

File Manager for z/OS V9R1



Addenda to V9R1 User's Guides and Customization Guide

File Manager for z/OS V9R1



Addenda to V9R1 User's Guides and Customization Guide

Contents

About this document v

Part 1. PTF/APAR documentation changes 1

UK64397, UK64398, UK64399, UK64400 3

PM28967. 3
Changes to the User's Guide and Reference for
DB2 3

UK63245 5

PM20660. 5
Changes to the User's Guide and Reference for
DB2 5
PM22660. 6
Changes to the User's Guide and Reference . . . 7

UK62005 9

PM20322. 9
Changes to the User's Guide and Reference for
DB2 9

UK58321, UK08097, UK08098, UK08099 13

PM08924 13
Changes to the User's Guide and Reference. . . 14

UK54336 15

PM02105 15
Changes to the User's Guide and Reference for
DB2 15
PK95961 16
Changes to the Customization Guide. 17

UK52267 19

PK93460 20
Changes to the Customization Guide. 20
PK94449 30
Changes to the Customization Guide. 30
PK95812 31
Changes to the User's Guide and Reference. . . 31
Changes to the User's Guide and Reference for
IMS Data 35

**UK49461, UK49462, UK49463, UK49464,
UK49465, UK49466, UK49467 37**

PK83865 37
Changes to the Customization Guide. 37
PK85062 38
Changes to the User's Guide and Reference. . . 38

**UK48204, UK48205, UK48206, UK48207,
UK48239, UK48243, UK48254 39**

PK84077 39
Changes to the Customization Guide. 39
Changes to the User's Guide and Reference for
DB2 39

UK47675 41

PK84110 41
Changes to the User's Guide and Reference for
DB2 42
PK90394 43
Changes to the User's Guide and Reference for
DB2 43

UK44837 45

PK77613 45
Changes to the User's Guide and Reference. . . 45

UK44322 53

PK79535 53
Changes to the User's Guide and Reference. . . 53

UK42251 55

PK75870 55
Changes to the User's Guide and Reference. . . 55
Changes to the User's Guide and Reference for
IMS 57
Changes to the Customization Guide. 60

Part 2. General documentation changes 63

Customization Guide (SC19-2494-00) 65

December 2008 65
Part 1. Customizing File Manager 65
Part 4. Customizing File Manager CICS
Component 65
Appendix A. File Manager Options 65
June 2009 66
Part 1. Customizing File Manager 66

**User's Guide and Reference
(SC19-2495-00) 67**

Change #8: June 2010 67
Chapter 16, "Functions" 67
Change #7: May 2010 67
Change #6: May 2010 67
Chapter 16, "Functions" 67
Change #5: April 2010 67
Change #4: August 2009 67
Change #3: May 2009 68
Change #2: December 2008 68
Change #1: November 2008 68

**User's Guide and Reference for DB2
Data (SC19-2496-00) 71**
Change #1 November 2010 71

**User's Guide and Reference for IMS
Data (SC19-2497-00) 73**

**User's Guide and Reference for CICS
(SC19-2498-00) 75**

Part 3. Appendixes. 77
Index 79

About this document

This document provides details of all the APAR service fixes that impact upon documentation, for IBM File Manager for z/OS Version 9.1, since the most recent edition of the product manuals in September 2008. These editions are:

- Customization Guide (SC19-2494-00) - First Edition
- User's Guide and Reference (SC19-2495-00) - First Edition
- User's Guide and Reference for DB2 Data (SC19-2496-00) - First Edition
- User's Guide and Reference for IMS Data (SC19-2497-00) - First Edition
- User's Guide and Reference for CICS (SC19-2498-00) - First Edition

The Addendum document is divided into two parts:

- **Part One: PTF/APAR documentation changes**

This section lists the changes to the File Manager for z/OS Version 9 Release 1 documentation that are required to reflect new behavior resulting from the application of APAR fixes.

The fixes are listed by PTF number, in reverse date order, so that the most recently released fix appears at the beginning of the document. Each description shows:

- The set of PTF numbers in the release
- The date of the PTF release
- The APARs included in the released fix
- Details of those APAR changes that affect documentation
- Page references for the manuals affected by the change

Notes:

1. This document does NOT describe those APAR fixes that do not have an impact upon documentation.
2. The enhancements and corrections described in this section are only available after applying the listed PTFs for the APAR.

- **General documentation changes**

This section describes enhancements, corrections and updates in the documentation for File Manager for z/OS Version 9 Release 1. These changes are not associated with PTF numbers, as they do not require the application of any code updates.

The changes are grouped by manual and listed within each section in reverse date order. That is, the most recent documentation change appears at the beginning of each manual section.

Part 1. PTF/APAR documentation changes

UK64397, UK64398, UK64399, UK64400	3	PK83865	37
PM28967.	3	Changes to the Customization Guide.	37
Changes to the User's Guide and Reference for		Chapter 2, "Customizing the operating	
DB2	3	environment for File Manager".	37
Chapter 4, "Viewing and changing DB2 data"	3	PK85062	38
UK63245	5	Changes to the User's Guide and Reference.	38
PM20660.	5	Chapter 16, "Functions"	38
Changes to the User's Guide and Reference for		UK48204, UK48205, UK48206, UK48207,	
DB2	5	UK48239, UK48243, UK48254	39
Chapter 15, "FM/DB2 panels and fields"	5	PK84077	39
PM22660.	6	Changes to the Customization Guide.	39
Changes to the User's Guide and Reference	7	Chapter 1, "Preparing to customize File	
Chapter 16, "Functions".	7	Manager"	39
UK62005	9	Changes to the User's Guide and Reference for	
PM20322.	9	DB2	39
Changes to the User's Guide and Reference for		Chapter 2, "Getting started with FM/DB2"	39
DB2	9	UK47675	41
Chapter 17, "Functions".	9	PK84110	41
UK58321, UK08097, UK08098, UK08099	13	Changes to the User's Guide and Reference for	
PM08924	13	DB2	42
Changes to the User's Guide and Reference.	14	Chapter 3, "Working with templates",	
Chapter 16, "Functions"	14	subsection "Handling special data"	42
UK54336	15	PK90394	43
PM02105	15	Changes to the User's Guide and Reference for	
Changes to the User's Guide and Reference for		DB2	43
DB2	15	Chapter 4, "Viewing and changing DB2 data",	
Chapter 15, "FM/DB2 panels and fields".	15	subsection "Handling special data"	43
Chapter 17, "FM/DB2 functions"	16	UK44837	45
PK95961	16	PK77613	45
Changes to the Customization Guide.	17	Changes to the User's Guide and Reference.	45
Appendix B, "FM/DB2 options"	17	Chapter 14, "Panels and fields".	46
UK52267	19	Chapter 16, "Functions"	46
PK93460	20	Replacement pages for "Copy To panel"	
Changes to the Customization Guide.	20	section	47
Chapter 20, "Customizing the FM/IMS		Replacement pages for "Find/Change Utility	
security environment".	20	panel" section	48
PK94449	30	Replacement pages for "DSC (Data Set Copy)"	
Changes to the Customization Guide.	30	section	49
Chapter 12, "Customizing the operating		Replacement pages for "DSM (Data Set	
environment for FM/DB2"	30	Copy)", "DSP (Data Set Print)", "DSU (Data	
PK95812	31	Set Update)" sections	50
Changes to the User's Guide and Reference.	31	Replacement pages for "FCH (Find/Change)"	
Chapter 16, "Functions"	31	section	51
Chapter 17, "File Manager messages".	34	Replacement pages for "External REXX	
Changes to the User's Guide and Reference for		functions" section	52
IMS Data	35	UK44322	53
Chapter 11, "Batch reference"	35	PK79535	53
UK49461, UK49462, UK49463, UK49464,		Changes to the User's Guide and Reference.	53
UK49465, UK49466, UK49467	37	Chapter 15, "Primary commands", subsection	
		"LOCATE primary command"	53

UK42251	55
PK75870	55
Changes to the User's Guide and Reference.	55
Chapter 14, "Panels and fields", subsection	
"Print Audit Trail panel"	55
Chapter 16, "Functions", section "File Manager	
functions", subsection "AUD (Print Audit Trail	
Report)"	56
Changes to the User's Guide and Reference for	
IMS	57
Chapter 9, "Panels and fields", subsection	
"Print Audit Trail panel"	57
Chapter 11, "Batch reference", subsection	
"Print Audit Report (AUD)"	58
Changes to the Customization Guide.	60
Chapter 5, "Customizing the File Manager	
audit facility", subsection "Setting the	
appropriate options to produce an audit trail"	60
Chapter 14, "Customizing the FM/DB2 audit	
facility", subsection "Determining if an audit	
trail is to be produced"	60
Chapter 19, "Customizing FM/IMS"	60
Chapter 20, "Customizing the FM/IMS	
security environment", subsection, "Security	
Exit Parameters", Table 30. Parameters - Exit	
Type A	61
Chapter 21, "Customizing the FM/IMS audit	
facility"	61
Chapter 29, "Customizing the FM/CICS audit	
facility"	61
Appendix A, "File Manager options"	61
Appendix B, "FM/DB2 options"	61
Appendix C, "FM/IMS options"	62

UK64397, UK64398, UK64399, UK64400

Release Date: 3 February 2011

This set of PTFs contains the following APAR fixes:

APAR #	APAR Abstract	Doc Impact
PM28967	When using FM/DB2 view or edit, additional rows are displayed than what exists in the DB2 table.	User's Guide and Reference for DB2 (SC19-2496-00)

PM28967

Initial problem description

In File Manager DB2 edit, large table mode, scrolling to the bottom (max) of the object shows extraneous rows beyond the last row in the object.

Outline of solution

File Manager DB2 component has been changed to correct the problem. This APAR also changes the error message displayed when the editor SORT command is issued in an FM/DB2 large edit session. The FM/DB2 User's Guide and Reference for DB2 data is updated to explicitly state that the editor SORT command is not available in a large mode editor session.

Documentation impact

This APAR requires changes to be made to the User's Guide and Reference for DB2 (SC19-2496-00).

Changes to the User's Guide and Reference for DB2

Chapter 4, "Viewing and changing DB2 data"

In the section "Starting and ending FM/DB2 editor sessions", subsection "Specifying the editor session mode: normal mode or large mode?", subsection "Choosing the appropriate editor mode", add the following to Table 4, "Characteristics of editor modes":

Table 1.

	"Normal mode"	"Large mode"
Sort command	Allowed.	Prohibited.

UK63245

Release Date: **24 December 2010**

This set of PTFs contains the following APAR fixes:

APAR #	APAR Abstract	Doc Impact
PM20660	Receive SQLCODE -530 when attempting to copy a DB2 table using FM/DB2 COPY utility.	User's Guide and Reference for DB2 (SC19-2496-00)
PM22660	FINDNEXT function returning "TOF" value instead of record number or zero.	User's Guide and Reference (SC19-2495-00)

PM20660

Initial problem description

In the FM/DB2 Copy utility, the option **Ignore referential integrity errors** is not working.

Outline of solution

FM/DB2 has been updated to correct the problem. The option description has been changed to better reflect the option's function. When the option is selected, any SQLCODE-530 or SQLCODE-545 errors are ignored.

Documentation impact

This APAR requires changes to be made to:

- User's Guide and Reference for DB2 (SC19-2496-00)

Changes to the User's Guide and Reference for DB2

Chapter 15, "FM/DB2 panels and fields"

In the section, "Copy Options panel":

- Replace the panel with:

<u>P</u> rocess	<u>O</u> ptions	<u>U</u> tilities	<u>H</u> elp
FM/DB2 (DFA2)	Copy Options	Global Settings	
From Table Concurrency Option: Enter "/" to select option / Use uncommitted read		To Table Locking Option: Locking 1 1. None 2. Share mode 3. Exclusive mode	
Copy Options: Duplicate key processing 2 1. Ignore 2. Update For ALL _____ duplicates		Enter "/" to select option - Delete existing rows - Ignore RI/Constraint errors 7 Create audit trail	
Command ==>			
F1=Help	F2=Split	F3=Exit	F7=Backward F8=Forward F9=Swap
F12=Cancel			

- Replace the text and description for the **Ignore referential integrity errors** option with the following:

Ignore RI/Constraint errors

Determines what processing occurs when the Copy utility encounters an SQLCODE-530 (RI error - no primary key) or SQLCODE-545 (Constraint error).

