ZCSR 4
```

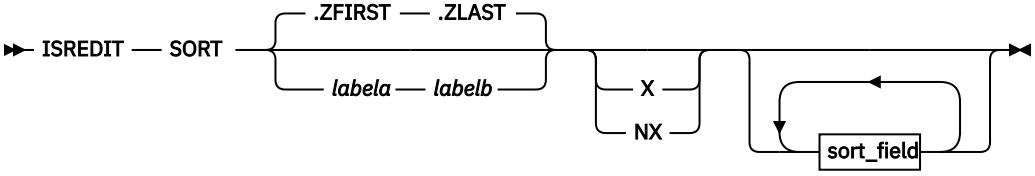
To shift data 2 columns to the right on the line with the label .LAB:

```
ISREDIT SHIFT > .LAB
```

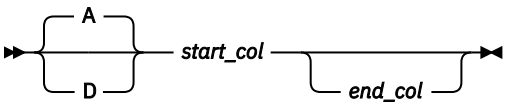
SORT—Sort Data

The SORT macro command puts data in a specified order.

Syntax



sort_field



labela, labelb
Labels identifying the start and end of the group of lines for the sort operation.

For more information about using labels to identify a group of lines, see "Labels and line ranges" on page 59.

X
Specifies that only excluded lines are to be sorted.

NX
Specifies that only non-excluded lines are to be sorted.

sort_field
Specifies the field to be used in sorting data. You can specify up to five sort fields using these operands:

A
Specifies ascending order. It can either precede or follow the column specification.

D
Specifies descending order. It can either precede or follow the column specification.

start_col
Defines the starting column of the field that is to be compared. It must be within the current boundaries.

end_col
Defines the ending column of the field that is to be compared. It must be within the current boundaries.

If you specify several fields, you must specify both the starting and ending columns of each field. The fields cannot overlap. If you specify A or D for one field, you must specify it for all fields.

Description

The SORT command operates in two different modes, based on the hexadecimal mode status. If hexadecimal mode is on, the data is ordered according to its hexadecimal representation. If hexadecimal mode is off, data is sorted in the collating sequence defined for the national language being used.

Sorting data without operands

For a SORT command with no operands, the editor compares the data within the current boundaries character by character, and then orders it line by line in the proper collating sequence. It ignores data outside the current boundaries during both operations. This means that only the data inside the current boundaries is changed. Labels, excluded lines, line numbers, and change, error, and special line flags are considered associated with the data, and therefore points to the same data fields after the sort as they did before the sort.

For example, if you issue a CHANGE ALL command that changes the first, third, and sixth lines in a data set, these lines are flagged with the change flag, ==CHG>. If you then issue a SORT command that results in the former lines 1, 3 and 6 becoming the first, second and third lines of the sorted file, the changed line flags would now exist on the first, second and third lines of the sorted data set.

It is important to properly set the boundaries before issuing the SORT command. SORT is a powerful tool for editing data that may be formatted in multiple columns. You can set the boundaries, for example, to the first half of a record and sort one column of data. Then you can set the boundaries to the last half of the record and sort a second column of data.

Limiting the SORT command

You can specify up to five sort fields by labeling starting and ending columns. You can identify each field as having data sorted in ascending or descending order.

Optionally, you can limit sorting to a range of lines by specifying the labels of the first and last lines of the range. You can also limit sorting to either excluded or non-excluded lines.

If you have labels or line ranges that are between the labels or line ranges specified with the SORT command, you can keep SORT from rearranging them by:

- Excluding them before you enter the SORT command
- Using the NX operand to sort only lines that are not excluded

See the definition of the NX operand and “EXCLUDE—Exclude Lines from the Display” on page 232 for more information.

Sorting DBCS data

When sorting data that contains DBCS character strings, you must ensure that no DBCS string crosses the boundaries. Also, all records must have the same format at the boundaries, although the format of the left and right boundaries can differ.

If a boundary divides a DBCS character, or if all records do not have the same format at the boundaries, the result is unpredictable.

Return codes

0
Normal completion

4
Lines were already in sort order

8
No records to sort

16
Not enough storage to perform sort

20
Severe error

Examples

To sort the data in descending order, using the sort key in columns 15 through 20:

```
ISREDIT SORT D 15 20
```

To sort all excluded lines in ascending order:

```
ISREDIT SORT X A
```

SOURCE—describe format of data

The SOURCE macro command instructs the editor to treat the source data as though it is in the specified format and converts it from that format to the CCSID of the terminal for display purposes, although the data remains unchanged within the file. When you input or modify data at the terminal, the editor translates the data entered from the CCSID of the terminal to the specified format prior to storing the data in the file.

Syntax

The SOURCE ASCII macro command is not available when editing a z/OS UNIX file. Instead, use the ASCII edit facility to have the data automatically translated from ASCII to the CCSID of the terminal.

character_encoding
The type of character encoding to be used for translating data when displaying or receiving input from the terminal.

Valid values are:

- ASCII

See "Working with ASCII data" on page 50 for more information.

Examples

To set source mode to ASCII:

```
SOURCE ASCII
```

To revert back to normal mode, use the RESET command:

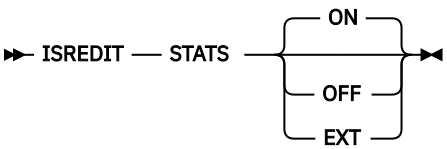
```
RESET SOURCE
```

STATS—Set or Query Stats Mode

The STATS macro command sets stats mode, which creates and maintains statistics for a member of a partitioned data set.

The STATS assignment statement either sets stats mode, or retrieves the setting of stats mode and places it in a variable.

Syntax



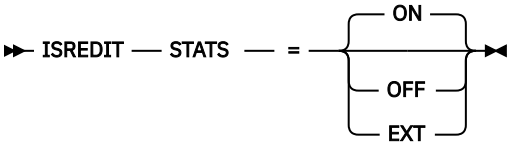
ON
Creates or updates library statistics when the data is saved.

If extended statistics are enabled in the site configuration and any of the line number statistic values exceed 65535, the statistics are automatically stored as extended statistics. Otherwise, the statistics are automatically stored as non-extended statistics. Extended statistics contain extended line count fields that can store values up to 2147483647; non-extended statistics do not contain the extended line count fields and can only store line number values up to 65535. If extended statistics are not enabled in the site configuration, 65535 is stored for line number statistic values that exceed 65535.

OFF
Does not create or update library statistics.

If STATS mode is off when you save a member, any previous statistics are lost.

EXT
Has the same function as ON.



varname
The name of a variable to contain the setting of stats mode, either ON or OFF.

ON
Same as macro command syntax.

OFF
Same as macro command syntax.

EXT
Same as macro command syntax.

See "Statistics for PDS members" on page 25 for more information.

Return codes

0
Normal completion

20
Severe error

Examples

To put the value of stats mode in variable &LIBSTAT:

```
ISREDIT (LIBSTAT) = STATS
```

To set stats mode on:

```
ISREDIT STATS = ON
```

To set stats mode off:

```
ISREDIT STATS OFF
```

To reset stats mode from the mode saved in variable &LIBSTAT:

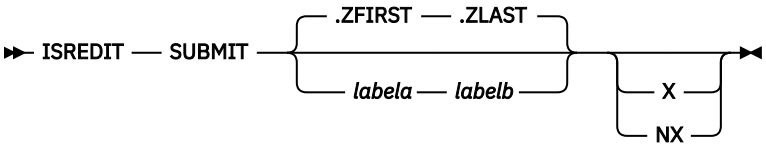
```
ISREDIT STATS = &LIBSTAT
```

SUBMIT—Submit Data for Batch Processing

The SUBMIT macro command submits the member or data set you are editing (or the part of the member or data set defined by the range of line pointers or the X or NX parameters) to be processed as a batch job.

Syntax

Macro command syntax



labela, labelb
Labels identifying the start and end of the group of lines to be submitted.

For more information about using labels to identify a group of lines, see linergn.dita#linergn.

X
Submits only lines that are excluded from the display.

NX
Submits only lines that are not excluded from the display.

Description

The editor does not supply a job statement when you enter the SUBMIT command. You can supply job statements as part of the data being submitted. When you supply a job statement, only the job name is logged to the ISPF log data set to ensure the protection of sensitive data.

If the file being edited is described as ASCII or UTF-8 then the data submitted to the internal reader is converted to EBCDIC.

PDF uses TSO SUBMIT to submit the job.

Return codes

0
Normal completion

20
Severe error (submit failed)

Examples

To submit the first 20 lines of the data as a batch job:

```
ISREDIT SUBMIT 1 20
```

To submit all of the data as a batch job:

```
ISREDIT SUBMIT
```

To submit only the non-excluded lines as a batch job:

```
ISREDIT SUBMIT NX
```

TABS—Set or Query Tabs Mode

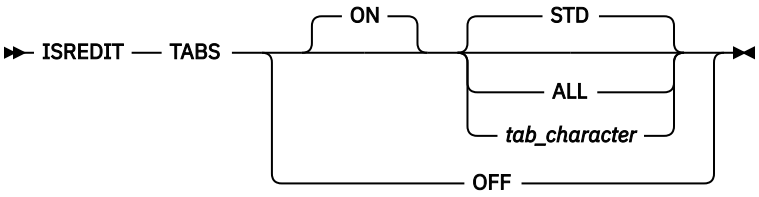
The TABS macro command:

- Turns tabs mode on and off
- Defines the logical tab character
- Controls the insertion of attribute bytes at hardware tab positions defined with the TABS line command

The TABS assignment statement does everything the macro command can do. It can also retrieve the setting of tabs mode and place it in a variable.

Use PROFILE to check the setting of tabs mode and the logical tab character. See “Using tabs” on page 63 if you need more information about using tabs.

Syntax



ON
Turns tabs mode on, which means that logical tabs can be used to break up strings of data.

OFF
Turns tabs mode off, which means that logical tabs cannot be used. Attribute bytes are deleted from all hardware tab positions, causing the Tab Forward and Tab Backward keys to ignore hardware tabs defined on the =TABS> line. Blanked-out characters occupying these positions reappear. The TABS OFF message appears in the profile display.

STD
Activates all hardware tab positions (asterisks) that contain a blank or null character. The editor inserts attribute bytes, which cannot be typed over, at these positions. You can use the Tab Forward and Tab Backward keys to move the cursor one space to the right of the attribute bytes. The `TABS ON STD` message appears in the profile display.

ALL
Causes an attribute byte to be inserted at all hardware tab positions. Characters occupying these positions are blanked out and the attribute bytes cannot be typed over. The Tab Forward and Tab Backward keys can be used to move the cursor one space to the right of these attribute bytes. The `TABS ON ALL` message appears in the profile display.

tab_character
Defines a single character that is not a number, letter, or command delimiter as the logical tab character. This character is used with hardware tab definitions. The `TABS ON tab character` message appears in the profile display.

You can enclose the character in quotes (`'` or `"`), although this is not necessary unless you want to use one of these characters as the tab character:

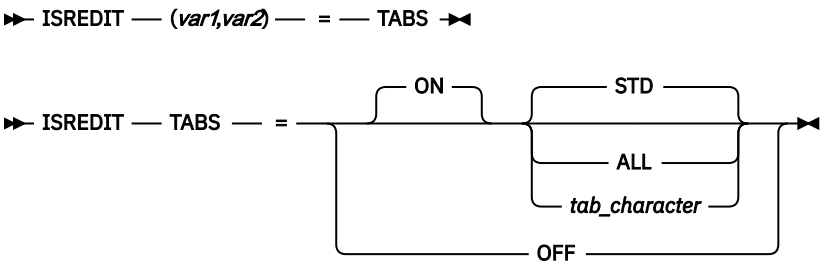
```
=  '  "  <  ,  (  +
```

The ampersand (&), left bracket ([), and right bracket (]) should not be used as tab characters at all.

The *tab_character* operand causes the data string that follows the logical tab character to align itself one space to the right of the first available hardware tab position when you press Enter. No attribute bytes are inserted.

If no hardware tabs are defined, the editor aligns the data vertically. If software tabs are defined, the first data string is aligned under the first software tab position and the remaining data strings are aligned at the left boundary. If neither software nor hardware tabs are defined, the editor aligns all the data strings at the left boundary.

With the *tab_character* operand, the Tab Forward and Tab Backward keys ignore hardware tab positions when the *tab_character* operand is used because no attribute bytes are inserted.



var1
The name of a variable to contain the setting of tabs mode, either ON or OFF.

var2
The name of a variable to contain the tab character and either ALL or STD. This variable may be blank.

ON
Same as macro command syntax.

OFF
Same as macro command syntax.

STD
Same as macro command syntax.

ALL
Same as macro command syntax.

tab_character
Same as macro command syntax.

Return codes

0
Normal completion

20
Severe error

Examples

To set the tab character to \ and set the tabs mode ON:

```
ISREDIT TABS ON \
```

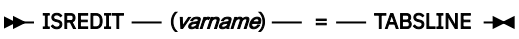
To set the value of tabs mode from variable &TABVAL:

```
ISREDIT TABS = (TABVAL)
```

TABSLINE—Set or Query Tabs Line

The TABSLINE assignment statement either sets the tabs line, or retrieves the tabs line and places it in a variable.

Syntax



varname
Specifies the name of a variable to hold the contents of the current tabs line.

data
Specifies the data used to set the tabs line. The only valid tab characters for this data are blanks, asterisks (*), hyphens (-), and underscores (_). These forms can be used:

- Simple string
- Delimited string
- Variable
- Template (< *col*,*string* >)
- Merge format (*string1* + *string2*, operand + *string2*, *string1* + operand)
- Operand:

  LINE *linenum*
  Data from the line with the given relative line number.

  LINE *label*
  Data from the line with the given label.

  MASKLINE
  Data from the mask line.

  TABSLINE
  Data from the tabs line.

Return codes

0
Normal completion

4
Data truncated

8
Invalid data detected and ignored

20
Severe error (invalid input)

Examples

To store the value of the tabs line in variable &OLDTABS:

```
ISREDIT (OLDTABS) = TABSLINE
```

To set the tabs line to "*___* *":

```
ISREDIT TABSLINE = "*___*   *"
```

To clear the tabs line:

```
ISREDIT TABSLINE = " "
```

To set tabs in columns 1 and 35:

```
ISREDIT TABSLINE = <1,*,35,*>
```

To add a tab in column 36:

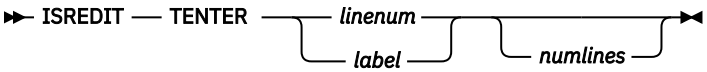
```
ISREDIT TABSLINE = TABSLINE + <36,*>
```

TENTER—Set Up Panel for Text Entry

The TENTER macro command provides one very long line wrapped around onto many rows of the panel to allow power typing for text entry. The editor does the formatting for you.

The TENTER command is different from the INSERT command in that the INSERT command inserts a specified number of separate, blank lines and the mask, if any, just as you typed it. With the TENTER command, however, mask line characters are applied only to the new lines created when the text is flowed outside the boundaries. Any mask line characters within the bounds are ignored.

Syntax



linenum
A relative line number identifying the line.

label
A label identifying the line.

numlines
Specifies the number of lines displayed for text entry; these lines are not saved unless they contain data. If you do not type a number, the remainder of the panel appears for text entry.

Description

It is important to make sure that the line referenced by the line pointer on TENTER appears; otherwise, the text area will not be visible to you. Use LOCATE to find and display the line for you.

Before you enter text entry mode:

- If you are going to be typing text in paragraph form, such as for a memo or letter, make sure caps mode is off. Otherwise, when you press Enter, your text will change to uppercase.
- You may want to turn off number mode to prevent sequence numbers from writing over any of your text.
- Make sure the bounds setting is where you want it so that the text flows correctly when you end text entry mode.
- Once you enter text entry mode, no macros can be run.

To enter text entry mode:

1. Include this command in an edit macro:

   ```
   ISREDIT TENTER linenum numlines
   ```

   or

   ```
   ISREDIT TENTER label numlines
   ```

   If *numlines* is greater than the number of rows remaining on the panel, the vertical bar that indicates where you will run out of room does not appear and the keyboard does not lock at the last character position on the panel. When you run the edit macro (see step “2” on page 419), you can scroll down to bring the additional blank text entry space into view.

2. Run the edit macro. The editor inserts a single continuous blank area for the specified number of rows or to the bottom of the panel.

To begin a new paragraph:

1. Use the return (Enter), cursor movement, or Tab keys to advance the cursor enough spaces to leave one blank row on the panel.

If there are insufficient blank spaces on the panel, the keyboard locks when you try to type beyond the last character position. A vertical bar (|) appears above the cursor at the locked position.

To generate more blank spaces:

1. Press the Reset key to unlock the keyboard.
2. Press Enter.

To end text entry mode:

1. Press Enter. The data is flowed together into a paragraph and any embedded blanks are preserved. The left and right sides of the paragraph are determined by the current bounds.

See "Word processing" on page 61 and "Entering text (power typing)" on page 63 for more information.

Return codes

0
Normal completion

12
Invalid line number

20
Severe error

Examples

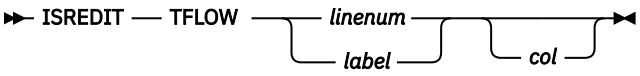
To find the last line in the data and set up the display for text entry following the last line:

```
ISREDIT LOCATE .ZL
ISREDIT TENTER .ZL
```

TFLOW—Text Flow a Paragraph

The TFLOW macro command restructures paragraphs. This is sometimes necessary after deletions, insertions, splitting, and so forth. See "Word processing" on page 61 and "Formatting paragraphs" on page 61 for more information.

Syntax



linenum
A relative line number identifying the line.

label
A label identifying the line.

col
Specifies the column to which the text should be flowed. If the column number is omitted, it defaults to the right boundary. This is different from the TF (text flow) line command, which defaults to the panel width when default boundaries are in effect.

If a number greater than the right boundary is specified, the right boundary is used.

Return codes

0
Normal completion

12
Invalid line number

20
Severe error

Examples

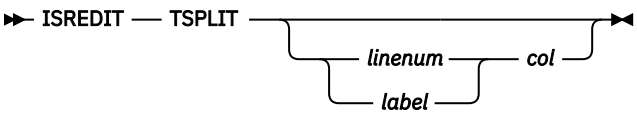
To limit the flow of text, starting at label .PP, to the displayed columns:

```
ISREDIT (,RCOL) = DISPLAY_COLS
ISREDIT TFLOW .PP &RCOL
```

TSPLIT—Text Split a Line

The TSPLIT macro command moves part or all of a line of text to the following line. This makes it easier for you to add new material to existing text.

Syntax



linenum
A relative line number identifying the line where the split is to occur.

label
A label identifying the line where the split is to occur.

col
Specifies the column at which the text is to be split.

If you omit both operands, the split point is assumed to be the current cursor position.

Description

The TSPLIT macro command is affected by the current setting of the boundaries. For instance, data beyond the right boundary is not moved to the line added by TSPLIT. Data between the split column and the right boundary is moved to a new line. The cursor position is set to the split point.

To rejoin lines, use the TFLOW macro command. See “TFLOW—Text Flow a Paragraph” on page 419 for more information.

For more information about splitting lines and other word processing commands, see “Word processing” on page 61 and “Splitting lines” on page 62.

Return codes

0
Normal completion

12
Invalid line number

20
Severe error

Examples

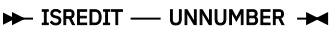
To split the line labeled .TOP at column 15:

```
ISREDIT (LINENBR) = LINENUM .TOP
ISREDIT TSPLIT &LINENBR 15
```

UNNUMBER—Remove Sequence Numbers

The UNNUMBER macro command sets all sequence fields to blanks, turns off number mode, and positions the data so that column 1 is the first column displayed.

Syntax



Description

The UNNUMBER command is valid only when number mode is also on. The standard sequence field, the COBOL sequence field, or both, are blanked out.

Return codes

0
Normal completion

12
Number mode not on

20
Severe error

Examples

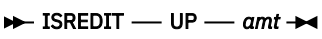
To set all sequence fields to blanks, turn number mode off, and position the panel so that column 1 is the first column displayed:

```
ISREDIT UNNUMBER
```

UP—Scroll Up

The UP macro command scrolls data up from the current panel position.

Syntax



amt
The scroll amount, the number of lines (0-9999) to scroll, or one of these operands:

MAX
Displays the first panel of data.

HALF
Displays the previous half-panel of data.

PAGE
Displays the previous full panel of data.

CURSOR
Scrolls until the line on which the cursor is located becomes the last data line on the panel.

DATA
Scrolls until the first data line on the current panel becomes the last data line on the next panel.

Description

To scroll up using the panel position when the macro was issued, use USER_STATE assignment statements to save and then restore the panel position operands.

When you issue the UP command, the non-data lines on the panel affect the number of lines scrolled. However, if you define a macro named UP, it only overrides UP when used from another macro. UP does not change the cursor position and cannot be used in an initial macro.

The actual number of lines to appear on the panel is determined by:

- The number of lines excluded from the panel
- The terminal display size and split panel line
- The number of special temporary lines displayed, such as the ==ERR>, ==CHG>, =PROF>, =MASK>, =BNDS>, =TABS>, ==MSG>, =NOTE=, =COLS>, and ====== lines.

The first line displayed is determined in one of two ways: (1) a LOCATE command can actually set the line to be first on the panel, or (2) the first line to be displayed depends on whether the cursor was explicitly set by a CURSOR assignment statement or implicitly set by a SEEK, FIND, CHANGE, or TSPLIT command. Since the cursor must be on the panel, the line that is first on the panel may be different from the line that was first when you started the macro.

Return codes

0
Normal completion

2
No more data UP

4
No visible lines

8
No data to display

12
Amount not specified

20
Severe error

Examples

To scroll up to the top of the data set:

```
ISREDIT UP MAX
```

To display the previous half panel of data:

```
ISREDIT UP HALF
```

To display the previous full panel of data:

```
ISREDIT UP PAGE
```

To make the line where the cursor is placed the last one on the display:

```
ISREDIT UP CURSOR
```

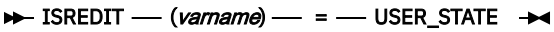
To display the previous page less one line:

```
ISREDIT UP DATA
```

USER_STATE—Save or Restore User State

The USER_STATE assignment statement saves or restores the state of edit profile values, FIND, CHANGE, SEEK, and EXCLUDE values, and panel and cursor values.

Syntax



varname
The name of a variable to contain your status information.

Note: The information in the variable is saved in an internal format that is subject to change. Dependence on the format can lead to macro errors.

Description

USER_STATE can be used at the beginning of a macro to save conditions, and at the end of a macro to restore the conditions that may have changed during processing. Many of the values saved by USER_STATE can be saved and restored individually. The USER_STATE assignment statement is a simple way of saving many values with a single statement.

These edit modes and values are saved and restored by USER_STATE:

```
AUTOLIST  CURSOR        NOTES    RECOVERY
AUTONUM   HEX           NULLS    STATS
AUTOSAVE  IMACRO        NUMBER   TABS
BOUNDS    MASKLINE      PACK     TABSLINE
CAPS      MODEL CLASS   PROFILE
```

Return codes

0
Normal completion

20
Severe error

Examples

To save the user state in variable &STATUS:

```
ISREDIT (STATUS) = USER_STATE
```

To restore the user state from variable &STATUS:

```
ISREDIT USER_STATE = (STATUS)
```

VERSION—Set or Query Version Number

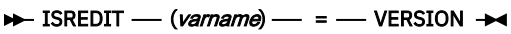
The VERSION macro command allows you to change the version number assigned to a member of an ISPF library.

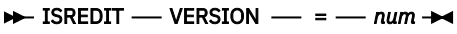
The VERSION assignment statement either sets the version number, or retrieves the version number and places it in a variable.

For more information about version numbers, see "Version and modification level numbers" on page 26.

Syntax

num
The version number. It can be any number from 1 to 99.





varname
The name of a variable to contain the version number. The version number is a 2-digit value that is left-padded with zeros.

num
Same as macro command syntax.

Return codes

0
Normal completion

4
Stats mode is off, the command is ignored

12
Invalid value specified (the version must be 1 to 99)

20
Severe error

Examples

To save the version number in variable &VERS:

```
ISREDIT (VERS) = VERSION
```

To set the version number to 1:

```
ISREDIT VERSION 1
```

To set the version number from variable &VERS:

```
ISREDIT VERSION = &VERS
```

VIEW—View from within an Edit Session

The VIEW macro command allows you to view a member of the same partitioned data set during your current edit session.

Syntax

member
A member of the library or other partitioned data set you are currently editing. You may enter a member pattern to generate a member list.

Description

Your initial edit session is suspended until the view session is complete. Editing sessions can be nested until you run out of storage.

To exit from the view session, END or CANCEL must be processed by a macro or entered by you. The current edit session resumes.

The VIEW service call, ISPEXEC VIEW, is an alternate method of starting view. It offers the option of viewing another data set and specifying an initial macro.

For more information on using the VIEW service, refer to the *z/OS ISPF Services Guide*.

Return codes

0
Normal completion

12
Your error (invalid member name, recovery pending)

20
Severe error

Examples

To view the member OLDMEM in your current ISPF library:

```
ISREDIT VIEW OLDMEM
```

VOLUME—Query Volume Information

The VOLUME assignment statement retrieves the volume serial number (or serial numbers) and the number of volumes on which the data set resides.

Syntax

var1
The name of a variable to contain the serial number of the volume on which the data set resides. For a multivolume data set, this will be the serial number of the first volume. The volume serial number is a six character value.

var2
The name of a variable to contain the number of volumes the data set occupies. The number of volumes is a two-character value.

var3
The name of a variable to contain the serial number of the volume of the original data set.

Return codes

0
Normal completion

4
The data set is a multivolume data set and the shared pool variable ZEDMVOL is set to contain all the volume serial numbers of the data set. ZEDMVOL has the length of the number of volumes times six.

20
Severe error

Examples

To retrieve just the volume serial number of the data set:

```
ISREDIT (VOL) = VOLUME
```

To retrieve just the number of volumes the data set occupies:

```
ISREDIT (,NUMVOL) = VOLUME
```

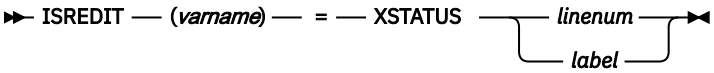
To retrieve both the volume serial number and the number of volumes the data set occupies:

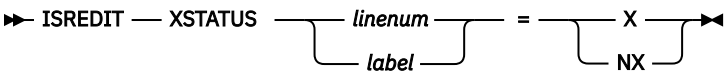
```
ISREDIT (VOL,NUMVOL) = VOLUME
```

XSTATUS—Set or Query Exclude Status of a Line

The XSTATUS assignment statement either sets the exclude status of the specified data line, or retrieves the exclude status of the specified data line and places it in a variable.

Syntax





varname
The name of a variable to contain the exclude status, either X or NX.

linenum
A relative line number identifying the line.

label
A label identifying the line.

X
Specifies that the specified line is to be excluded.

NX
Specifies that the specified line is to be shown (non-excluded).

Description

Exclude status determines whether the line is excluded.

If you want to exclude several lines at one time, the EXCLUDE command should be used. Similarly, to show several lines at one time, use the FIND command.

Return codes

0
Normal completion

8
An attempt to set a line status to NX could not be performed. The line has a pending line command on it. For example, if an excluded line contains an M line command in the line command field, then the `MOVE/COPY IS PENDING` message is displayed and the lines cannot be shown. The reset command can be used to remove your line commands from the line command field.

12
Line number is not an existing line.

20
Severe error

Examples

Use XSTATUS together with SEEK and CHANGE to preserve the exclude status of a line. For example, to store the exclude status of the line whose number is in variable &N in variable &LINEX:

```
ISREDIT (LINEX) = XSTATUS &N
```

To exclude line 1:

```
ISREDIT XSTATUS 1 = X
```

To locate a string and change it, saving and then restoring the exclude status:

```
ISREDIT SEEK &DATA
IF &LASTCC = 0 THEN -
  DO
    ISREDIT (XLINE) = XSTATUS .ZCSR
    ISREDIT CHANGE &DATA &NEWDATA .aZCSR .ZCSR
```