/ The Copy process ignores the error and continues with the next row. The row is not copied or updated and is not included in the copy count.

(blank)

Copy utility canceled. This is the default setting.

PM22660

Initial problem description

FINDPREV() FINDNEXT functions are:

1. Returning EOF or TOF instead of 0.
2. Not search the current record as documented.
3. Only finding one occurrence of a string in a given record. DOWN() command returns EOF inconsistently.

Outline of solution

FINDPREV(), FINDNEXT() have been changed to ensure the documented behaviour is followed.

1. 0 will be returned when string is not found.
2. Searching will take place from the current record as documented. Repeated searching will resume from the previously located string unless the record location has been changed by a UP(), Down(), TOP(), or BOT() commands.
3. DOWN() behavior will be changed so that it only returns end of file when you are positioned on the last record and the command is issued.

Documentation impact

This APAR requires changes to be made to the User's Guide and Reference (SC19-2495-00).

Changes to the User's Guide and Reference**Chapter 16, "Functions"**

In the section "File Manager functions", subsection "DOWN (DSEB only)", change the sentence:

If, after moving, the current record is the last record, then the DOWN function returns the string value "EOF" (end of file). Otherwise, DOWN returns 0.

to:

If, before moving, the current record is the last record, then the DOWN function returns the string value "EOF" (end of file). Otherwise, DOWN returns 0.

In the section "File Manager functions", subsection "FINDNEXT, FINDPREV (DSEB only)", add the following:

If a FINDNEXT() or FINDPREV() has located a needle on a given record, then repeated searching resumes from the previously located needle plus 1 for FINDNEXT and minus 1 for FINDPREV. If the record location has been changed by an UP(), Down(), TOP(), or BOT() command, or there has been no previous located needle, then FINDPREV searches from the end of the current record backward, and FINDNEXT searches from the start of the current record forward.

UK62005

Release Date: 22 November 2010

This set of PTFs contains the following APAR fixes:

APAR #	APAR Abstract	Doc Impact
PM20322	File Manager alternate field heading too short for DBCS headings.	User's Guide and Reference for DB2 (SC19-2496-00)

PM20322

Initial problem description

1. Header field for template edit is restricted to 20 bytes.
2. There is no function to create or update a DB2 template in batch.

Outline of solution

1. Header field space has been increased to 36 bytes.
2. A new function D2TP has been created to facilitate batch create and update of DB2 templates.

Documentation impact

This APAR requires changes to be made to:

- User's Guide and Reference for DB2 (SC19-2496-00)

Changes to the User's Guide and Reference for DB2

Chapter 17, "Functions"

Subsection: "File Manager DB2 batch functions"

Add the following new function:

D2TP (Template create/update) batch command:

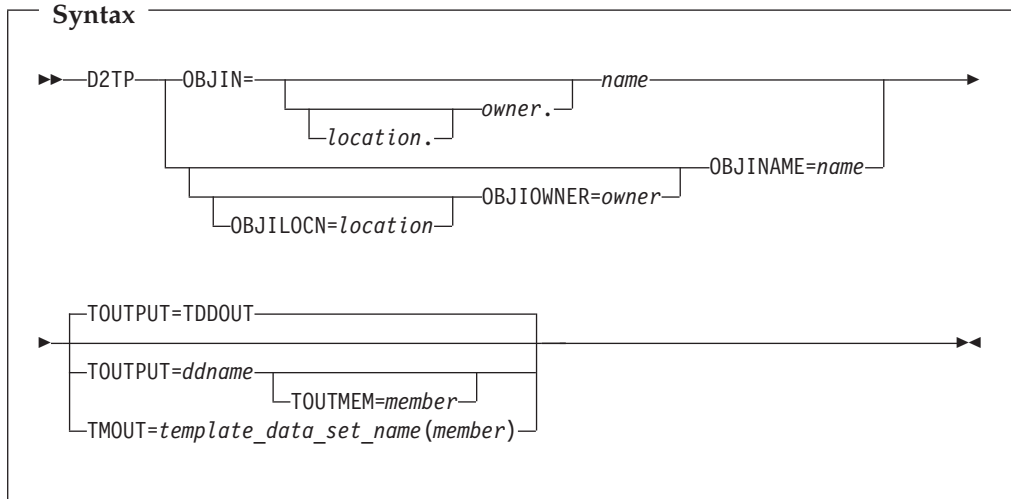
Purpose

To create or update a template based on one DB2 object. The object must be accessible from the currently connected DB2 system.

Usage

The FM/DB2 template create/update utility creates a template if it does not exist, or updates the template if it does exist.

Syntax



OBJIN location.owner.object

The optional name of the DB2 remote server (*location*) where the source object is located; the optional name of the owner of the source object (*owner*) and the source object name (*name*).

When *location* is not specified the current (local) DB2 server is used. When the *owner* is not specified the object name is qualified using the current SQLID. When FM/DB2 generates the utility control statements, the *owner* value is non-blank.

OBJIN should be used when the fully qualified name fits on a single line in the JCL deck. The last usable column is column 71. When the fully qualified name does not fit on a single line in the JCL deck, use one or more of the OBJILOCN, OBJIOWNR, OBJINAME keywords to specify the object.

OBJILOCN=location

The optional name of the DB2 remote server (*location*) where the source object is located. See "Specifying a DB2 object name".

OBJIOWNR=owner

The optional name of the owner of the source object (*owner*). See "Specifying a DB2 object name".

OBJINAME=name

The object name (*name*) for the source object. See "Specifying a DB2 object name".

TOUTPUT=ddname

Defines a reference to a DD statement for the data sets which contain the DB2 template that describes the target DB2 object. Concatenated DD statements are not supported and the referenced data set must be catalogued. If you have not specified a member name in the referenced DD statement, then you must provide a TOUTMEM keyword. If no TOUTPUT or TMOUT parameter has been provided, then TOUTPUT=TDDOUT is used. See "Specifying the template for a DB2 object".

TOUTMEM=member

The name of the template member in the dataset identified by the TOUTPUT or TMOUT parameter. This parameter is ignored if the member

name is provided with the DD statement or the TMOUT parameter. See "Specifying the template for a DB2 object".

TMOUT=*template_data_set_name*(*member*)

The PDS (*template_data_set_name*) and member name (*member*) of the File Manager DB2 template that describes the target DB2 object. See "Specifying the template for a DB2 object".

Examples

Example 1: Create a DB2 template for the DSN8810.EMP using TMOUT keyword.

```
//D2TPJOB (acct),'name'
/* Create template FMN.TEMPLATE(EMP) for DSN8810.EMP table
/*
//FMNDB2 EXEC PGM=FMNDB2,PARM=('SSID=DSN1,SQID=ID1')
//STEPLIB DD DSN=FMN.SFMNMOD1,DISP=SHR
// DD DSN=DB2V810.DSN1.SDSNEXIT,DISP=SHR
// DD DSN=DB2.V810.SDSNLOAD,DISP=SHR
//SYSPRINT DD SYSOUT=*
//FMNTSPRT DD SYSOUT=*
//SYSTEM DD SYSOUT=*
//SYSIN DD *
$$FILEM D2TP OBJIN="DSN8810"."EMP",
$$FILEM TMOUT=FMN.TEMPLATE(EMP)
/*
```

Example 2: Use TOUTPUT, TOUTMEM keywords to create output template for DSN8810.EMP

```
//D2TPJOB (acct),'name'
/* Create template FMN.TEMPLATE(EMP) for DSN8810.EMP table
/*
//FMNDB2 EXEC PGM=FMNDB2,PARM=('SSID=DSN1,SQID=ID1')
//STEPLIB DD DSN=FMN.SFMNMOD1,DISP=SHR
// DD DSN=DB2V810.DSN1.SDSNEXIT,DISP=SHR
// DD DSN=DB2.V810.SDSNLOAD,DISP=SHR
//TOUT DD DSN=FMN.TEMPLATE,DISP=SHR
//SYSPRINT DD SYSOUT=*
//FMNTSPRT DD SYSOUT=*
//SYSTEM DD SYSOUT=*
//SYSIN DD *
$$FILEM D2TP OBJIN="DSN8810"."EMP",
$$FILEM TOUTPUT=TOUT,TOUTMEM=EMP
/*
```

Example 3: Use Default output DD TDDOUT with TOUTMEM to create output template for DSN8810.EMP

```
//D2TPJOB (acct),'name'
/* Create template FMN.TEMPLATE(EMP) for DSN8810.EMP table
/*
//FMNDB2 EXEC PGM=FMNDB2,PARM=('SSID=DSN1,SQID=ID1')
//STEPLIB DD DSN=FMN.SFMNMOD1,DISP=SHR
// DD DSN=DB2V810.DSN1.SDSNEXIT,DISP=SHR
// DD DSN=DB2.V810.SDSNLOAD,DISP=SHR
//TDDOUT DD DSN=FMN.TEMPLATE,DISP=SHR
//SYSPRINT DD SYSOUT=*
//FMNTSPRT DD SYSOUT=*
//SYSTEM DD SYSOUT=*
//SYSIN DD *
$$FILEM D2TP OBJIN="DSN8810"."EMP",
$$FILEM TOUTMEM=EMP
/*
```

Example 4: Use Default output DD TDDOUT with member to create output template for DSN8810.EMP

```
//D2TPJOB (acct),'name'
/* Create template FMN.TEMPLATE(EMP) for DSN8810.EMP table
/*
//FMNDB2 EXEC PGM=FMNDB2,PARM=('SSID=DSN1,SQID=ID1')
//STEPLIB DD DSN=FMN.SFMNMOD1,DISP=SHR
// DD DSN=DB2V810.DSN1.SDSNEXIT,DISP=SHR
// DD DSN=DB2.V810.SDSNLOAD,DISP=SHR
//TDDOUT DD DSN=FMN.TEMPLATE(EMP),DISP=SHR
//SYSPRINT DD SYSOUT=*
//FMNTSPRT DD SYSOUT=*
//SYSTEM DD SYSOUT=*
//SYSIN DD *
$$$FILEM D2TP OBJIN="DSN8810"."EMP"
/*
```

Example 5: Create 3 templates

```
//D2TPJOB (acct),'name'
/* Create template TOUT1-3 for DSN8810.EMP table
/*
//FMNDB2 EXEC PGM=FMNDB2,PARM=('SSID=DSN1,SQID=ID1')
//STEPLIB DD DSN=FMN.SFMNMOD1,DISP=SHR
// DD DSN=DB2V810.DSN1.SDSNEXIT,DISP=SHR
// DD DSN=DB2.V810.SDSNLOAD,DISP=SHR
//TOUT1 DD DSN=FMN.TEMPLATE(T1),DISP=SHR
//TOUT2 DD DSN=FMN.TEMPLATE(T2),DISP=SHR
//TOUT3 DD DSN=FMN.TEMPLATE(T3),DISP=SHR
//SYSPRINT DD SYSOUT=*
//FMNTSPRT DD SYSOUT=*
//SYSTEM DD SYSOUT=*
//SYSIN DD *
$$$FILEM D2TP OBJIN="DSN8810"."EMP",TOUTPUT=TOUT1
$$$FILEM D2TP OBJIN="DSN8810"."EMP",TOUTPUT=TOUT2
$$$FILEM D2TP OBJIN="DSN8810"."EMP",TOUTPUT=TOUT3
/*
```

UK58321, UK08097, UK08098, UK08099

Release Date: 6 July 2010

This set of PTFs contains the following APAR fixes:

APAR #	APAR Abstract	Doc Impact
PM07801	File Manager FLDI, FLDO, TESTC functions using relative start positions produces incorrect results.	None.
PM08432	ABEND0C4 in FMNDVT.	None.
PM08924	The column name is not displayed correctly after the second page is running DSP function with RLEN=Y.	User's Guide and Reference (SC19-2495-00)
PM09604	File Manager DSM (Data Set Compare) job repeats ABENDDD37 followed by ABEND001 if no DCB on SYSPRINT DD statement.	None.
PM11583	Queue names are not displayed correctly on WebSphere MQ queue list panel with TERMTYPE=3270KN environment.	None.
PM12162	PGM=FMNMAIN => S0C4.	None.
PM12366	Receive abend S0C4-X'10' when running FM DSU function in batch mode.	None.
PM13069	Japanese panels for DB2 feature, for example FMN2PD2E, has SBCS middle dot ' ' (X'46') instead of DBCS (X'4345').	None.
PM14437	File Manager miscellaneous problems.	None.
PM15254	Receive compilation errors when running FM from a CLIST.	None.