```
    ISREDIT XSTATUS .ZCSR = (XLINE)
  END
```

Appendix A. Abbreviations for Commands and Other Values

This topic lists the command names and keywords that can be aliased, followed by the allowable aliases and abbreviations.

Note:

1. To improve readability, do not use abbreviations in edit macros.
2. ISPF scans the NUMBER macro as a command. If you want to define NUMBER as a program macro and use the abbreviated form, define the abbreviations as program macros also.

Edit line commands

Table 21 on page 429 shows the allowable aliases and abbreviations for Edit line commands.

Table 21. Aliases and abbreviations for Edit line commands

Alias or abbreviation	Full line command
BND	BOUNDS
BNDS	BOUNDS
BOU	BOUNDS
BOUND	BOUNDS
COL	COLS
LCLC	LCC
MDMD	MDD
TAB	TABS
UCUC	UCC

Edit primary commands

Table 22 on page 429 shows the allowable aliases and abbreviations for Edit primary commands.

Table 22. Aliases and abbreviations for Edit primary commands

Alias or abbreviation	Full primary command
BND	BOUNDS
BNDS	BOUNDS
BOU	BOUNDS
BOUND	BOUNDS
C	CHANGE

Table 22. Aliases and abbreviations for Edit primary commands (continued)

Alias or abbreviation	Full primary command
CAN	CANCEL
CHA	CHANGE
CHG	CHANGE
COL	COLS
COLUMNS	COLS
CRE	CREATE
DEF	DEFINE
DEL	DELETE
EDSET	EDITSET
EX	EXCLUDE
EXC	EXCLUDE
EXCLUDE	EXCLUDE
EXCLUDED	EXCLUDE
F	FIND
HI	HILITE
HILIGHT	HILITE
L	LOCATE
LEV	LEVEL
LOC	LOCATE
MOD	MODEL
NONUM	NONUMBER
NONUMB	NONUMBER
NONUMBR	NONUMBER
NOTE	NOTES
NUL	NULLS
NULL	NULLS
NUM	NUMBER
NUMB	NUMBER
PR	PROFILE
PRO	PROFILE
PROF	PROFILE
REC	RECOVERY
RECOV	RECOVERY
RECOVER	RECOVERY

Table 22. Aliases and abbreviations for Edit primary commands (continued)

Alias or abbreviation	Full primary command
RECOVRY	RECOVERY
RECVR	RECOVERY
RECVRY	RECOVERY
REN	RENUM
REP	REPLACE
REPL	REPLACE
RES	RESET
SETU	SETUNDO
SUB	SUBMIT
TAB	TABS
UNN	UNNUMBER
UNNUM	UNNUMBER
UNNUMB	UNNUMBER
VER	VERSION
VERS	VERSION
X	EXCLUDE

Parameters

Table 23 on page 431 shows the allowable abbreviations for parameters.

Table 23. Allowable abbreviations for parameters

Abbreviation	Full parameter name
AFT	AFTER
BEF	BEFORE

Keywords/Operands

Table 24 on page 431 shows the allowable aliases and abbreviations for keywords and operands.

Table 24. Aliases and abbreviations for keywords and operands

Alias or abbreviation	Full keyword/operand
CHAR	CHARS
CHG	CHANGE
COM	COMMAND

Table 24. Aliases and abbreviations for keywords and operands (continued)

Alias or abbreviation	Full keyword/operand
CUR	CURSOR
DIS	DISABLED
DISAB	DISABLED
DISABLE	DISABLED
DISP	DISPLAY
DISPL	DISPLAY
DO	DOLOGIC
ERR	ERROR
IF	IFLOGIC
LAB	LABEL
LABELS	LABEL
PRE	PREFIX
REC	RECOVER
RECOVERY	RECOVER
SPE	SPECIAL
STD	STANDARD
STG	STORAGE
STO	STORAGE
STOR	STORAGE
STORE	STORAGE
SUF	SUFFIX
VERT	VERTICAL

Scroll amounts

Table 25 on page 432 shows the allowable aliases and abbreviations for scroll amounts.

Table 25. Aliases and abbreviations for scroll amounts

Alias or abbreviation	Full scroll operand
C	CUR
CSR	CUR
D	DATA
H	HALF
M	MAX
P	PAGE

Appendix B. Edit-related sample macros

The edit macros listed here are included in the ISPF samples library.

These sample macros are explained in Part 2, “Edit macros,” on page 77. They demonstrate various techniques you can use when writing, running, and testing macros.

ISRBLOCK
Source code for the Block Letter Model selection panel.

ISRBOX
Edit macro that draws a box with its upper left corner at the cursor position.

ISRCHGS
Sample edit macro that shows the lines most recently changed and excludes all other lines.

ISRCOUNT
Edit macro that finds occurrences of a string and returns a count of the number found. Demonstrates passing parameters, and retrieving and returning information.

ISRDASH
Edit macro that deletes all lines that begin with a dash except the first one.

ISRFLAG
ISPF/PDF edit macro to add change flags to a new file based on the differences between the new file and an ancestor of that file.

ISRIMBED
Sample edit macro that builds a list of imbed (.im) statements found in the member that is entered as an operand.

ISRMASK
Sample edit macro that overlays lines with data from a mask line, for example to place a comment area over existing lines.

ISRMBRS
Processes all members of partitioned data set, running a second, user-specified, ISPF edit macro against each member.

ISRONLY
An ISPF Edit macro written in REXX that combines the ISPF Edit commands EXCLUDE and FIND such that only the lines containing the search string are displayed.

ISRSEPC
Version of the macro ISRSLREX written in COBOL. Demonstrates calling edit functions from a COBOL program.

ISRSEPP
Version of the macro ISRSLREX written in PL/I. Demonstrates calling edit functions from a PL/I program.

ISRSLREX
REXX version of an edit macro that separates each line of data with a line of dashes.

ISRTDATA
Edit macro that demonstrates using a loop structure and conditional logic to generate test data.

ISRTDWRI
A version of the sample edit macro ISRTDATA that demonstrates using CLIST WRITE statements as a debugging aid.

ISRTRYIT
Processes another edit macro command and displays the return code. Useful for experimenting with command or assignment statements without actually writing a complete macro.

Appendix C. Accessibility

Accessible publications for this product are offered through IBM Knowledge Center (www.ibm.com/support/knowledgecenter/SSLTBW/welcome).

If you experience difficulty with the accessibility of any z/OS information, send a detailed email message to mhvrcfs@us.ibm.com.

Accessibility features

Accessibility features help users who have physical disabilities such as restricted mobility or limited vision use software products successfully. The accessibility features in z/OS can help users do the following tasks:

- Run assistive technology such as screen readers and screen magnifier software.
- Operate specific or equivalent features by using the keyboard.
- Customize display attributes such as color, contrast, and font size.

Consult assistive technologies

Assistive technology products such as screen readers function with the user interfaces found in z/OS. Consult the product information for the specific assistive technology product that is used to access z/OS interfaces.

Keyboard navigation of the user interface

You can access z/OS user interfaces with TSO/E or ISPF. The following information describes how to use TSO/E and ISPF, including the use of keyboard shortcuts and function keys (PF keys). Each guide includes the default settings for the PF keys.

- *z/OS TSO/E Primer*
- *z/OS TSO/E User's Guide*
- *z/OS ISPF User's Guide Vol I*

Dotted decimal syntax diagrams

Syntax diagrams are provided in dotted decimal format for users who access IBM Knowledge Center with a screen reader. In dotted decimal format, each syntax element is written on a separate line. If two or more syntax elements are always present together (or always absent together), they can appear on the same line because they are considered a single compound syntax element.

Each line starts with a dotted decimal number; for example, 3 or `3.1` or `3.1.1`. To hear these numbers correctly, make sure that the screen reader is set to read out punctuation. All the syntax elements that have the same dotted decimal number (for example, all the syntax elements that have the number `3.1`) are mutually exclusive alternatives. If you hear the lines `3.1 USERID` and `3.1 SYSTEMID`, your syntax can include either `USERID` or `SYSTEMID`, but not both.

The dotted decimal numbering level denotes the level of nesting. For example, if a syntax element with dotted decimal number 3 is followed by a series of syntax elements with dotted decimal number 3.1, all the syntax elements numbered 3.1 are subordinate to the syntax element numbered 3.

Certain words and symbols are used next to the dotted decimal numbers to add information about the syntax elements. Occasionally, these words and symbols might occur at the beginning of the element itself. For ease of identification, if the word or symbol is a part of the syntax element, it is preceded by the backslash (\) character. The * symbol is placed next to a dotted decimal number to indicate that the syntax element repeats. For example, syntax element *FILE with dotted decimal number 3 is given the format 3 * FILE. Format 3* FILE indicates that syntax element FILE repeats. Format 3* * FILE indicates that syntax element * FILE repeats.

Characters such as commas, which are used to separate a string of syntax elements, are shown in the syntax just before the items they separate. These characters can appear on the same line as each item, or on a separate line with the same dotted decimal number as the relevant items. The line can also show another symbol to provide information about the syntax elements. For example, the lines 5.1*, 5.1 LASTRUN, and 5.1 DELETE mean that if you use more than one of the LASTRUN and DELETE syntax elements, the elements must be separated by a comma. If no separator is given, assume that you use a blank to separate each syntax element.

If a syntax element is preceded by the % symbol, it indicates a reference that is defined elsewhere. The string that follows the % symbol is the name of a syntax fragment rather than a literal. For example, the line 2.1 %OP1 means that you must refer to separate syntax fragment OP1.

The following symbols are used next to the dotted decimal numbers.

? indicates an optional syntax element
The question mark (?) symbol indicates an optional syntax element. A dotted decimal number followed by the question mark symbol (?) indicates that all the syntax elements with a corresponding dotted decimal number, and any subordinate syntax elements, are optional. If there is only one syntax element with a dotted decimal number, the ? symbol is displayed on the same line as the syntax element, (for example 5? NOTIFY). If there is more than one syntax element with a dotted decimal number, the ? symbol is displayed on a line by itself, followed by the syntax elements that are optional. For example, if you hear the lines 5 ?, 5 NOTIFY, and 5 UPDATE, you know that the syntax elements NOTIFY and UPDATE are optional. That is, you can choose one or none of them. The ? symbol is equivalent to a bypass line in a railroad diagram.

! indicates a default syntax element
The exclamation mark (!) symbol indicates a default syntax element. A dotted decimal number followed by the ! symbol and a syntax element indicate that the syntax element is the default option for all syntax elements that share the same dotted decimal number. Only one of the syntax elements that share the dotted decimal number can specify the ! symbol. For example, if you hear the lines 2? FILE, 2.1! (KEEP), and 2.1 (DELETE), you know that (KEEP) is the default option for the FILE keyword. In the example, if you include the FILE keyword, but do not specify an option, the default option KEEP is applied. A default option also applies to the next higher dotted decimal number. In this example, if the FILE keyword is omitted, the default FILE(KEEP) is used. However, if you hear the lines 2? FILE, 2.1, 2.1.1! (KEEP), and 2.1.1 (DELETE), the default option KEEP applies only to the next higher dotted decimal number, 2.1 (which does not have an associated keyword), and does not apply to 2? FILE. Nothing is used if the keyword FILE is omitted.

*** indicates an optional syntax element that is repeatable**
The asterisk or glyph (*) symbol indicates a syntax element that can be repeated zero or more times. A dotted decimal number followed by the * symbol indicates that this syntax element can be used zero or more times; that is, it is optional and can be repeated. For example, if you hear the line 5.1* data area, you know that you can include one data area, more than one data area, or no data area. If you hear the lines 3* , 3 HOST, 3 STATE, you know that you can include HOST, STATE, both together, or nothing.

Notes:

1. If a dotted decimal number has an asterisk (*) next to it and there is only one item with that dotted decimal number, you can repeat that same item more than once.
2. If a dotted decimal number has an asterisk next to it and several items have that dotted decimal number, you can use more than one item from the list, but you cannot use the items more than once each. In the previous example, you can write HOST STATE, but you cannot write HOST HOST.

3. The * symbol is equivalent to a loopback line in a railroad syntax diagram.

+ indicates a syntax element that must be included

The plus (+) symbol indicates a syntax element that must be included at least once. A dotted decimal number followed by the + symbol indicates that the syntax element must be included one or more times. That is, it must be included at least once and can be repeated. For example, if you hear the line `6.1+ data area`, you must include at least one data area. If you hear the lines `2+, 2 HOST,` and `2 STATE`, you know that you must include `HOST, STATE,` or both. Similar to the * symbol, the + symbol can repeat a particular item if it is the only item with that dotted decimal number. The + symbol, like the * symbol, is equivalent to a loopback line in a railroad syntax diagram.

Notices

This information was developed for products and services that are offered in the USA or elsewhere.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
United States of America

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

This information could include missing, incorrect, or broken hyperlinks. Hyperlinks are maintained in only the HTML plug-in output for the Knowledge Centers. Use of hyperlinks in other output formats of this information is at your own risk.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Site Counsel
2455 South Road

Poughkeepsie, NY 12601-5400
USA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or

reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM Online Privacy Statement

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user, or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

Depending upon the configurations deployed, this Software Offering may use session cookies that collect each user's name, email address, phone number, or other personally identifiable information for purposes of enhanced user usability and single sign-on configuration. These cookies can be disabled, but disabling them will also eliminate the functionality they enable.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM's Privacy Policy at ibm.com/privacy and IBM's Online Privacy Statement at ibm.com/privacy/details in the section entitled "Cookies, Web Beacons and Other Technologies," and the "IBM Software Products and Software-as-a-Service Privacy Statement" at ibm.com/software/info/product-privacy.

Policy for unsupported hardware

Various z/OS elements, such as DFSMS, JES2, JES3, and MVS, contain code that supports specific hardware servers or devices. In some cases, this device-related element support remains in the product even after the hardware devices pass their announced End of Service date. z/OS may continue to service element code; however, it will not provide service related to unsupported hardware devices. Software problems related to these devices will not be accepted for service, and current service activity will cease if a problem is determined to be associated with out-of-support devices. In such cases, fixes will not be issued.

Minimum supported hardware

The minimum supported hardware for z/OS releases identified in z/OS announcements can subsequently change when service for particular servers or devices is withdrawn. Likewise, the levels of other software products supported on a particular release of z/OS are subject to the service support lifecycle of those products. Therefore, z/OS and its product publications (for example, panels, samples, messages, and product documentation) can include references to hardware and software that is no longer supported.

- For information about software support lifecycle, see: IBM Lifecycle Support for z/OS (www.ibm.com/software/support/systemsz/lifecycle)
- For information about currently-supported IBM hardware, contact your IBM representative.

Programming Interface Information

This publication primarily documents information that is NOT intended to be used as Programming Interfaces of ISPF.

This publication also documents intended Programming Interfaces that allow the customer to write programs to obtain the services of ISPF. This information is identified where it occurs, either by an introductory statement to a chapter or section or by the following marking:

```
+---------------------Programming Interface information----------------------+

+------------------End of Programming Interface information------------------+
```

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at Copyright and Trademark information (www.ibm.com/legal/copytrade.shtml).

Index

Special Characters

! (exclamation point), for implicit edit macro 107
.ZCSR 59, 104
.ZDEST 104, 108
.ZFIRST 59, 104
.ZFRANGE 104, 108
.ZLAST 59, 104
.ZLRANGE 104, 108
((column shift left), line command 138
) (column shift right), line command 140
& prefix for edit commands 14
&LASTCC variable 110
< (data shift left), line command 142
> (data shift right), line command 144

Numerics

3850 virtual volumes, accessing 7

A

A (after), line command 146
A operand, REXX TRACE statement 114
abbreviations for commands and other values 429
accessibility
    contact IBM 435
    features 435
ACCOUNT command 7
add a data set member 391
adding
    a line 162, 357
    edit macro command 87
    models 72
adding data 271
after, line command 146
AK (after, multiple targets), line command 146
alias name, defining with edit macro 106
alias, assigning 222, 327
application-wide macros 25
ASCII
    data, working with 50
    linefeed character
        LF macro command 351
        LF primary command 51, 248
    SOURCE primary command 50
ASCII, translating 284, 412
assignment statement
    AUTOLIST 300
    AUTONUM 301
    AUTOSAVE 303
    BLKSIZE 304
    BOUNDS 305
    CAPS 309
    CHANGE COUNT 313
    CURSOR 320
    DATA_CHANGED 323
assignment statement *(continued)*
    DATA_WIDTH 324
    DATAID 325
    DATASET 326
    description 96
    DISPLAY_COLS 330
    DISPLAY_LINES 331
    EXCLUDE_COUNTS 337
    FIND_COUNTS 340
    FLIP 341
    FLOW_COUNTS 342
    HEX 342
    HIDE 241
    how to use 98
    IMACRO 349
    LABEL 104, 352
    LEVEL 354
    LINE 355
    LINE_AFTER 357
    LINE_BEFORE 358
    LINENUM 362
    LRECL 365
    MACRO_LEVEL 103, 367, 368
    MASKLINE 368
    MEMBER 370
    NOTES 374
    NULLS 375
    NUMBER 376
    PACK 379
    parentheses guidelines 98
    PROFILE 384
    RANGE_CMD 108, 386
    RECFM 387
    RECOVERY 389
    reference section 295
    RMACRO 109, 396
    SCAN 95, 400
    SEEK_COUNTS 403
    STATS 412
    summary 295
    TABS 415
    TABSLINE 416
    USER_STATE 423
    VERSION 424
    XSTATUS 426
assistive technologies 435
attribute bytes, used with tabs 65
AUTOLIST
    assignment statement 301
    macro command 300
    primary command 196
autolist mode
    defined 19
    querying the value 300
    setting the value 196, 300
automatic generation of source listing 196, 300
automatic saving of data 199, 303

AUTONUM
assignment statement 301
macro command 301
primary command 19, 198
autonum mode 19
AUTOSAVE
assignment statement 303
macro command 303
primary command 19, 199
autosave mode, defined 19

B

B (before), line command 41, 149
batch processing, submitting data for 285, 414
batch processing, using edit macros in 102
batch, ending a macro 370
before, line command 149
beginning an edit session 4
BK (before, multiple targets), line command 149
BLKSIZE, assignment statement 304
block size, retrieving 304
boundaries
controlling 201, 305
default 24
definition line 24
setting 151
BOUNDS
assignment statement 305, 306
line command 151
macro command 305, 306
primary command 201
BROWSE
macro command 307
primary command 202
built-in command
disabling 222, 327
processing 203
built-in labels 59
BUILTIN
macro command 308
primary command 203

C

C (copy), line command
description 153
used with CREATE command 218
used with REPLACE command 273
CANCEL
macro command 308, 309
primary command 203
canceling edit changes 203, 308
CAPS
assignment statement 309, 310
DBCS data 205
macro command 309, 310
primary command 19, 204, 208
caps mode
defined 19
overview 20
querying the value 309
setting the value 204, 309
CHANGE
macro command
column-dependent data, defined 50
DBCS data 50
description 310, 312
EBCDIC data 50
RCHANGE command 387
saving and restoring values 423
primary command
column-dependent data, defined 50
DBCS data 50
description 44, 205, 207
EBCDIC data 50
qualifying search strings 53
specifying search strings 45
repeating 55
change a data string 310
CHANGE command, repeating 267
change count, retrieving 313
CHANGE_COUNTS, assignment statement 313
changed lines 22
changing a data string 205
changing data 44
changing models 75
character encoding 284, 412
character string
changing 205
finding 234, 338
how to use 46
specifying 45
characters
converting 204, 309
converting to lowercase 165
converting to uppercase 188
displaying hexadecimal 238, 342
clipboard, pasting lines from 263
CLIST CONTROL statements 114
CLIST edit macro statements 79, 87
CLIST WRITE statements 113
COBOL sequence field, defined 27
color in editor, changing 242
COLS
primary command 208
COLS, line command 155
column
shift left 138
shift right 140
column identification line, displaying 155
column limitations 54
column positions, referring to 106
column shifting
DBCS data 43
destructive 43
line command 43
columns
identifying 155
line command 155
query display 330
shift left 406
shift right 407
columns indicator line, displaying 208
command line 13, 193
command names, overriding 106
command procedure statements 88

command scan mode, setting the value 400
command, PROFILE RESET 21
command, querying 386
commands
  reading syntax diagrams xxi
commands, reversing last edit 288
compare command 314
COMPARE command examples 212
compare command return codes 316
compare command syntax 314
COMPARE command syntax 209
Compare, edit command 314
COMPARE, edit command 209
compress data 379
compressing data 262
CONLIST operand, CLIST CONTROL statement 114
contact
  z/OS 435
control and display your profile 384
control edit recovery 268, 389
control null spaces 375
control version number 292, 424
CONTROL, ISPEXEC statement 111
controlling and displaying profile 264
controlling null spaces 259
controlling the edit boundaries 201, 305
controlling the edit environment 17
controlling the search for a data string 52
convert characters to lowercase 165
converting characters 204, 309
converting note lines to data 171
COPY
  macro command 317
  primary command
    description 212, 214
    how to use 41
copy a model into the current data set 370
copying a model into data set 251
copying data
  into the current data set 41
  lines of data 153
  macro command 317
  primary command 212
  using edit macro 99
CREATE
  macro command 318, 319
  primary command
    description 217, 218
    how to use 41
creating
  a data set member 217, 318
  data 41
  new data 9
current member name, querying 370
cursor position
  querying the value 320
  setting the value 320
cursor values, saving and restoring 423
CURSOR, assignment statement
  positioning cursor on command line 321
Cut and Save Lines 322
Cut Macro command 322
CUT Primary command 220
cutting and saving lines 220

D

D (delete) line command 157
data
  adding 271
  canceling changes 203, 308
  changing 44, 205, 310
  column-dependent, defined 50
  compressing 262, 379
  controlling the string search 52
  converting data 188
  copying 41, 212, 317
  copying lines 153
  creating 41
  creating new 9
  DBCS considerations 50
  deleting 224, 329
  description 207
  EBCDIC considerations 50
  editing existing 9
  excluding 44, 232, 335
  finding 44, 234, 338
  inserting 350
  managing 41
  moving 41
  packing 16
  realigning, LF primary command 248
  replacing 41, 271
  retrieving the changed status 323
  retrieving the ID 325
  retrieving the width 324
  saving automatically 199, 303
  saving the current 278, 397
  seek a data string 401
  shift left 142, 408
  shift right 144, 409
  shifting 42, 44
  sorting 282, 409
  split a line 420
  submitting for batch processing 285, 414
  test flow a paragraph 419
data field, defined 376
data in controlled libraries, editing 16
data lines, referring to 105
data modes 20
data set
  adding a member 391
  copying a model into 251, 370
  creating a member 217, 318
  creating a new 9
  editing a member 226, 333
  editing existing 9
  generating statistics 285, 412
  moving a member 254, 372
  password specification 7
  renumbering lines automatically 269, 390
  replacing a member 391
  retrieving the current name 326
  security 7
DATA_CHANGED, assignment statement 323
DATA_WIDTH, assignment statement 324
data-changed status, retrieving 323
DATAID, assignment statement 325
DATASET, assignment statement 326

DBCS data
CHANGE command 50
column shifting 43
display boundary 4
hardware tabs 65
SORT command 284, 411
TE (text entry) line command 63
TF (text flow) 62
TS (text split) line command 62
DDNames 87
debugging edit macros 113
debugging edit macros with ISREMSPY 116
DEFINE
edit macro command 90, 106
macro command 327
primary command 222
define tabs mode 415
defining
a name 222, 327
an alias for a command 106
an edit profile 17
defining macros
implicit 107
overriding command names 106
resetting definitions 106
scope of definitions 106
using an alias 106
defining tabs mode 286
DELETE
macro command 329
primary command 224
deleting
edit macro labels 105
labels 59
lines 157, 224
models 76
delimited string 45, 46
destination, specifying 108
destructive shift, defined 43
dialog development models 69
dialog service errors, debugging 113
dialog service requests 88
dialog variable name, defined 96
direction of the search 52
disabling a command 106
disabling a macro or built-in command 222, 327
display and control your profile 384
display boundary, DBCS data 4
display columns 330
display model notes 374
DISPLAY_COLS, assignment statement 330
DISPLAY_LINES, assignment statement 331