PM08924

Initial problem description

When printing data in TABL format and the edit option 'Display record length' or batch option RLEN has been selected, the heading on the second and subsequent pages of the report do not include the 'Length' heading and the heading is not aligned correctly.

Outline of solution

The Length heading will be included in the heading of all pages of the report when the option has been selected and the record length is being included in the TABL format report.

Documentation impact

This APAR requires changes to be made to the User's Guide and Reference (SC19-2495-00).

Changes to the User's Guide and Reference

Chapter 16, "Functions"

In the section "File Manager functions", subsection "DSP (Data Set Print)", change the default for parameter RLEN from NO to YES:



UK54336

Release Date: 19 February 2009

This set of PTFs contains the following APAR fixes:

APAR #	APAR Abstract	Doc Impact
PM02105	FM/DB2 export not wrapping graphic fields with shift out/shift in for CSV output.	User's Guide and Reference for DB2 (SC19-2496-00)
PK95961	On panel FMN2PEO7 the settings for the two options are swapped.	Customization Guide (SC19-2494-00)

PM02105

Initial problem description

A number of issues, specific to FM/DB2 export utility, CSV format.

1. GRAPHIC data is not wrapped with the appropriate shift out/shift in characters.
2. No ability to produce the column headers in CSV format.
3. Data with embedded commas is not enclosed within double quotes.
4. Embedded double quotes need escape character or another double quote.

Outline of solution

1. Graphic data will be wrapped with shift out/shift in values.
2. A new online option "Include column headers" and a new batch keyword CSVHDR has been added to optionally produce column header values for the first record.
3. Data with an embedded delimiter value has been enclosed in double quotes.
4. An extra double quote will be inserted for each embedded double quote.

Documentation impact

This APAR requires changes to be made to:

- User's Guide and Reference for DB2 (SC19-2496-00)

Changes to the User's Guide and Reference for DB2

Chapter 15, "FM/DB2 panels and fields"

In the section "Export Options (2 of 3) panel", change the panel to include new panel option **Include column headers** after the **Separator character** field as follows:

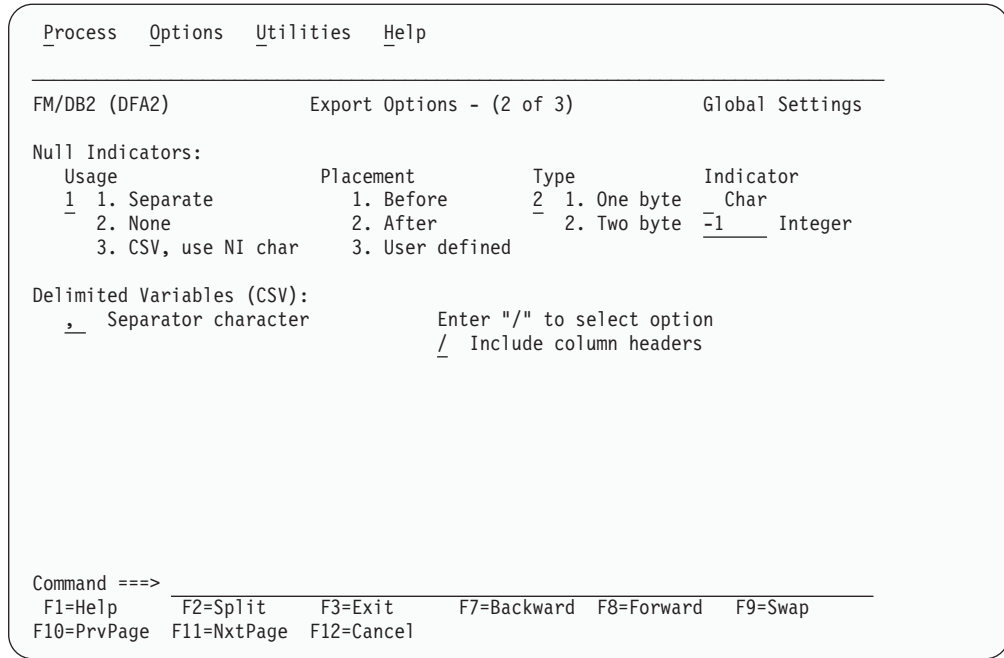


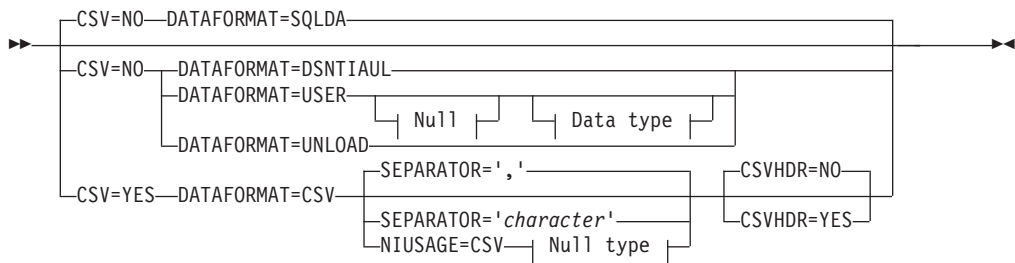
Figure 1. Export Options (2 of 3) panel

Include column headers

Selecting this option produces comma-delimited column headers as the first export record.

Chapter 17, "FM/DB2 functions"

In the section "File Manager DB2 batch functions", subsection "DBX (Export) batch command", add new keyword CSVHDR which is a subset of CSV=YES to the syntax diagram.



CSVHDR

Specifies whether column headers are to be produced on the first record of the export file when CSV=YES has been specified.

NO Column headers are not produced.

YES Column headers are produced.

PK95961

Initial problem description

1. FM/DB2 issues "LOCK TABLE" statements in edit even though the appropriate editor option is set for no (table) locking.
2. FM/DB2 is not adding the "WITH UR" clause to the SELECT statement used to access DB2 data for browse and view only, even though the appropriate editor option is set for uncommitted read.

Outline of solution

File Manager DB2 component has been updated to correct the problems.

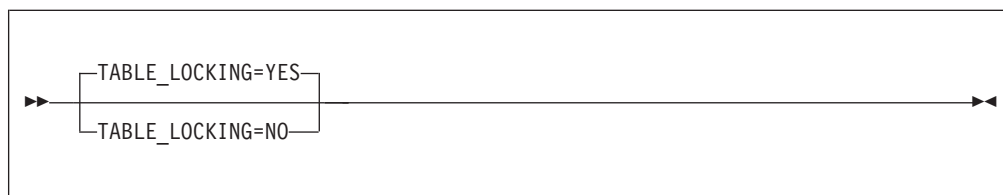
Documentation impact

This APAR requires changes to be made to:

- Customization Guide (SC19-1238-01)

Changes to the Customization Guide**Appendix B, "FM/DB2 options"**

In the section, "FMN2SSDM", add the following new option:

TABLE_LOCKING:**TABLE_LOCKING**

specifies whether the table being edit should be locked.

YES The table is to be locked.

NO The table is not to be locked.

UK52267

Release Date: 27 December 2009

This set of PTFs contains the following APAR fixes:

APAR #	APAR Abstract	Doc Impact
PK92977	FM/DB2 unable to locate catalog table SYSIBM.SYSUSERAUTH	None.
PK93285	Large files load slowly into File Manager when using a template	None.
PK93460	File Manager IMS security enhancement to include IMS subsystem name in SAF rules.	Customization Guide (SC19-2494-00)
PK94449	FM/DB2 requires all product end users to have access to SYSIBM.SYSUSERAUTH.	Customization Guide (SC19-2494-00)
PK94529	File Manager rollback and commit reporting for DB2 and IMS audit reports.	None.
PK95244	Display of template/copybook fields is incorrect for variable records.	None.
PK95812	File Manager miscellaneous base enhancements	User's Guide and Reference (SC19-2495-00) User's Guide and Reference for IMS Data (SC19-2497-00)
PK96107	Update segmented template from copybook (option 7.1), template is not updated with copybook changes.	None.
PK98418	Receive ABEND0C4 running FM FCH function with FINDNOT primary command.	None.
PK99540	Receive S0C4 when running function DVT and VM volumes are online to z/OS.	None.
PM00235	RECEIVE ABEND0C4 when displaying VTOC list using File Manager option 3.5.	None.
PM01134	Getting S0C4 abend trying to view a DSN using a template.	None.

APAR #	APAR Abstract	Doc Impact
PM01888	DSC function with FB input generates an error in the output file when output is VB tape.	None.

PK93460

Initial problem description

User wants to control File Manager access to IMS subsystems using SAF Facility classes. File Manager does not allow for specifying a subsystem ID in the SAF resource names.

Outline of solution

File Manager has been enhanced to perform an extra level of SAF resource checking by appending the subsystem ID to the function and group level resource names and checking the sub system qualified SAF resource access before the current security checking is performed. If the SAF resource with sub system ID appended has not been defined, then the security checking remains the same.

Documentation impact

This APAR requires changes to be made to:

- Customization Guide (SC19-1238-01)

Changes to the Customization Guide

Chapter 20, "Customizing the FM/IMS security environment"

Replace the entire contents of the chapter with the following:

This chapter describes:

- How you can control access to the FM/IMS functions and IMS subsystems.
- How you can use user-written security exit routines to control access to various IMS resources.

Controlling access to IMS subsystems and FM/IMS functions: FM/IMS allows you to control which IMS subsystems a user can access when using each of the functions listed in Table 2. These functions are protected by default when you receive FM/IMS.

Table 2. Protected FM/IMS Functions

Function code	Description	UPDATE or READONLY
DBI	Initialize dialog - generates JCL for the initialize function	UPDATE
DDD	Delete or define dialog - generates JCL to delete or define database data sets	UPDATE
DIB	Initialize - initialize databases (batch)	UPDATE
IB	Browse - browse a database	READONLY
IBBO	Batch browse dialog - generate JCL for the batch browse function	READONLY
IBB	Batch browse - read a database in batch (batch)	READONLY
IE	Edit - edit a database	UPDATE

Controlling access to IMS subsystems and FM/IMS functions

Table 2. Protected FM/IMS Functions (continued)

Function code	Description	UPDATE or READONLY
IEBO	Batch edit dialog - generates JCL for the batch edit function	UPDATE
IEB	Batch edit - edit a database in batch (batch)	UPDATE
IPRO	Print dialog - generates JCL for the print function	READONLY
IPR	Print - print data from databases (batch)	READONLY
IX	Extract dialog - generates JCL for the extract function	READONLY
IXB	Extract - extract data from databases (batch)	READONLY
IL	Load dialog - generates JCL for the load function	UPDATE
ILB	Load - load data into databases (batch)	UPDATE

Table 2 on page 20 lists the function's code (column 1), a brief description of the function (column 2), and whether the function has been classified as an update or a read-only function (column 3).

The facility allows you to grant or deny some or all users access to:

1. Individual IMS subsystems by individual functions in Table 2 on page 20.
2. Individual functions in Table 2 on page 20. When you grant or deny users access to individual functions, they are granted or denied access to all IMS subsystems when using these functions.
3. Individual IMS subsystems by the update or read-only functions.
4. The update or read-only functions. When you grant or deny users access to the update or read-only functions, they are granted or denied access to all IMS subsystems when using the update or read-only functions.

FM/IMS provides security for these functions, in one of two ways, either through RACF (or an equivalent security product) or through the FMNSECUR exit.

If Security Server, RACF 1.9 (or later), or an equivalent security product is active, the System Authorization Facility (SAF) with the File Manager enhanced security facility is used for access control and authorization verification. Authorization is controlled by FM/IMS-specific profiles in the FACILITY class. This chapter describes the FM/IMS-specific profiles that you must define to RACF or your equivalent security product. It also describes how you define these profiles to RACF. If you use another security product, consult the documentation for your product to determine how to define these profiles to your product.

If SAF with RACF 1.9 (or an equivalent security product), is not active when an FM/IMS function is started, the function access control checks are passed to the FMNSECUR user exit instead of to SAF.

To use FMNSECUR, it must be installed in the LPA. If the FMNSECUR module is required and it cannot be found in the LPA, an error message is issued and the FM/IMS function will not start.

FMNSECUR is a customizable exit. It provides FMNS macros which allow you to define a table of user names or job names, File Manager-protectable resources (called profiles), and access levels. For information on FMNSECUR, see "Setting up the security environment by using FMNSECUR".

Controlling access to IMS subsystems and FM/IMS functions

Notes:

1. The FMNSECUR module is not used (even if present) if SAF with RACF 1.9 (or an equivalent security product), is active when an FM/IMS function is started.
2. FM/IMS functions that are not listed in Table 2 cannot be protected by RACF (or an equivalent security product) or by the FMNSECUR exit.

Most installations will use RACF or an equivalent security product to protect their functions. The rest of this section describes how you implement RACF (or an equivalent security product) security for the functions in Table 2.

Controlling access to the update or read-only functions: The update functions in Table 2 are protected by the profile FILEM.IMS.UPDATE. The read-only functions in this table are protected by the profile FILEM.IMS.RDONLY. As a minimum, you need to define these profiles and grant or deny users access to them.

You define these profiles in the FACILITY class. You do this by entering the following RACF commands:

```
RDEFINE FACILITY FILEM.IMS.UPDATE UACC(READ or NONE)
RDEFINE FACILITY FILEM.IMS.RDONLY UACC(READ or NONE)
```

Specify:

- UACC(READ), if you want users or groups to be granted access to these resources, unless they are specifically denied access.
- UACC(NONE), if you want users or groups to be denied access to these resources, unless they are specifically granted access.

In the following, assume that:

- The RDEFINE for the FILEM.IMS.UPDATE profile specifies UACC(NONE), so users and groups are denied access to the update functions unless they are specifically granted access.
- The RDEFINE for the FILEM.IMS.RDONLY profile specifies UACC(READ), so users and groups are granted access to the read-only functions unless they are specifically denied access.

To grant a user (with userid *userid*) or a group (with groupid *groupid*) access to the update functions, you enter one of the following RACF commands:

```
PERMIT FILEM.IMS.UPDATE CLASS(FACILITY) ID(userid) ACCESS(READ)
PERMIT FILEM.IMS.UPDATE CLASS(FACILITY) ID(groupid) ACCESS(READ)
```

To deny a user (with userid *userid*) or a group (with groupid *groupid*) access to the read-only functions, you enter one of the following RACF commands:

```
PERMIT FILEM.IMS.RDONLY CLASS(FACILITY) ID(userid) ACCESS(NONE)
PERMIT FILEM.IMS.RDONLY CLASS(FACILITY) ID(groupid) ACCESS(NONE)
```

Controlling access to individual IMS subsystems by the update or read-only functions: Access to individual IMS subsystems by the update functions in Table 2 on page 20 is protected by the profiles FILEM.IMS.UPDATE.ssid, where ssid is the IMS subsystem ID. Access to individual IMS subsystems by the read-only functions in Table 2 on page 20 is protected by the profiles FILEM.IMS.RDONLY.ssid, where ssid is the IMS subsystem ID.

If you want to grant or deny some or all users access to individual IMS subsystems by the update or read-only functions in Table 2 on page 20, you will need to define the aforementioned profiles.

Controlling access to individual IMS subsystems by the update or read-only functions

You define these profiles in the FACILITY class. You do this by entering the following RACF commands:

```
RDEFINE FACILITY FILEM.IMS.UPDATE.ssid UACC(READ or NONE)
RDEFINE FACILITY FILEM.IMS.RDONLY.ssid UACC(READ or NONE)
```

Specify:

- UACC(READ), if you want users or groups to be granted access to these resources, unless they are specifically denied access
- UACC(NONE), if you want users or groups to be denied access to these resources, unless they are specifically granted access.

You use the PERMIT commands to grant or deny users and groups access to these resources, in the same way as for the FILEM.IMS.UPDATE and FILEM.IMS.RDONLY profiles described in “Controlling access to the update or read-only functions” on page 22.

Controlling access to individual functions: Individual functions in Table 2 on page 20 are protected by the profiles FILEM.FUNCTION.*fc*, where *fc* is the function code. If you want to grant or deny some or all users access to individual functions in Table 2, you will need to define the profiles for these functions.

You define these profiles in the FACILITY class, by entering the following RECF command:

```
RDEFINE FACILITY FILEM.FUNCTION.fc UACC(READ or NONE)
```

Specify:

- UACC(READ), if you want users or groups to be granted access to this function, unless they are specifically denied access
- UACC(NONE), if you want users or groups to be denied access to this function, unless they are specifically granted access.

You use the PERMIT commands to grant or deny users and groups access to these function, in the same way as for the FILEM.IMS.UPDATE and FILEM.IMS.RDONLY profiles.

Controlling access to individual IMS subsystems by individual functions:

Access to individual IMS subsystems by individual functions in Table 2 is protected by the profiles FILEM.FUNCTION.*fc.ssid*, where *fc* is the function code and *ssid* is the IMS subsystem ID.

If you want to grant or deny some or all users access to individual IMS subsystems by individual functions in Table 2, you will need to define the aforementioned profiles.

You define these profiles in the FACILITY class. You do this by entering the following RACF command:

```
RDEFINE FACILITY FILEM.FUNCTION.fc,ssid UACC(READ or NONE)
```

Specify:

- UACC(READ), if you want users or groups to be granted access to these resources, unless they are specifically denied access
- UACC(NONE), if you want users or groups to be denied access to these resources, unless they are specifically granted access.

Controlling access to individual IMS subsystems by individual functions

You use the PERMIT commands to grant or deny users and groups access to these resources in the same way as for the FILEM.IMS.UPDATE and FILEM.IMS.RDONLY profiles described in “Controlling access to the update or read-only functions” on page 22.

What governs whether access is granted or denied: Access to a subsystem *ssid* by an *update* function in Table 2 with function code *fc* is governed by the first profile in this list

- FILEM.FUNCTION.*fc.ssid*
- FILEM.FUNCTION.*fc*
- FILEM.IMS.UPDATE.*ssid*
- FILEM.IMS.UPDATE

Access to a subsystem *ssid* by a *read-only* function in Table 2 with function code *fc* is governed by the first profile in this list that has been defined in the FACILITY class:

- FILEM.FUNCTION.*fc.ssid*
- FILEM.FUNCTION.*fc*
- FILEM.IMS.RDONLY.*ssid*
- FILEM.IMS.RDONLY

Figure 2 on page 25 illustrates the security checking that FM/IMS performs when a function in Table 2 attempts to access an IMS subsystem.

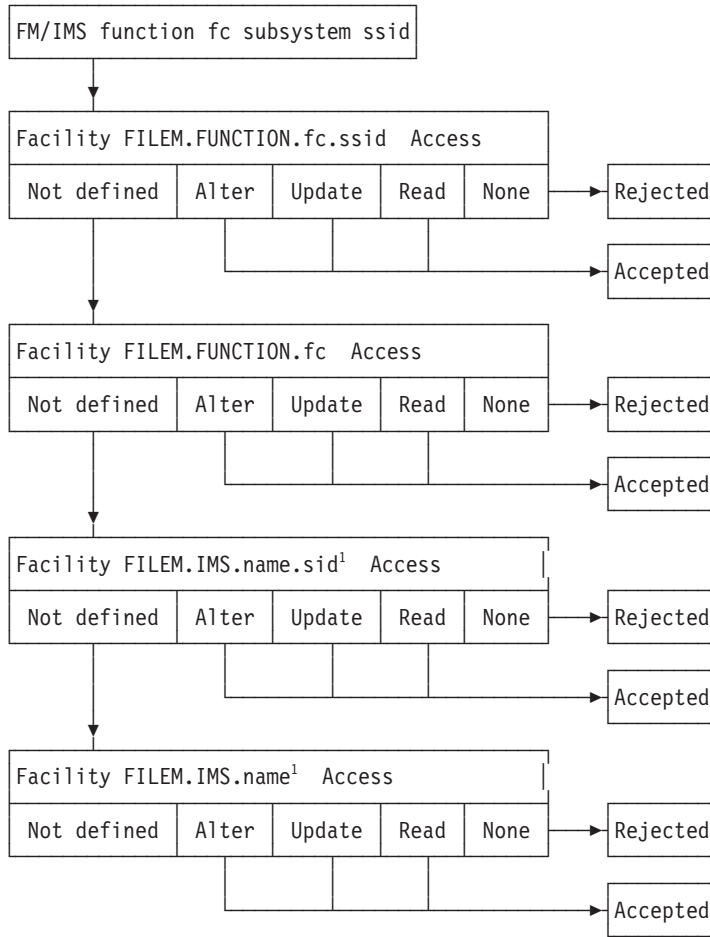


Figure 2. Security checking for FM/IMS functions

Notes:

1. FILEM.IMS.name is either FILEM.IMS.UPDATE or FILEM.IMS.RDONLY

ALTER, UPDATE or READ access means that the user can use the function. Access NONE means that the user cannot use the function.

What governs whether access is granted or denied

Important information for users of non-IBM security products

FM/IMS issues a RACROUTE TYPE=AUTH for FILEM.FUNCTION,*fc*, and if this profile is not defined, RACF returns RC=4. If RC=4 is returned, FM/IMS issues a RACROUTE for either FILEM.IMS.UPDATE (if it is an update function), or FILEM.IMS.RDONLY (if it is a read-only function). If the return code from the RACROUTE TYPE=AUTH for FILEM.FUNCTION,*fc* is greater than 4, no RACROUTE is issued for FILEM.IMS.UPDATE or FILEM.IMS.RDONLY.

You should be aware that not all non-IBM security products issue RC=4 when a RACROUTE TYPE=AUTH for a profile fails; in this case no RACROUTE is issued for FILEM.IMS.UPDATE or FILEM.IMS.RDONLY, and the access request fails immediately. If this applies to your security product, you will have to provide individual profiles for all FM/IMS functions listed in Table 2 on page 20.

Customizing the FM/IMS security exit: FM/IMS provides a security exit module, FMN1SXT. FMN1SXT is called from four different points during processing.

- Exit Type A - Prior to allocating an audit trail data set when editing a database.
- Exit Type D - When allocating a database data set (DLI mode only).
- Exit Type I - Prior to invoking the IMS region controller.
- Exit Type T - After the IMS region controller terminates.

The default security program supplied returns control immediately for all exit types thus allowing normal processing to continue.

You can provide your own version of FMN1SXT in High Level Assembler or COBOL, using either of the sample source decks, FMN1XITA (HLASM) or FMN1XITC(COBOL), as a base. FMN1XITA and FMN1XITC are distributed in FMN1.SFMNSAM1.

Types of security exits: The following describes what you can use each exit type for.

Audit Trail Exit - Type A: The Audit Trail exit can be used to:

- Force the creation of an audit trail for a certain database or group of databases or for a certain user or group of users, or both.
- Override the standard audit trail data set name that is constructed by FM/IMS.
- Force the use of System Management Facilities (SMF) recording for the audit trail instead of using an audit trail data set. If you want to use SMF recording, the SMF record ID to be used must have been specified in the options macro, FMN1POPT.
- This exit is called only for the Edit function.

Database Data set Allocation Exit - Type D: The database data set allocation exit is called when the function runs in DLI mode. For example, this exit can be used to:

- Control access to database data sets.
- Override the database data set allocation status from OLD to SHR during database edit if IMS data sharing is used at your installation.

IMS Initialization Exit - Type I: The IMS Initialization exit can be used to:

- Control access to databases.

- Validate the value entered by the user for the IMS log data set. You can either accept, override, or disallow the value entered by the user.
- Override the standard FM/IMS log data set naming conventions.
- Override the Profile Option MAXGN.

IMS Termination Exit - Type T: The IMS Termination exit can be used to perform post-IMS processing of the log data set. This exit type has no parameters, it is provided as a point where you can add your own REXX code for termination processing

Invoking the security exit: The security exit program is invoked as follows:
 CALL FMN1SXT (FMN_SECURITY_PARAMETERS, FMN_SECURITY_WORKAREA, FMN_IMS_SECURITY_PARAMETERS)

The security exit interface parameters are:

- The security parameter list passed to the security exit. This parameter list is documented in the following pages.
- A security area work area which is a 256-byte area initialized to binary zeroes. This area is unchanged by FM/IMS and can be used to pass information between multiple calls to the security exit.
- IMS security parameters which are passed to the IMS Initialization and Termination exits.