displaying an edit profile 17
displaying and controlling profile 264
displaying hexadecimal characters 238, 342
displaying model notes 258
displaying the Edit Settings panel 228
distributed editing 4
DOWN, macro command 332
duplicating lines 176

E

EBCDIC data 50
edit
beginning a session 4
canceling changes 203, 308
column shifting 43
command reference section 193
command summary 13
considerations 15
controlling the boundaries 201, 305
controlling the environment 17
controlling the recovery 389
copying data 41
creating data 41
data display panel 9
displaying processed commands 14
editing data in controlled libraries 16
ending a session 12
entry panel 8
excluding lines 58
introduction to 3, 11
line command macros 16
line commands 13
macro command 15, 333
managing data 41
models 69
modes 19
moving data 41
number mode 27
option 2 4
primary command
description 226
syntax 226
primary commands, description 13
profiles 17
recovery, controlling 268
recursive 226, 333
replacing data 41
rules for entering line commands 135
selecting the editor 4
sequence number display 27
sequence number format 26
sequence numbers 26
shifting columns 43
shifting data 42, 44
splitting text 61
text entry 61
text flow 61
undisplayable characters 11
undoing edit interactions 66
word processing 61
EDIT
primary command
description 226
example 227
Edit - Entry panel 8
edit a member 333
Edit and View Settings panel 228
edit assignment statements
elements
keyphrase 96
overlays 97
value 96
how to use 98
manipulating data 99
Edit command errors, debugging 113

edit commands and PF key processing 14
edit compare command 314
edit COMPARE command 209
Edit data display panel 9
edit macro
  alias name 106
  assignment statements 87, 96
  CLIST macro, differences from program macros 90
  column positions, referring to 106
  command procedure statements 88
  command summary 15
  commands 87
  creating 87
  data lines, referring to 105
  defining 106
  definition of 3
  description 79
  dialog service requests 88
  identifying 366
  implicit definition using an exclamation point 107
  initial macro 24
  introduction to 79
  ISRBOX macro 119
  ISRCHGS macro 126
  ISRIMBED macro 121
  ISRMASK macro 130
  ISRMBRS macro 124
  labels
    description 103
    editor-assigned 103
    passing 105
    referring to 105
    using 104
  levels 103
  line command functions, how to perform 100
  messages 102
  naming 95
  NOPROCESS operand 107
  parameters 101
  PROCESS command and operand 107
  program macro
    description 89
    differences from CLIST macros 90
    differences from REXX macros 90
    parameter passing 90
    running 94
    writing 91
  recovery macro 109
  reference section 295
  replacing built-in edit commands 106
  resetting a command to previous status 106
  return codes 109
  REXX macro, differences from program macros 90
  samples 119
  testing
    CLIST CONTROL statements 114
    CLIST WRITE statements 113
    description 113
    experimenting with edit macro commands 115
    return codes 111
    REXX SAY statements 113
    REXX TRACE statements 114
  TSO commands 89
  using 79
edit macro *(continued)*
  variable substitution 95
  variables 95
edit macros, debugging with ISREMSPY 116
Edit mode defaults 20
edit processing of PF keys 14
edit profile
  autolist mode 196
  autonum mode 198, 301
  autosave mode 199, 303
  boundary settings 151
  caps mode 204
  control and display 384
  controlling and displaying 264
  defaults 20, 21
  defining 17
  definition of 17
  displaying 17
  initial macro 246, 349
  lock 384
  locking 264
  modifying 18
  naming 17
  note mode 258
  nulls mode 259
  profile name 17
  recovery macro 278
  saving and restoring 423
  specifying 7
  tabs mode 286
  types 17
Edit Profile Initialization, Site-wide 21
edit profile name, definition 17
edit profiles, locking 19
edit recovery
  Edit Recovery panel 38
  turning off 39
  turning on 38
edit session, ending 231, 334
Edit Settings panel, displaying 228
edit, distributed 4
editing a member 226
editing existing data 9
editor-assigned labels 59
editor, ISPF 3
EDITSET primary command 228
EDSET primary command 228
eliminating labels 59
END
  macro command 334
  primary command 231
end a macro 370
END command 199
end the edit session 334
ending an edit session 12, 231
entering text, text entry command 181
error codes for severe errors 110
error lines 22
EXCLUDE
  macro command 335
  primary command
    description 44, 232, 233
    qualifying search strings 53
    specifying search strings 45

EXCLUDE *(continued)*
repeating 55
exclude counts, querying the value 337
exclude status, reversing 236
EXCLUDE_COUNTS, assignment statement 337
excluded line limitations 54
excluded line messages
hiding 241
excluded lines
hiding 344
line status, set or query 426
redisplaying 58, 275
excluding a line 58, 189, 335
excluding data 44
explicit shifts, defined 42
extent of a search 52

F

F (show first line), line command 158
feedback xxvii
FIND
macro command
description 338, 339
RFIND command 277, 395
saving and restoring values 423
when to use instead of SEEK 402
primary command
description 44, 234, 235
qualifying search strings 53
specifying search strings 45
repeating 55
find counts, querying the value 340
FIND_COUNTS, assignment statement 340
finding a data string 234
finding a search string 338
finding data 44
finding lines, LOCATE primary command 249
finding models 75
first line, showing 158
flagged lines
changed lines 22
error lines 22
special lines 22
FLIP
assignment statement 341
definition 59
macro command 341
primary command 236
flow counts, querying the value 342
FLOW_COUNTS, assignment statement 342
Format Name field 8
formatted edit mode, defined 170
formatting input 368
fragments, syntax diagrams xxi

G

generate sequence numbers 376
generating data set statistics 285, 412
generating sequence numbers 260
guidelines for using the editor 15

H

Hardware Tab field, defined 65
hardware tabs
DBCS data 65
defining 65
description 63
fields, how to use 65
HEX
assignment statement 342
macro command 342
primary command 19, 238
hexadecimal characters
displaying 238, 342
format 19
mode 342
showing individual records in 160
string 45
HIDE
assignment statement 241
macro command 241, 344
primary command 241
hiding lines, EXCLUDE primary command 232
HILITE
macro command
description 344, 348
how to use 345
primary command
description 246
how to use 242
HILITE function description 27
HX (lowercase), line command 160

I

I (insert) line command 162
I operand, REXX TRACE statement 114
identify an edit macro 366
identifying columns 155
IMACRO
assignment statement 349
macro command 349
primary command 19, 246
implicit macro definition 107
implicit shifts, defined 42
initial macro, specifying 246, 349
initial macros
DEFINE commands used in 106
specifying in the EDIT service call 24
specifying on the Edit - Entry panel 24
starting 24
Initialization, Site-wide Edit Profile 21
INSERT, macro command 350
inserting
data 350
lines 162
interactive column numbers 106
introduction to edit macros 79
ISPEXEC 88
ISPF list data set 196, 300
ISPF Workstation Tool Integration dialog 4
ISPF, definition 3
ISRBLOCK, sample macro 433
ISRBOX, sample macro 119, 433

ISRCHGS, sample macro 126, 433
ISRCOUNT, sample macro 83, 433
ISRDASH, sample macro 80, 433
ISREDIT service 90
ISREDIT statements 87, 100
ISREMSPY 116
ISREMSPY, sample macro 433
ISRFLAG, sample macro 433
ISRIMBED, sample macro 121, 433
ISRMASK, sample macro 130, 433
ISRMBRS, sample macro 124, 433
ISRONLY, sample macro 433
ISRSEPC, sample macro 94, 433
ISRSEPP, sample macro 433
ISRSETLN, edit macro sample 400
ISRSLPLI, sample macro 93
ISRSLREX, sample macro 92, 433
ISRTDATA, sample macro 81, 433
ISRTDWRI, sample macro 114, 433
ISRTRYIT, sample macro 115, 433

K

keeping an edit command on the command line 14
keyboard
    navigation 435
    PF keys 435
    shortcut keys 435
keyphrase, defined 96
keywords, syntax diagrams xxi
kinds of search strings 45

L

L (show last line), line command 164
L operand, REXX TRACE statement 114
LABEL
    assignment statement
        description 352
        overview 104
    querying the value 352
    setting the value 352
labeled line, querying 362
labels
    defined 59
    deleting 59
    editor-assigned 59
    eliminating 59
    in macro commands 59
    specifying a range 60
labels in edit macros
    deleting 105
    description 103
    editor-assigned 103
    how to use 104
    levels 103
    nested macros 105
    passing 105
    referring to 105
languages for edit macros 79, 87
last line, showing 164
LC (lowercase), line command 165
left
left *(continued)*
    scroll 353
    shift columns 406
    shift data 408
LEFT
    macro command 353
LEVEL
    assignment statement 354
    macro command 354
    primary command 247
level number, specifying 247, 354
limiting the SORT command 283, 411
LINE
    adding 358
    assignment statement 355
    querying the number 355
    querying the value 355
    setting the value 355
line command field
    resetting 45
line command functions in edit macros 100
line command macros in edit 16
line commands
    ((column shift left) 138
) (column shift right) 140
    < (data shift left) 142
    > (data shift right) 144
    A, AK (after) 146
    B (before) 149, 150
    B, BK (before) 149
    BOUNDS 151
    C (copy) 153
    COLS 155
    D (delete) 157
    description 135
    F (show first line) 158
    HX (lowercase) 160
    I (insert) 162
    L (show last line) 164
    LC (lowercase) 165
    M (move) 167
    MASK 169
    MD (make dataline) 171
    O (overlay) 173
    OK (overlay, multiple targets) 173
    R (repeat) 176
    rules for entering 135
    S (show line) 58, 178
    summary 136