Common Exit Parameters:

Table 3. Common Parameters - All Exit Types

Field	Update	Size	Description
Request Type	N	Char(1)	A Audit Trail allocation D Database data set allocation I IMS Initialization T IMS Termination
Option	N	Char(1)	B Browse E Edit L Extract/Load P Batch printing U Utilities
User Id	N	Char(7)	TSO userid
Permitted	Y	Char(1)	Y Allow intended action N Disallow intended action Blank N/A Not used for Exit Type A.

Security Exit Parameters:

Table 4. Parameters - Exit Type A.

Field	Update	Size	Description
DB DSN	N	Char(44)	This field is obsolete and no longer populated.
DSN	Y	Char(44)	The Audit Trail DSN to be allocated.
SMF Record Id	Y	Binary(16)	Default is 00 (SMF not used for audit trail). Choose a number between 128 and 255. Only applies if the 'Create Indicator' (below) is set to Y.

Security Exit Parameters

Table 4. Parameters - Exit Type A. (continued)

Field	Update	Size	Description
Create	Y	Char(1)	Input value from the Edit Entry Panel. Override the value with one of the following: Y Create an audit trail. N Do not create an audit trail. D See IMSAUDLG=D description.
Keep	Y	Char(1)	Y Keep the audit trail data set. N Delete the audit trail data set after printing.
Report	Y	Char(1)	Y Produce the Audit Trail report at the end of the edit session. N Do not produce the Audit Trail report at the end of the edit session.
Job Type	N	Char(1)	B DLI mode M BMP mode
DBD DSN	N	Char(44)	The DBD library that has been allocated.
DBD Name	N	Char(8)	The DBD being processed.
IMS System Id	N	Char(4)	The IMS SYSTEM ID entered.
Appl. Group Name	N	Char(8)	The AGN used.
PSB Type	N	Char(1)	S Static D Dynamic
PSB Name	N	Char(8)	The name of the Program Specification Block (PSB).
PSB DSN	N	Char(44)	The name of the PSB library data set (static DLI).

Table 5. Parameters - Exit Type D.

Field	Update	Size	Description
Primary DBD Name	N	Char(8)	The primary DBD being processed.
Physical DBD Name	N	Char(8)	The physical DBD that has the DATASET= statement for this DDNAME.
DDNAME	N	Char(8)	The DDNAME of the database being allocated.
DSN Status	Y	Char(3)	SHR or OLD. Default is SHR for Browse and OLD for Edit.
DSN	N	Char(44)	The DSN of the database.

Table 6. Parameters - Exit Type I

Field	Update	Size	Description
DBD DSN	N	Char(44)	The DBD library that has been allocated.
DBD Name	N	Char(8)	The DBD being processed.

Table 6. Parameters - Exit Type I (continued)

Field	Update	Size	Description
Subfunction	N	Char(1)	Option subfunction. Function L, Extract/Load: E Extract L Load Function U, Utilities: D Delete/define database data sets I Initialize IMS databases
IMS Log Indicator	Y	Char(1)	The user entered IMS log indicator. Only applicable during the online portion of the Edit and Load functions. Values are: K Allocate and keep log data set. D Allocate and delete log data set. N Do not use a log data set
IEFRDER DSN	Y	Char(44)	The IMS log data set name as constructed by FM/IMS when the 'IMS log data set' option has been selected in the Edit and Load online functions.
Job Type	N	Char(1)	B DLI mode M BMP mode
IMS System Id	N	Char(4)	The IMS SYSTEM ID entered.
Appl. Group Name	N	Char(8)	The AGN used.
Processing Option	N	Char(1)	The Database Load Processing Option as specified by the user. 1 Update and insert segments. 2 Insert new segments only.
Time	N	Char(6)	The time the exit call is made. Format is HHMMSS.
Date	N	Char(8)	The date the exit call is made. Format is YYYYMMDD.
PSB Type	N	Char(1)	S Static D Dynamic
PSB Name	N	Char(8)	The name of the Program Specification Block (PSB).
PSB DSN	N	Char(44)	The name of the PSB library data set (static DLI).
MAXGN Value	Y	Binary(16)	This value is only applicable during the Edit and Browse functions. The maximum number of GN (Get Next) calls allowed to satisfy a FIND or CHANGE command. Used to override the Profile Option MAXGN value.

Sample programs for a security exit: Copybooks, sample program source and JCL for the security exit are supplied for High Level Assembler and COBOL. The sample program source and JCL are distributed in FMN.SFMNSAM1, and the copybooks in FMN.SFMNMAC1. They are:

FMN1AXIT HLASM copybook for security exit parameters.

FMN1XITA Sample HLASM code for program FMN1SXT.

FMN1UMDS Usermod to install an HLASM version of FMN1SXT.

Sample programs for a security exit

FMN1CXIT COBOL copybook for security exit parameters.
FMN1XITC Sample COBOL code for program FMN1SXT.
FMN1SECC Job control to install a COBOL version of FMN1SXT.

To provide your version of FMN1SXT in HLASM you use the usermod FMN1UMDS as follows:

1. Copy the member FMN1XITA from FMN.SFMNSAM1 to your own source library.
2. Code your version of FMN1XITA in your source library, using FMN1XITA from FMN.SFMNSAM1 as a base.
3. Modify the usermod FMN1UMDS member in FMN.SFMNSAM1 to meet your requirements. Refer to the usermod for information about changes you might need to make.
4. Install SMP/E usermod FMN1UMDS.

To provide your version of FMN1SXT in COBOL:

1. Copy the member FMN1XITC from FMN.SFMNSAM1 to your own source library.
2. Code your version of FMN1XITC in your source library, using FMN1XITC from FMN.SFMNSAM1 as a base.
3. Modify the sample job FMN1SECC in FMN.SFMNSAM1 to meet your site's requirements. Refer to the sample job for information about any changes you might need to make.
4. Run the job to FMN1SECC to compile and link your version of FMN1SXT. This job will link FMN1SXT into FMN.SFMNMOD1.

To implement your exit, add FMN.SFMNMOD1 to your LINKLIST or to the STEPLIB DD statement in your TSO logon procedure.

Note: If the security exit program is written in COBOL, the performance of the application may be impacted.

PK94449

Initial problem description

Unable to use File Manager DB2 component when SELECT access to SYSIBM.SYSUSERAUTH has not been granted to the user.

Outline of solution

File Manager DB2 component has been changed to ignore any errors associated with users not having SELECT access to SYSIBM.SYSUSERAUTH (as documented in the Customization Guide).

Documentation impact

This APAR requires changes to be made to:

- Customization Guide (SC19-1238-01)

Changes to the Customization Guide

Chapter 12, "Customizing the operating environment for FM/DB2"

In the section, "Granting access to the DB2 catalogs (required)", replace the first paragraph with the following:

"FM/DB2 uses dynamic SQL, issued against the DB2 catalog, as part of its processing. To make FM/DB2 available and to ensure the product's correct and optimum operation, you need to grant SELECT access against the DB2 catalog tables to all FM/DB2 users.

Granting access to the DB2 catalog tables can be done in a number of ways, as described below. Some sites have security requirements in place that conflict with FM/DB2's requirement for DB2 catalog access. If, after reading the various options described below, it is not possible to grant the required level of DB2 catalog table access to FM/DB2 users, you should not attempt to install the FM/DB2 product."

PK95812

Initial problem description

1. PRINT function is REXX only and therefore uses high CPU.
2. WRITE() statement to UNIT=AFF data sets ends up in ABENDS413-04.
3. No ability to specify multiple member names to DSP, DSU and FCH.

Outline of solution

New FASTREXX functions PRT_OUT, PRT_IN, and PRT_VAR have been developed to support printing with FASTREXX. WRITE() processing will now detect UNIT=AFF and force a close for a any previous WRITE() statement that are holding the unit. If a WRITE statement causes a re-open of the unit, then the File Manager procedure will terminate.

Documentation impact

This APAR requires changes to be made to:

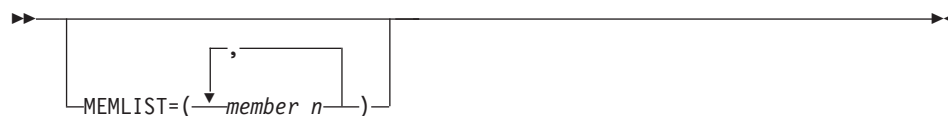
- User's Guide and Reference (SC19-2495-00)
- User's Guide and Reference for IMS Data (SC19-2497-00)

Changes to the User's Guide and Reference

Chapter 16, "Functions"

Subsection: "File Manager functions"

- For the DSC function, in the description for the keyword CPYMBR, change the following sentence in the description for the variable *from_mem* from:
"Generic name masks are not allowed."
to:
"Generic name masks are allowed."
- For the BTB and BTU functions, in the description for the keyword MEMLIST, change the following sentence in the description for the variable *member_n* from:
"Generic name masks are not allowed."
to:
"Generic name masks are allowed."
- For the DSP, FCH, and DSU functions:
 - Add the MEMLIST keyword to the syntax diagram:



- Add the following description after the syntax diagram:

MEMLIST

Allows you to specify a list of member names.

member_n

The name of the member to be processed. Generic name masks are allowed.

Subsection: "External REXX functions"

- Add new functions PRT_OUT, PRT_VAR, and PRT_IN:

PRT_IN

<p>Syntax</p> <p>▶—PRT_IN(<i>format</i>,<i>count</i>)—▶</p>
--

Can be used in FASTREXX condition expressions.

Prints the input record in the format specified.

format

Format in which the input record is to be printed. Valid values are: CHAR, HEX, SNGL, or TABL.

If you specify TABL or SNGL format:

- On the function or panel that you are enhancing, you must specify a copybook or template that describes the record type to be printed.
- File Manager determines the type of the record to be printed by using the record identification criteria if present, or by comparing its length with the record types in the template.
- When using PRT_IN with DSC or the Copy Utility (option 3.3) and you have specified both an input and an output copybook or template, the copybook or template used to format the printed record is the output template.
- Only those fields that have been selected in the template are printed.

count FASTREXX only. The maximum number of times this function will be performed. The default is no limit.

Note: Avoid using PRT_IN in a REXX procedure for:

- DSP function
- FCH function
- Print Utility (option 3.2)

Output from PRT_IN is interspersed with the normal output (from DSP, FCH, or Print) which can be confusing.

Using PRT_IN in a REXX procedure that runs from the Find/Change Utility panel (option 3.6) does not result in this problem because the report produced by the panel is sent to a data set, separate from print output.

Example

Print the first 5 records in hex format and print the remainder in tabular format.

```
PRT_IN('HEX',5)
If recsin() > 5 then PRT_IN('TABL')
```


PRT_OUT

Syntax

```
▶▶PRT_OUT(format,count)◀◀
```

Can be used in FASTREXX condition expressions.

Prints the output record in the format specified.

format

Format in which the output record is to be printed. Valid values are: CHAR, HEX, SNGL, or TABL.

If you specify TABL or SNGL format:

- On the function or panel that you are enhancing, you must specify a copybook or template that describes the record type to be printed.
- File Manager determines the type of the record to be printed by using the record identification criteria if present, or by comparing its length with the record types in the template.
- When using PRT_OUT with DSC or the Copy Utility (option 3.3) and you have specified both an input and an output copybook or template, the copybook or template used to format the printed record is the output template.
- Only those fields that have been selected in the template are printed.

count FASTREXX only. The maximum number of times this function will be performed. The default is no limit.

Note: Avoid using PRT_OUT in a REXX procedure for:

- DSP function
- FCH function
- Print Utility (option 3.2)

Output from PRT_OUT is interspersed with the normal output (from DSP, FCH, or Print) which can be confusing.

Using PRT_OUT in a REXX procedure that runs from the Find/Change Utility panel (option 3.6) does not result in this problem because the report produced by the panel is sent to a data set, separate from print output.

Example

Print the first 5 records in hex format and print the remainder in tabular format.

```
PRT_OUT('HEX',5)
If recsin() > 5 then PRT_OUT('TABL')
```

PRT_VAR

Syntax

```
▶▶PRT_VAR(name,format,count)◀◀
```

Can be used in FASTREXX condition expressions.

Prints the specified variable in the format specified.

name The name of the variable to be printed. This must be the name of an existing character variable.

format

Format in which the output record is to be printed. Valid values are: CHAR, HEX, SNGL, or TABL.