    TABS 180
    TE (text entry) 61, 63, 181
    TF (text flow) 61, 184
    TS (text split) 61, 186
    UC (uppercase) 188
    usage 13
    X (exclude) 54, 58, 189
line label
    querying the value 352
    setting the value 352
line number, ordinal 249
line numbers
    restoring 275
line pointer
    COPY macro command 317

line pointer *(continued)*
CREATE macro command 318
CURSOR assignment statement 320
DELETE macro command 329
incomplete 319
INSERT macro command 350
invalid 318, 373
LABEL assignment statement 352
LINE assignment statement 355
LINE_AFTERassignment statement 357
LINE_BEFORE assignment statement 359
LOCATE macro command 363
MASKLINE assignment statement 369
MODEL macro command 371
MOVE macro command 373
referring to labels 105
SHIFT (macro command 406
SHIFT) macro command 407, 408
SHIFT > macro command 409
TABSLINE assignment statement 417
TENTER macro command 418
TFLOW macro command 419
TSPLIT macro command 420
XSTATUS assignment statement 427
line pointer range
CREATE macro command 319, 322, 329, 364, 392, 394
DELETE macro command 329, 364
LOCATE macro command 364
SUBMIT macro command 414
line range 60
LINE_AFTER, assignment statement 357
LINE_BEFORE, assignment statement 358
LINE_STATUS 360
linefeed character
LF macro command 351
LF primary command 51, 52, 248
LINENUM, assignment statement 362
lines
adding 162
copying 153
deleting 157, 329
exclude status 426
excluded limitations 54
excluding 58, 232, 335
inserting 162
locating 249, 363
moving 167
numbering automatically 198
overlaying 173
query display 331
renumbering automatically 269, 390
repeating 176
showing 178
showing the first 158
showing the last 164
specifying ranges 59
splitting 62, 420
literal character string, defined 96
LOCATE
macro command
generic syntax 363
specific syntax 363
primary command
generic syntax 250
LOCATE *(continued)*
primary command *(continued)*
specific syntax 249
locate lines 363
locating lines, LOCATE primary command 249
lock your profile 384
locking an edit profile 19
locking your profile 264
logical record length, querying 365
logical tabs, description 64
lowercase, converting to 165
lptr
COPY macro command 317
CURSOR assignment statement 320
DELETE macro command 329
incomplete 319
INSERT macro command 350
invalid 318, 373
LABEL assignment statement 352
LINE assignment statement 355
LINE_AFTER assignment statement 357
LINE_BEFORE assignment statement 359
LOCATE macro command 363
MASKLINE assignment statement 369
MODEL macro command 371
MOVE macro command 373
referring to labels 105
SHIFT (macro command 406
SHIFT) macro command 407, 408
SHIFT > macro command 409
TABSLINE assignment statement 417
TENTER macro command 418
TFLOW macro command 419
TSPLIT macro command 420
XSTATUS assignment statement 427
lptr-range
CREATE macro command 319, 322, 329, 364, 392, 394
DELETE macro command 329, 364
LOCATE macro command 364
LRECL, assignment statement 365

M

M (move), line command
description 167
used with CREATE command 218
used with REPLACE command 273
macro
ending in batch 370
specifying a recovery 278, 396
specifying an initial 246, 349
Macro command profile reset syntax 385
macro commands
abbreviations 429
assignment statements 96
AUTOLIST 300
AUTONUM 301
AUTOSAVE 303
BOUNDS 305
BROWSE 307
BUILTIN 308
CANCEL 308
CAPS 309
CHANGE 310

macro commands *(continued)*
COPY 317
CREATE 318
CUT 322
DEFINE 327
DELETE 329
disabling 222, 327
DOWN 332
EDIT 333
END 334
EXCLUDE 335
FIND 338
FLIP 341
HEX 342
HIDE 241
HILITE 345
identifying 222, 327
IMACRO 349
INSERT 350
introduction to 79
labels 59
LEFT 353
LEVEL 354
LF 351
LOCATE 363
MACRO 366
MEND 370
MODEL 370
MOVE 372
NONUMBER 373
NOTES 374
NULLS 375
NUMBER 376
PACK 379
PASTE 380
PROCESS 383
PROFILE 384
RCHANGE 267, 387
RECOVERY 389
reference section 295
RENUM 390
REPLACE 391
RESET 393
RFIND 277, 395
RIGHT 395
RMACRO 109, 396
SAVE 397
SCAN 400
SEEK 44, 401
SETUNDO 404
SHIFT (406
SHIFT) 407
SHIFT < 408
SHIFT > 409
SORT 409
SOURCE 412
STATS 412
SUBMIT 414
summary 295
TABS 415
TENTER 61, 418
TFLOW 61, 419
TSPLIT 61, 420
UNNUMBER 421
macro commands *(continued)*
UP 421
usage 15
VERSION 424
VIEW 425
macro definitions, resetting 106
macro nesting level
querying 367, 368
retrieving 103
MACRO_LEVEL, assignment statement 105, 367, 368
MACRO, macro command 366
macros, sample 433
managing data 41
mask line, set or query 368
mask, defining 169
MASK, line command 169
MASKLINE, assignment statement
description 368, 369
overlays 97
using 98
MD (make dataline), line command 171
member name, querying 370
MEMBER, assignment statement 370
member, editing 226, 333
MEND, macro command 370
messages, displayed from edit macros 82, 102
mixed data, used with data strings 89
Mixed Mode field 8
model
adding 72
changing 72, 75
class, defined 69
copying into data set 251
copying into the current data set 370
deleting 72, 76
edit, defined 69
finding 72, 75
hierarchy 69
kinds 69
locating 75
logical name 69
macro command 370
name, defined 70
primary command 251
qualifier, defined 70
using 70
model notes, displaying 258, 374
model selection panels 71
modes, edit 19, 20
modification flag 249
modification level number, specifying 247, 354
modification level, description 26
modifying an edit profile 18
MOUNT authority 7
MOVE
macro command 372
primary command 41, 254
move a data set member 372
moving a data set member 254
moving a line of data in an edit macro 100
moving data into the current data set 41
moving lines 167
multiple parameters in an edit macro 101

N

name, defining 222, 327
naming edit macros 95
navigation
  keyboard 435
nested macros, starting 103
nesting level, querying 367, 368
NOCONLIST operand, CLIST CONTROL statement 115
NOLIST operand, CLIST CONTROL statement 115
non-destructive shifting, defined 44
NONUMBER
  macro command 373
  primary command 258
NOPROCESS 107
normal, defined for stats mode 25
NOSYMLIST operand, CLIST CONTROL statement 115
note lines, converting to data 171
note mode
  description of 19
  querying the value 374
  setting the value 258, 374
NOTES
  assignment statement 374
  macro command 374
  primary command 19, 258
notes, displaying model 258, 374
null spaces, controlling 259, 375
NULLS
  assignment statement 375
  macro command 375
  primary command 19, 259
nulls mode
  description of 19
  querying the value 375
  setting the value 259, 375
NUMBER
  assignment statement 376
  macro command 376
  primary command
    description 19, 260
    DISPLAY operand 27
number mode
  defined 20
  description 19, 260
  initializing 27
  setting, edit 26
  turning off 258, 373
  used with RENUM command 269, 390
number, specifying the modification level 247, 354
numbering lines automatically 198, 301
numbers
  controlling version 292, 424
  generating sequence 260, 376
  modification level 26
  remove sequence 421
  removing sequence 290
  sequence 26
  turning off number mode 258, 373

O

O (overlay), line command 173
O operand, REXX TRACE statement 114
OK (overlay, multiple targets), line command 173
ordinal line number 249
overlaying lines 173
overlays, guidelines on how to perform 97
overriding, built-in edit commands 106

P

PACK
  assignment statement 379
  macro command 379
  primary command 19, 262
pack mode 19, 20, 262
packing data, edit 16
panel
  excluding lines 189
  process the 383
  resetting the 393
  set up for text entry 418
panel data, resetting 275
panel values, saving and restoring 423
panels
  Edit data display 9
  Edit Entry 6, 227
  edit profile display 17, 266
  Edit Recovery 38
  model selection 71
parameters in an edit macro 101
passing labels 105
passing parameters to an edit macro
  description 101
  multiple 101
  processing an Edit command 90
  program macros 90
password protection 7
Paste Lines 380
Paste Macro command 380
PASTE primary command 263
pasting lines 263
pathnames, specifying for z/OS UNIX files 16
PDF, defined 3
PF key processing in edit 14
PF keys, scroll commands 12
picture string 45, 46
power typing, defined 63
prepare display for data insertion 350
PRESERVE command 12, 264
PRESERVE macro 381
primary commands
  abbreviations 429
  AUTOLIST 19, 196
  AUTONUM 19, 198
  AUTOSAVE 19, 199
  BOUNDS 201
  BROWSE 202
  BUILTIN 203
  CANCEL 203
  CAPS 19, 204
  CHANGE 44, 205
  COPY 41, 212
  CREATE 41, 217
  CUT 220
  DEFINE 222
  DELETE 224

primary commands *(continued)*
displaying after processing 14
EDIT 226
END 231
EXCLUDE 44, 232
FIND 44, 234
FLIP 59, 236
HEX 19, 238
HIDE 241
HILITE 242
IMACRO 19, 246
LEVEL 247
LF 51, 52, 248
LOCATE 249
MODEL 251
MOVE 41, 254
NONUMBER 258
NOTES 19, 258
NULLS 19, 259
NUMBER 19, 260
PACK 19, 262
PASTE 263
PROFILE 19, 264
RECOVERY 19, 268
reference section 193
RENUM 269
REPLACE 41, 271
RESET 59, 275
RMACRO 278
SAVE 278
SETUNDO 20, 280
SORT 282
SOURCE 50, 284
STATS 20, 285
SUBMIT 285
summary 193
TABS 20, 286
UNDO 288
UNNUMBER 290
usage 13
VERSION 292
VIEW 294
PROCESS command and operand 107
PROCESS, macro command
description 383
used with RANGE_CMD assignment statement 386
processing built-in commands 203, 308
PROFILE
assignment statement 384
macro command
description 384
profile control syntax 384
profile lock syntax 385
primary command
description 19, 266
display or define a profile 17
profile control syntax 265
profile lock syntax 265
profile defaults 20, 21
PROFILE RESET command 21
Profile Reset syntax 265
Profile reset syntax, macro command 385
profile, edit
autolist mode 196, 370
profile, edit *(continued)*
autonum mode 198, 301
autosave mode 199, 303
boundaries 201
boundary settings 151
caps mode 204
control and display 384
controlling and displaying 264
defining 17
description 17
displaying 17
initial macro 246, 349
lock 384
locking 19, 264
modifying 18
note mode 258