If you specify TABL or SNGL format:

- On the function or panel that you are enhancing, you must specify a copybook or template that describes the record type to be printed.
- File Manager determines the type of the variable (record) to be printed by using the record identification criteria if present, or by comparing its length with the record types in the template.
- When using PRT_VAR with DSC or the Copy Utility (option 3.3) and you have specified both an input and an output copybook or template, then the copybook or template used to format the printed variable is the input template for any variable other than ZOUTREC. ZOUTREC is formatted with the output record.
- Only those fields that have been selected in the template are printed.

count FASTREXX only. The maximum number of times this function will be performed. The default is no limit.

Note: Avoid using PRT_VAR in a REXX procedure for:

- DSP function
- FCH function
- Print Utility (option 3.2)

Output from PRT_VAR is interspersed with the normal output (from DSP, FCH, or Print) which can be confusing.

Using PRT_VAR in a REXX procedure that runs from the Find/Change Utility panel (option 3.6) does not result in this problem because the report produced by the panel is sent to a data set, separate from print output.

Example

Print the first 5 records in hex format and print the remainder in tabular format.

```
PRT_VAR('ZINREC','HEX',5)
If recsin() > 5 then PRT_VAR('ZINREC','TABL')
```

- For the WRITE function, add this note:

Note: If you have multiple WRITE() statements to more than one file that share the same tape unit via UNIT=AFF then File Manager will perform a close for a file opened by any previous write statement for the shared unit when processing the WRITE() statement to the unit with affinity specified. If the file associated with a WRITE() statement has been closed as a result of a subsequent WRITE() statement to a different file then the latter file cannot be re-opened and if the WRITE() statement for the closed file is processed again the procedure will be terminated with an error message.

Chapter 17, "File Manager messages"

Add the following error message:

FMNBA055 WRITE(&dd) issued that would cause a re-open of a file with UNIT=AFF specified

Explanation: The DD name referenced has been closed due to processing of another WRITE function that

shares the same tape unit. This file cannot be re-opened once it has been closed.

User response: Examine the logic in your procedure and correct the logic so that you are not writing to one file then the next file and then the previous file again.

Changes to the User's Guide and Reference for IMS Data

Chapter 11, "Batch reference"

Subsection: "File Manager functions"

- For the ITU, IVU, and ICU functions, in the description for the keyword MEMLIST, change the following sentence in the description for the variable *member_n* from:

"Generic name masks are not allowed."

to:

"Generic name masks are allowed."

UK49461, UK49462, UK49463, UK49464, UK49465, UK49466, UK49467

Release Date: 31 August 2009

This set of PTFs contains the following APAR fixes:

APAR #	APAR Abstract	Doc Impact
PK83865	File Manager is not handling short of storage conditions correctly for the queue list.	Customization Guide (SC19-2494-00)
PK85062	Receive messages FMNBA091 AND FMNBB291 when number of digits specified for parameter position on DSC is greater than 8.	User's Guide and Reference (SC19-2495-00)

PK83865

Initial problem description

After selecting a queue manager from the Websphere MQ Managers panel, a message "Insufficient Virtual Storage" may occur while generating a list of queues, or a list containing blank elements may be presented which prevents working with the list of queues.

Outline of solution

File Manager has been updated in order to recognize the short of storage condition while listing queues and to issue a short message "Limited results shown" which will allow a filter to be applied to the queue list.

Documentation impact

This APAR requires changes to be made to:

- Customization Guide (SC19-1238-01)

Changes to the Customization Guide

Chapter 2, "Customizing the operating environment for File Manager"

In the section, "Enabling File Manager to work with certain products", subsection "Enabling WebSphere MQ support", in the paragraph:

"For File Manager base function, the WebSphere MQ load libraries SCSQANLE, SCSQAUTH and SCSQLOAD must be made available to the TSO user, either in the linklist, or as part of the STEPLIB in the TSO procedure, or as part of the ISPLLIB concatenation."

- Remove the phrase "or as part of the ISPLLIB concatenation".
- Add:

Note: ISPF LIBDEF ISPLLIB cannot be used to access these libraries.

PK85062
Initial problem description

1. Messages FMNBA091 and FMNBB291 are issued when the number of digits specified in the POSITION parameter of the DSC function is greater than 8.
2. The message IEC036I is displayed followed by an ISPF Dialog error when editing a PDSE load module.

Outline of solution

1. The number of digits which will be accepted when a record number is entered as either a start, skip, limit, include, or count has been increased from 8 to 9.
2. If editing a PDSE load module, the function will be changed from an edit to a browse function.

Documentation impact

- This APAR requires changes to be made to:
- User's Guide and Reference (SC19-2495-00)

Changes to the User's Guide and Reference

Chapter 16, "Functions"

Change the definition of the identified fields in the functions listed below to increase the maximum size to 9 digits (999 999 999):

Function	Fields
DSC	NLRECS
DSG	NLRECS
DSM	CMPOLD, CMPNEW
DSP	NLRECS

UK48204, UK48205, UK48206, UK48207, UK48239, UK48243, UK48254

Release Date: 16 July 2009

This set of PTFs contains the following APAR fixes:

APAR #	APAR Abstract	Doc Impact
PK84077	FM/DB2 fails when using the generate SQL from DB2 catalog function.	Customization Guide (SC19-2494-00) User's Guide and Reference for DB2 (SC19-2496-00)

PK84077

Initial problem description

1. When using the GEN prefix command to generate SQL from a DB2 Catalog and the TSO Execution Mode is specified the message 'Data set not found' is issued.
2. When using the GEN prefix command to generate SQL from a DB2 Catalog and the Batch Execution Mode is specified the message 'JCL Generation failed' is issued.

Outline of solution

1. The correct data set name will be used for the generated SQL.
2. The JCL to generate SQL from a DB2 Catalog will be generated.

Documentation impact

This APAR requires changes to be made to:

- Customization Guide (SC19-1238-01)
- User's Guide and Reference for DB2 (SC19-2496-00)

Changes to the Customization Guide

Chapter 1, "Preparing to customize File Manager"

In the section, "Alternatives for making File Manager available", subsection "Modifying the TSO logon procedure", add the following to the list of File Manager libraries you need to add to your TSO logon procedure:

```
DDNAME SYSPROC: add library FMN.SFMNCLIB
```

Changes to the User's Guide and Reference for DB2

Chapter 2, "Getting started with FM/DB2"

In the section, "Starting and exiting FM/DB2", subsection "TSO region size", add the following sentence to the end of the existing paragraph:

If using the GEN prefix command, you may require more than 8MB of storage.

UK47675

Release Date: 9 July 2009

This set of PTFs contains the following APAR fixes:

APAR #	APAR Abstract	Doc Impact
PK84110	Changing data in a Unicode-encoded table results in SQLCODE+100	User's Guide and Reference for DB2 (SC19-2496-00)
PK90394	AE PK84110 FIX COMPLETION	User's Guide and Reference for DB2 (SC19-2496-00)

PK84110

Initial problem description

In File Manager DB2 component, it is not possible to edit the contents of a DB2 column when the data is stored (by DB2) in Unicode. "SQLCODE+100 row not found" error message.

Outline of solution

This APAR introduces an enhancement to FM/DB2 component only, that allows data in a Unicode-encoded table to be displayed and changed using the FM/DB2 editor. Other FM/DB2 functions are NOT affected by this change. The main features of this enhancement are:

- Data in CHAR, VARCHAR, GRAPHIC, and VARGRAPHIC columns that are Unicode-encoded within DB2 will appear in the FM/DB2 editor in the CCSID of the terminal.
- Any characters that could not be converted to the CCSID of the terminal will appear as unprintable ('.') characters (x'3F' in hexadecimal mode).
- When the data for a Unicode-encoded column is displayed in hexadecimal, the native Unicode representation will be shown, rather than the hexadecimal characters for the converted (displayed) characters.
- Data can be overtyped in the usual way, however data corruption will occur when the original field contains any unconverted Unicode characters.
- Data can also be changed by overtyping the hexadecimal characters in the column. To use this method, the user must enter the Unicode hexadecimal representation of the required characters.
- The Find/Change and related commands operate by converting text data, eg 'ABC' to Unicode prior to making the search or change. Hexadecimal data eg '414243'x is not converted prior to making the search or change, and the hexadecimal characters must therefore be in Unicode, rather than in the CCSID of the terminal.

Documentation impact

This APAR requires changes to be made to:

- User's Guide and Reference for DB2 (SC19-2496-00)

Changes to the User's Guide and Reference for DB2

Chapter 3, "Working with templates", subsection "Handling special data"

Add this new section:

UNICODE data: The FM/DB2 editor handles character data stored in Unicode differently to data stored in EBCDIC or ASCII. (Requires APAR PK84110). The differences are summarized here.

FM/DB2 normally retrieves character data from DB2 with automatic conversion to the CCSID of the FM/DB2 plan, normally CCSID 37. This is US EBCDIC; other EBCDIC CCSIDs may also be used.

When FM/DB2 retrieves character data stored in Unicode (within DB2), this automatic data conversion does not occur. The FM/DB2 editor processes the character data internally in native Unicode format. This means that, prior to display on the terminal, the data is converted from Unicode to the CCSID of the terminal. This may result in conversion errors when a Unicode character has no corresponding code point in the CCSID of the terminal. Any characters that cannot be converted are shown on the display as periods, indicating an unconverted character. When data displayed (and possibly changed) on the terminal is processed, it is converted from the CCSID of the terminal into Unicode, prior to submission to DB2. This conversion will always be successful, since Unicode includes code points for all characters in commonly used CCSIDs.

Data corruption is possible when the displayed data includes a period indicating an unconverted Unicode character, and the data for the column is changed by overtyping. Any periods in the modified data are converted to the Unicode equivalent. Therefore, if the period represents an unconverted Unicode character, that character will be corrupted by the change. When character data is displayed in hexadecimal mode, the hexadecimal characters normally represent the encoding for the displayed character in the CCSID of the terminal. For example, 'A' is represented by 'C1'x in EBCDIC. When the character data is for a Unicode-encoded column, the hexadecimal characters represent the native Unicode data, not the encoding of the displayed character in the CCSID of the terminal. For example, 'A' is represented by '41'x when Unicode data is being displayed.

You can also make changes to the data in a column stored in Unicode by turning hexadecimal display on (HEX ON command) and overtyping the hexadecimal characters directly. When you do this, you should use the Unicode representation of any character, not the terminal's CCSID representation.

Assuming the target column is character data stored within DB2 in Unicode, the FIND, CHANGE, and EXCLUDE commands operate like this:

- Strings such as 'ABC' are converted to Unicode prior to any search or change occurring.
- Hexadecimal strings such as '414243'x are *not* converted to Unicode prior to any search or change occurring. Therefore F 'ABC' and F '414243'x are equivalent commands when the target column contains Unicode data, but not when the target column contains EBCDIC data.

PK90394**Initial problem description**

Dialog error attempting to access FM/DB2 under FM/CICS.

Outline of solution

File Manager has been updated to correct the problem.

Documentation impact

This APAR requires changes to be made to:

- User's Guide and Reference for DB2 (SC19-2496-00)

Changes to the User's Guide and Reference for DB2**Chapter 4, "Viewing and changing DB2 data", subsection
"Handling special data"**

At the end of the "Unicode data" topic, add:

Note: When running FM/DB2 under FM/CICS, the terminal CCSID always defaults to 37 (US EBCDIC).

UK44837

Release Date: 23 March 2009

This set of PTFs contains the following APAR fixes:

APAR #	APAR Abstract	Doc Impact
PK77613	File Manager batch enhancements to support data ageing and other improvements	User's Guide and Reference (SC19-2495-00)

PK77613

Initial problem description

1. Ability to add and subtract from date values.
2. Ability to use variables in FASTREXX procedures.
3. Ability to default packed decimal lengths for all affected functions.
4. Ability to default character lengths for comparisons.
5. Ability to force ISPF stats when copying or updating members.
6. Ability to interpret a start position with reference to an RDW.
7. Ability to set the batch return code for compare processing when a specified number of differences have been found.
8. Panel 'FMNPEQC' error when viewing cut and paste clipboard.

Outline of solution

1. New Mod_date function that can perform arithmetic on date fields described in a copybook or by start, length type.
2. New functions to create and change variables for FASTREXX processing.
3. Packed decimal lengths defaulted for all functions that refer to a packed decimal.
4. New FLDI/O functions default lengths for character comparisons to a literal length.
5. STATS=FORCE option added.
6. INRDW, OUTRDW options added to adjust start locations.
7. RCDIFF keyword added for data set compare.
8. Panel error for clipboard displays addressed.

Documentation impact

- This APAR requires changes to be made to:
- User's Guide and Reference (SC19-2495-00)

Changes to the User's Guide and Reference

Changes in the replacement pages are marked with a vertical change bar in the left margin

Chapter 14, "Panels and fields"

Copy To panel

Replace the section with the pages shown at "Replacement pages for "Copy To panel" section" on page 47.

Find/Change Utility panel

Replace the section with the pages shown at "Replacement pages for "Find/Change Utility panel" section" on page 48.

Chapter 16, "Functions"

DSC (Data Set Copy)

Replace the section with the pages shown at "Replacement pages for "DSC (Data Set Copy)" section" on page 49.

DSM (Data Set Copy), DSP (Data Set Print), DSU (Data Set Update)

Replace the sections with the pages shown at "Replacement pages for "DSM (Data Set Copy)", "DSP (Data Set Print)", "DSU (Data Set Update)" sections" on page 50.

FCH (Find/Change)

Replace the section with the pages shown at "Replacement pages for "FCH (Find/Change)" section" on page 51.

External REXX functions

Replace the whole section with the pages shown at "Replacement pages for "External REXX functions" section" on page 52.

Replacement pages for “Copy To panel” section

Copy To panel

You use the Copy To panel to specify the data set to which you want your copy to be made. This panel comes in three versions, depending on whether or not you have specified a template in the Copy From panel and requested that the output be in an external format.

Panel and field definitions

```

  _Process  _Options  _Help
  -----
Copy from USERID.FMDATA(DATA1)

To Partitioned, Sequential or VSAM Data Set, or HFS file:
Data set name . . . . . FMDATA
Member . . . . . DATA2
Volume serial . . . . .

Processing Options:
Disposition          Execution "/" options      ISPF Packing
 1 1. Old or Reuse   - Replace members          1 1. Asis
 2 2. Mod            - Binary mode, reclen     2. Pack
Stats Option
 1 1. Off
 2 2. Force

Command ==>
F1=Help   F2=Split   F3=Exit   F4=Expand   F7=Backward   F8=Forward
F9=Swap   F10=Actions F12=Cancel

```

Figure 137. Copy Utility: "To" panel (no copybook or template in Copy From panel)

Copy from

Lists the data set or HFS file (directory) from which you are copying data.

To data set/file: Data set/path name

Can be a fully-qualified data set name or a pattern, or a HFS file or directory. The data set name may include a member name or name pattern in parenthesis. If the member is specified here, the associated **Member** field must be empty.

When you specify an HFS file or directory, you must enter a full path name. If the path name is longer than the displayed entry field, press the Expand function key (F4) to display a pop-up window in which you can enter a longer name.

To Data Set: Member

If you specified the name of a partitioned data set (PDS) without including a member name or name pattern in parenthesis in the **Data set name** field, then you can use this field to specify the member name or a member name pattern.

Volume serial

Serial number of the volume which is to contain the copied data set. Required for data sets which are not cataloged.

Disposition

The To data set status:

Old/Reuse

Writes copied records into the output data set, starting from the beginning of the set and replacing any existing records.

Mod Appends the input records to the end of the data set.

Replace members

Replace like-named members in an output partitioned data set.

Binary mode

When processing an HFS file, allows you to specify binary mode (selected) or text mode (unselected).

reclen When processing an HFS file and the **Binary mode** option is selected, records are derived based on the fixed record length specified. The default is 80.

Can be in the range: 1–32760

Stats Determines whether ISPF statistics (if present) for the PDS members being processed are updated:

Blank Update ISPF statistics.

1 Off Do not update ISPF statistics.

2 Force Always update or create ISPF statistics.

Use I/O exit

Allows you to specify a user I/O exit for compressed or encrypted data sets.

This option has two fields. To select the option, enter “/” in the field to the left of Use I/O exit. With this selected, you can then specify which exit to use in the field to the right of the field label.

Notes:

1. The field only displays if File Manager is installed with the option **USEIOX=ENABLE**, and the **Exit enabled** field (in the Set System Processing Options panel) is set to YES. If a default is specified with either of those options, it is displayed in the field to the right of Use I/O exit.
2. I/O exits can only be used to process the data sets that you are using. They cannot be used to process the copybook or template that you are using to format the data set.

ISPF Packing

Provided that the output data set is a sequential, PDS or PDSE file, an I/O exit routine is not used and the DISP is set to OLD, one of these options can be used to control the copy behavior when processing data that is in ISPF PACK format.

1. Asis

If the input data set is packed, it is unpacked before any processing. The output is written in packed format only when the input is packed.

2. Unpack

If the input data set is packed, it is unpacked before processing. The output is always written in unpacked format.

3. Pack

If the input data set is packed, it is unpacked before processing. The output is always written in packed format.

4. None

No checking or processing of ISPF packed data occurs. This option is forced if an I/O exit has been used.

5. Skip

If the input data is packed, no processing or copying occurs.

When you have specified a template in the Copy From panel, the Copy To panel contains additional fields, so that you can choose to specify a template for the output data set.

<u>P</u> rocess	<u>O</u> ptions	<u>H</u> elp
Copy from FMNUSER.EXPORT		Top of data
To Partitioned, Sequential or VSAM Data Set, or HFS file:		
Data set/path name . .	'FMNUSER.DATX'	+
Member name (or mask) .	_____	
Volume serial	_____	
To Copybook/Template From: FMNUSER.DATA(TEMPA)		
Data set name	_____	
Member	_____	(Blank or pattern for member)
Processing Options:		
Copybook/template usage	Disposition	Enter "/" to select option
<u>2</u> 1. Above	<u>1</u> 1. Old or Reuse	- Replace members
2. None	2. Mod	- Edit template mapping
3. Create dynamic	Stats Option	- Edit template source
ISPF Packing	<u>1</u> 1. Off	- Binary mode, reclen _____
<u>1</u> 1. Asis	2. Force	
2. Pack		
3. Unpack		
4. None		
5. Skip		
Command ==>		
F1=Help	F2=Split	F3=Exit
F9=Swap	F10=Actions	F12=Cancel
	F4=CRetriev	F7=Backward
		F8=Forward

Figure 138. Copy Utility: "To" panel (copybook or template specified in Copy From panel)

Copybook/Template From

Lists the copybook or template specified in the Copy From panel, if applicable.

To Copybook or Template: Data set name

Data set name of the template or copybook to be used when mapping fields from the From Copybook or Template. This template does not affect the record selection or field format of the copied data. The field is ignored when the Copybook/template usage field is set to **2. None**.

Note: In the case of a copybook, this can be the name of a CA-Panvalet library, or a library accessed using the Library Management System Exit.

To Copybook or Template: Member

If you specified the name of a partitioned data set (PDS) or CA-Panvalet or other external library in the **Data set name** field, then use this field to specify the member name or member name pattern. This field is ignored when the Copybook/template usage field is set to **2. None**.

Copy To panel

Edit template mapping

Specifies that you want to change the mapping of input fields to output fields or the data creation patterns for new fields.

Edit template source

Specifies that you want to edit the copybook used to generate the template.

If you selected the **Export mode** option on the Copy From panel, the Copy To panel contains additional fields specific for external format that allow you to customize the result of copy.

```

  _Process  _Options  _Help
-----
Copy from FMNUSER.FMDATA

To Partitioned, Sequential or VSAM Data Set, or HFS file:
Data set/path name . . 'FMN.FMDATA' _____ +
Member name (or mask) . _____
Volume serial . . . . . _____

Processing Options:
Disposition          Execution "/" options          Non-print. characters
 1 1. Old or Reuse    - Replace members              2 1. Asis
 2 2. Mod             - Binary mode, reclen         2 2. Hex
ISPF Packing         - Include fillers             3 3. Replace with . _____
 1 1. Asis           - Include redefines           4 4. Skip
 2 2. Pack           - Convert to Unicode           Special characters
 3 3. Unpack         - Split output line           1 1. Escape
 4 4. None
 5 5. Skip
Format              1 1. Off
 1 1. XML            - 2. Force
                    Indent step 1
                    Invalid data
                    1 1. Hex
                    2 2. Replace with * _____
                    3 3. Skip

Command ==>
F1=Help    F2=Split    F3=Exit    F4=Expand    F7=Backward  F8=Forward
F9=Swap    F10=Actions F12=Cancel

```

Figure 139. Copy Utility: "To" panel (Export mode specified in Copy From panel)

Disposition

1. Old or Reuse

Copies from the beginning of the existing data set.

2. Mod

Appends the input records to the end of the data set. MOD is invalid for a member of a partitioned data set.

ISPF Packing

Provided an I/O exit routine is not used, one of these options can be selected to control the copy behaviour when considering data that is in ISPF packed format. (Note that packed options do not apply to VSAM input.)

1. Asis

If the input data set is packed, it is unpacked before any processing. The output is written in packed format if the input was packed.

2. Pack

If the input data set is packed, it is unpacked before processing. The output is written in packed format.

3. Unpack

If the input data set is packed, it is unpacked before processing. The output is written in unpacked format.

4. None

No checking or processing of ISPF packed data occurs. This option is forced if an I/O exit has been used.

5. Skip

If the input data is packed, no processing or copying occurs.

Format

Specifies an external format to be used for the output.

1. XML

The output is generated in XML format.

Execution "/" options**Replace members**

Replace like-named members in an output partitioned data set.

Binary mode

Data in an HFS file is processed without record delimiters (in binary mode).

reclen: The logical record length used to deblock data into fixed records (default: 80).

Include fillers

Indicates whether fillers (unnamed items), defined in the input template (COBOL copybook or PL/I include), are to be represented in the output or ignored.

Include redefines

Indicates whether redefinitions of data items, specified in the input template (COBOL copybook or PL/I include), are to be represented in the output or ignored.

Convert to Unicode

Indicates whether the output is to be converted to Unicode, or not.

Split output line

Indicates whether the output lines resulting from processing an input record are to be spanned contiguously over multiple output records. If so, the output records will not match output lines. If not, each output line must fit (as the only line) into a single output record, otherwise File Manager truncates the output, ends processing, and reports an error.

Use I/O exit

Activates the specified exit routine which handles a compressed or encrypted data set. Specify the name of the exit routine unless the default exit is in effect and can be used.

Indent step

Defines the number of blanks used to indent each level of XML tag nesting (each nested level in the template, COBOL copybook, or PL/I include causes an increase in indentation by the specified number of blanks). Valid range: 0-9 (default:1).

Non-print. characters

Indicates how non-printable characters are to be represented in the output.

1. Asis

Non-printable characters appear unchanged in the output.

2. **Hex** A value with one or more non-printable characters is substituted by its hexadecimal representation.

3. Replace with *replacing-character*

Each non-printable character is substituted with a *replacing character*, or each substring of non-printable characters is converted to its hexadecimal representation and surrounded by nested <HEX> and </HEX> tags. The set of allowable replacing characters is limited to printable characters with the exception of special characters.

You can specify the replacing character in one of the following forms:

char Each non-printable character is replaced with a character, such as "?". Default character: "." (dot).

C'char Each non-printable character is replaced with a character without case translation.

X'cc' Each non-printable character is replaced with a character defined by its hexadecimal value.

HEX If you specify HEX instead of a replacing character, each substring of consecutive special characters is replaced by its hexadecimal representation, tagged by <HEX> and </HEX>, and nested into the content of the element. In other words, each string of consecutive special characters is represented by:

`<HEX>hex-representation-of-string-of-non-printable-characters</HEX>`

nested in the content of the element.

4. Skip

Value is skipped if it contains any non-printable characters.

Special characters

Indicates how special characters are to be represented in XML output.

1. Escape

Special characters are converted into escaped strings:

">" for ">"

"<" for "<"

"'" for "'"

""" for "\""

"&" for "&"

2. CData

The string containing special characters is left unchanged. It is enclosed in the CDATA section.

3. **Hex** A value with one or more special characters is substituted with its hexadecimal representation.