nullsmode 259
recovery macro 278
saving and restoring 423
tabs mode 286
types 17
program macros
differences from CLISTs 90
differences from REXX EXECs 90
how to write 91
implicit definition 107
passing parameters 90
running 94

Q

qualifying the search string 53
query
a line 355
autolist mode 300
autonum mode 301
autosave mode 303
block size 304
caps mode 309
change count 313
command entered 386
current member name 370
cursor position 320
data ID 325
data set name 326
data width 324
data-changed status 323
display columns 330
display lines 331
edit boundaries 305
edit profile 384
exclude counts 337
exclude status for a line 426
find counts 340
flow counts 342
hexadecimal mode 342
initial macro 349
line label 352
line number 362
logical record length 365
macro nesting level 367, 368
mask line 368
modification level number 354
note mode 374

query *(continued)*
nulls mode 375
number mode 376
pack mode 379
record format 387
recovery mode 389
seek counts 403
tabs line 416
tabs mode 415
version number 424
Query Source and Change Information for a Line in a Data Set, LINE_STATUS 360
Query Volume Information 426

R

R (repeat) line command 176
R operand, REXX TRACE statement 114
range
specifying 108
using labels to specify 60
RANGE_CMD, assignment statement
description 108, 386
used with the PROCESS command 386
RC variable 110
RCHANGE, macro command
description 267, 387
used to repeat CHANGE command 55
realigning data, LF primary command 248
RECFM, assignment statement 387
record format, query 387
recovery
controlling edit 389
edit 38
macro 109, 396
macro, saving the name of 278
mode 19, 389
of data after system failure 268
RECOVERY
assignment statement 389
macro command 389
primary command 19, 268
recursive editing, defined 226, 333
redisplaying excluded lines 58
referring to column positions 106
referring to data lines 105
reformatting a paragraph 184
regular expression 46
regular expressions 48
relative line number of cursor, setting or retrieving 320
relative line numbers 106
remove sequence numbers 421
removing lines 224, 329
removing sequence numbers 290
RENUM
macro command 390
primary command 269
RENUMBER primary command, DISPLAY operand 27
renumbering lines automatically 269, 390
repeatable items, syntax diagrams xxi
repeating a change 267, 387
repeating a search
RCHANGE command, Edit 55
RFIND command, Edit 55
repeating lines 176
REPLACE
macro command 391
primary command
description 271, 272
how to use 41
replace a data set member 391
replacing
data 41, 271
lines 99
RESET
macro command 393
primary command 275
RESET command, PROFILE 21
reset the data display 393
resetting macro definitions 106
resetting the data panel 275
resetting the line command field 45
retrieving the change count 313
retrieving the data ID 325
retrieving the data set name 326
retrieving the data width 324
retrieving the data-changed status 323
return codes
&LASTCC variable 110
0 to 20 109
above 20 110
ISPF editor 110
RC variable 110
reversing exclude status of data 236
reversing last data change 288
REXX edit macro statements 79, 87
REXX SAY statements, using to debug edit macros 113
REXX TRACE statements, using to debug edit macros 114
RFIND command
description 277, 395
used to repeat FIND and EXCLUDE commands 55
RIGHT
macro command 395
scroll 395
RMACRO
assignment statement
description 396
overview 109
macro command 396
primary command
description 278
overview 109

S

S (show line), line command
description 178
redisplaying excluded lines 58
S operand, REXX TRACE statement 114
sample edit macros 119
SAVE
macro command 397, 398
primary command 278
save data automatically 303
save the current data 397
SAVE_LENGTH command 399
saving and restoring
CHANGE macro command values 423

saving and restoring *(continued)*
cursor and panel values 423
edit profile 423
FIND macro command values 423
saving current data 278
saving data automatically 199
SCAN
assignment statement 400
macro command 400
SCAN assignment statement 95
scope of macro definitions 106
scroll
down 332
left 353
right 395
up 421
using PF keys 12
search
controlling 52
DBCS search string, delimiting 45
extent 52
qualifying 53
starting point and direction 52
search string, finding 277
search strings
character 45
delimited 45
finding 338
hexadecimal 45
picture 45
simple 45
security, data set 7
seek a data string 401
seek counts, query 403
SEEK_COUNTS, assignment statement 403
SEEK, macro command
description 44, 401, 402
when to use instead of FIND 339
sending to IBM
reader comments xxvii
sequence numbers
display 27
format 26
generating 260, 376
initializing 27
setting, edit 26
set
a line 355
autolist mode 300
autonum mode 301
autosave mode 303
caps mode 309
command scan mode 400
cursor position 320
edit boundaries 201, 305
edit profile 384
exclude status for a line 426
hexadecimal mode 342
initial macro 349
line label 352
mask line 368
modification level number 354
note mode 374
nulls mode 259, 375
set *(continued)*
number mode 376
pack mode 379
recovery mode 389
tabs line 416
tabs mode 286, 415
version number 424
setting
mask 169
setting the edit boundaries 201, 305
SETUNDO
macro command 404
primary command 66, 280
SETUNDO command 280
SHIFT (, macro command 406
SHIFT), macro command 407
SHIFT <, macro command 408
SHIFT >, macro command 409
shift columns
left 406
right 407
shift data
left 408
right 409
shifting data
edit
columns 43
explicit 42
implicit 42
non-destructive 44
shortcut keys 435
showing first line 158
showing last line 164
showing lines 178
SI characters, delimiting a search 45
simple editing 11
simple string 45, 46
Site-wide Edit Profile Initialization 21
site-wide macro 15
SO characters, delimiting a search 45
software tab field, defined 181
software tabs
defining 64
description 63
fields, how to use 181
SORT
macro command
DBCS data 411
description 409, 410
limiting 411
without operands 410
primary command
DBCS data 284
description 282, 283
limiting 283
without operands 283
sorting data 282, 409
source listing, create 196, 300
spaces, controlling null 259, 375
special lines 22
specify a recovery macro 109, 396
specifying
an initial macro 15, 24, 349
the level number 354

specifying a recovery macro 278
split screen, searching within 54
splitting a line of text 186
splitting lines 62
splitting text 61
standard sequence field, defined 26
starting point of a search 52
statistics
    creation and maintenance of 25
    generating for a data set 285, 412
STATS
    assignment statement 412
    macro command 412
    primary command 20, 285
stats mode 20, 25
strings, kinds of search
    character 45
    delimited 45
    hexadecimal 45
    picture 45
    simple 45
SUBMIT
    macro command 414
    primary command 285
submit data for batch processing 414
submitting data for batch processing 285
summary of changes xxix
SYMLIST operand, CLIST CONTROL statement 114
syntax diagrams, how to read xxi
Syntax, macro command profile reset 385
syntax, Profile Reset 265

T

TABS
    assignment statement 415
    controlling and querying 64, 415
    line command
        defining hardware tabs 65
        defining software tabs 64
        description 180
        limiting hardware tab columns 65
        using software tab fields 181
    macro command 415
    primary command 20, 286
tabs line
    querying the value 416
    setting the value 416
tabs mode
    description 20, 64
    setting the value 286, 415
TABSLINE, assignment statement 416
TE (text entry), line command
    DBCS data, using a DBCS terminal 63
    description 63, 181, 182
    example 182
    syntax 181
template (overlay)
    definition 98
    how to design 98
TENTER, macro command 418
text entry
    in word processing 61
    setting up the panel 418
text entry *(continued)*
    TE line command 181
text flow 61
text flowing a paragraph 184, 419
text split a line 420
TF (text flow), line command
    DBCS data, using a DBCS terminal 62
    description 61, 184, 185
TFLOW, macro command 419
trademarks 442
trailing blanks, saving 264
TS (text split), line command
    DBCS data 62
    description 186
TSO commands in edit macros 89
TSPLIT, macro command 420
turn off number mode 373
turning off number mode 258

U

UC (uppercase), line command 188
undisplayable characters 11
UNDO
    primary command 288
    SETUNDO requirement 404
    with SETUNDO macro 280
undoing edit interactions
    description 288
    how to use 66
    UNDO primary command 288
UNDOSIZE 66
UNIX
    specifying pathnames 16
UNIX files
    copying and moving data 41
    creating and replacing data 41
UNNUMBER
    macro command 421
    primary command 290
UP, macro command 421
uppercase, converting data to 188
user interface
    ISPF 435
    TSO/E 435
USER_STATE, assignment statement 423
using the ISPF editor 3
UTF-8
    data, working with 51
    linefeed character
        LF primary command 52

V

value portion of an edit macro statement 96
variable substitution, controlling 95
variables in edit macros 95
variables, syntax diagrams xxi
verifying parameters 107
VERSION
    assignment statement 424
    macro command 424
    primary command 292

version number
    controlling 292, 424
    description 26
VIEW
    macro command 425
    primary command 294
VOLUME assignment statement 426
Volume Information 426

W

writing program macros 89, 91

X

X (exclude), line command
    using 54, 58
XSTATUS, assignment statement 426

Z

z/OS UNIX
    specifying pathnames 16
z/OS UNIX files
    copying and moving data 41
    creating and replacing data 41
ZDEFAULT edit profile 21
ZEDILMSG dialog variable 110
ZEDISMSG dialog variable 110
ZEDITCMD variable 102
ZEDLMSG 102
ZEDMSGNO dialog variable 110
ZEDSAVE variable 324
ZEDSMSG 102
ZUSERMAC variable 25

SC19-3621-30