4. Replace with *replacing-character*

Each special character is substituted with a *replacing character*, or each substring of special characters is converted to its hexadecimal representation and surrounded by nested <HEX > and </HEX > tags. If a replacing character is specified or defaulted, each special character is substituted with the replacing character. The set of

allowable replacing characters is limited to printable characters with the exception of special characters.

You can specify the replacing character in one of the following forms:

char Each special character is replaced with a character, such as "?". Default character: "_" (underscore).

C'char' Each special character is replaced with a character without case translation.

X'cc' Each special character is replaced with a character defined by its hexadecimal value.

HEX If you specify HEX instead of a replacing character, each substring of consecutive special characters is replaced by its hexadecimal representation, tagged by <HEX> and </HEX>, and nested into the content of the element. In other words, each string of consecutive special characters is represented by:

`<HEX>hex-representation-of-string-of-special-characters</HEX>`

nested in the content of the element.

Invalid data

Indicates how invalid data is to be represented in the output.

1. Hex Any invalid value is substituted by its hexadecimal representation.

2. Replace with *replacing character*

Any invalid value is replaced with a string of *replacing characters* for the length of the value. The set of allowable characters is limited to printable characters with the exception of special characters.

You can use:

char The value is replaced with a string of characters, such as "?". Default character: "*" (asterisk).

C'char' The value is replaced with a string of characters without case translation.

X'cc' The value is replaced with a string of characters, each character being defined by its hexadecimal value.

3. Skip

Any invalid value is skipped.

Parent panels

- "Copy From panel" on page 471

Child panels

- "Copy From panel" on page 471 (the Copy To data set and template, if specified, was fully qualified)
- "Data Set Selection panel" on page 499 (a pattern has been entered in the Data set name field)
- "Member Selection panel" on page 579 (a pattern or a blank has been entered in the Member field)

Copy To panel

- “Record Type Selection panel” on page 613 (Edit Template is selected and the specified template was based on a copybook with more than one record type).
- “Field Selection/Edit panel” on page 546 (Edit Template is selected and the specified template was based on a copybook with only one record type).
- “Dynamic Template panel” on page 510 (Create Dynamic option is selected or Edit Template is selected and the specified template was created dynamically).
- “Personal Data Set List panel” on page 590 (Current Data Set List option selected from the Process drop-down menu, or REFL fastpath command entered).
- “Personal Data Set Lists panel” on page 592 (Personal Data Set Lists option selected from the Process drop-down menu, or REFD fastpath command entered).

Equivalent functions

“DSC (Data Set Copy)” on page 854

Related tasks and examples

- “Copying data sets” on page 229
- “Supplying a procedure when using a File Manager panel” on page 395

Replacement pages for “Find/Change Utility panel” section

Find/Change Utility panel

The Find/Change Utility allows you to search for or change a string in a PDS, a VSAM data set, or a sequential data set by entering a FIND or CHANGE command on the Command line. You can also search for strings in HFS files.

Panel and field definitions

```

  _Process  _Options  _Help
  _____
File Manager                Find/Change Utility

Input Partitioned, Sequential or VSAM Data Set, or HFS file:
Data set/path name 'FMNUSER.DATA' _____ +
  Member . . . . * _____ (Blank - selection, pattern - process list)
  Volume serial . . _____ (If not cataloged)
  Record count . . ALL _____ (Number of records to be searched)
- Additional options _____
Listing data set . SRCHFOR.LIST _____

Enter "/" to select option
- JCL Source format _____ Immediate change 1 1. Long 1 1. Asis
- Use REXX proc _____ Batch execution 2. Summary 2. Pack
- REXX no update _____ Directory integrity Stats Option 3. Unpack
- Advanced member selection _____ 2 1. Off 4. None
Binary mode, reclen _____ CAPS initially on 2. Force 5. Skip

Process List:
Sel Name Prompt Alias-of Size Created Changed ID
- M100 Selected
- M1000 Selected
- M10000 Selected
- M10001 Selected
- M10003 Selected
- M10005 Selected
- M10006 Selected
- M10007 Selected
- M10008 Selected

Command ==> _____ Scroll PAGE
F1=Help F2=Split F3=Exit F4=CRetriev F5=Refresh F7=Up
F8=Down F9=Swap F12=Cancel

```

Figure 172. Find/Change Utility panel

Data set/path name

Can be a fully-qualified data set name or a pattern, or a HFS file or directory. The name may include a member name or name pattern in parenthesis. If the member is specified here, the associated **Member** field must be empty.

When you specify an HFS file or directory, you must enter a full path name. If the path name is longer than the displayed entry field, press the Expand function key to display a pop-up window in which you can enter a longer name.

Member

If you specified the name of a partitioned data set (PDS) without including a member name or name pattern in parenthesis in the **Data set name** field, then you can use this field to specify the member name or a member name pattern.

Volume serial

Serial number of the volume which contains the data set. Required for data sets which are not cataloged.

Find/Change Utility panel

Record count

Number of logical records to be searched for FIND/CHANGE/FINDNOT commands. For a PDS, the number of logical records to be searched per member.

Range = 1 - 99,999,999; default = ALL.

Additional options

Collapses or expands the following section of the Find/Change Utility panel.

When a minus sign ("-") is shown, position the cursor on the minus sign and press Enter to expand the following section of the panel.

When a plus sign ("+") is shown, position the cursor on the plus sign and press Enter to collapse the following section of the panel.

Listing data set

Specifies the data set where File Manager find/change results are to be stored. Use the default name or enter a sequential data set name.

Default: *'userid.SRCHFOR.LIST'*

JCL source format

Indicates that the data set contains JCL and that the JCL syntax is to be preserved.

If not successful at maintaining the number and size of records, File Manager attempts to rewrite the file:

- More errors are possible in this case. For example, a PDS(E) may run out of room.
- If a logical line is changed and requires more physical records, the file is rewritten. The data in columns 73–80 for new physical records is copied from the last related original physical record.

The file must be non-VSAM and have a fixed record length of 80.

When using the **JCL source format** option, the columns searched are set to 3 through 71, unless the statement is not a JCL statement. A statement is considered to be a JCL statement if it begins with the strings `"/**"` or `"//"`. If the statement does not begin with either of these strings, it is not considered to be a JCL statement in which case any column range specified on the FIND (or CHANGE, respectively) command or preset using the BOUNDS command is honored. If no column range has been specified, the full record is searched.

Use REXX proc

You can use this option to perform either of these actions

- Enter a temporary REXX procedure for one-time use by entering a single asterisk (*). File Manager displays an Edit panel, in which you can create a new REXX procedure.
- Specify the name of the member containing the REXX procedure you want to use. The member must belong to the PDS allocated to ddname FMNEXEC. You can enter any of the following:
 - The name of the member.
 - A member name pattern (other than a single *) to list all matching members. You can then select the required member by entering an S in the **Sel** field. A member name pattern can consist of any characters that are valid in a member name and the following two special pattern characters:

asterisk (*)

Represents any number of characters. As many asterisks as required can appear anywhere in a member name. For example, if you enter *d*, a list of all members in the data set whose name contains " d" is displayed.

percent sign (%)

A place-holding character representing a single character. As many percent symbols as necessary can appear anywhere in a member name. For example, if you enter %%%%, a list of all members in the data set whose name is four characters in length is displayed.

Note: If you select this option but leave the **Use REXX proc** member entry field blank, File Manager displays a member name list. You can then select the required member by entering S in the **Sel** field.

(Also, see "Supplying a procedure when using a File Manager panel" on page 395.)

REXX no update

Allows you to specify that you intend no updates to the FCH data set while executing the utility. This option is valid only when a REXX procedure has been specified and is ignored otherwise. If selected, it forces the allocation of the data set as input only. All updates to the data are ignored.

Advanced member selection

Enter "/" to specify a range of members to be selected rather than a specific or generic member name.

Use I/O exit

Allows you to specify a user I/O exit for compressed or encrypted data sets.

This option has two fields. To select the option, enter "/" in the field to the left of Use I/O exit. With this selected, you can then specify which exit to use in the field to the right of the field label.

Notes:

1. The field only displays if File Manager is installed with the option **USEIOX=ENABLE**, and the **Exit enabled** field (in the Set System Processing Options panel) is set to YES. If a default is specified with either of those options, it is displayed in the field to the right of Use I/O exit.
2. An I/O exit can only be used to process the data set in which you are creating records. It cannot be used to process the copybook or template that you are using to format the data set.

Immediate change

When you use the CHANGE command, the input data set is updated immediately (without displaying the changes in the listing data set).

Batch execution

Creates JCL to reflect the command entered. The JCL is presented in an Edit session which you can edit before submitting.

Batch execution restricts the member selection to the pattern specified in the member field. Batch execution does not produce a pop-up selection

Find/Change Utility panel

panel for member selection. If you leave the member field blank, an asterisk (*) is substituted in the JCL generated. For more information, see "FCH (Find/Change)" on page 990.

	Stats	Determines whether ISPF statistics (if present) for the PDS members being processed are updated:
	Blank	Update ISPF statistics.
	1 Off	Do not update ISPF statistics.
	2 Force	Always update or create ISPF statistics.

Directory integrity

Forces an override of the default PDS(E) member processing method which allows for faster PDS directory access.

This option has significant performance impact. When selected, the members are processed in a way which allows concurrent directory updates as File Manager accesses the members using current directory information.

When not selected, the member processing is performed faster, but may be affected by PDS(E) directory updates, possibly causing I/O errors if the data set is updated concurrently.

Listing Option

Determines the format of the output report.

- 1 A full report, including each record found or changed.
- 2 A summary report providing totals for records processed and strings found and changed.

ISPF Packing

Provided that the data set is a sequential, PDS or PDSE file and an I/O exit routine is not used, one of these options can be used to control the utility's behavior when processing data that is in ISPF PACK format.

1. Asis

If the data set is packed, it is unpacked before any processing. The data set is rewritten in packed format only when it was packed initially.

2. Unpack

If the data set is packed, it is unpacked before processing. The data set is always rewritten in unpacked format.

3. Pack

If the data set is packed, it is unpacked before processing. The data set is always rewritten in packed format.

4. None

No checking or processing of ISPF packed data occurs. The FIND and CHANGE commands operate on the packed data. This option is forced if an I/O exit has been used.

5. Skip

If the data is packed, no processing occurs.

Binary mode

When processing an HFS file, allows you to specify binary mode (selected) or text mode (unselected).

reclen When processing an HFS file and the Binary mode option is selected, records are derived based on the fixed record length specified. The default is 80. Can be in the range: 1–32760.

Available commands

- “BOUNDS primary command” on page 711
- “CAPS primary command” on page 712
- “CHANGE/CX primary command” on page 716
- “FIND/FX primary command” on page 736
- “FINDNOT primary command” on page 744
- “LOCATE primary command” on page 753
- “REFRESH primary command” on page 773
- “RESET primary command” on page 775
- “SELECT primary command” on page 783
- “SORT primary command” on page 788
- “VCONTEXT primary command” on page 803

Parent panels

- “Utility Functions menu panel” on page 676

Equivalent functions

- “FCH (Find/Change)” on page 990

Related tasks and examples

- “Finding and changing data in multiple PDS members” on page 246
- “Selecting a range of PDS(E) members” on page 25

Replacement pages for “DSC (Data Set Copy)” section

DSC (Data Set Copy)

Purpose

Use the DSC function to copy data from any sequential or VSAM data set, PDS, or HFS file (directory) to any other sequential or VSAM data set, PDS, or HFS file (directory). The function's performance when copying PDS(E)s greatly depends on the available storage: larger regions generally result in better performance.

Usage notes

- You can select the records to be copied using:
 - Member name selection criteria
 - Date created selection criteria
 - Date last modified selection criteria
 - User ID selection criteria
 - The start key (VSAM only)
 - The skip field
 - The copy count field
 - A conditional expression defined in the *from* template
- Change data set attributes. File Manager can copy records where the input and output data sets have different record formats, record lengths, or block sizes. The copy process truncates or pads records appropriately. Specify the pad character in the PAD field of the SET function. For details see "SET (Set Processing Options)" on page 1025.
- Copy from field to field. Using both a "From" and a "To" template lets you copy selected fields, change the size and type of fields, and create new fields in the output data set. For details, see "Copying data sets" on page 229.
- Copy to output in external format. The "From" template defines the traditional format of a dataset, but also determines a natural character representation of the data. The result of the generation is an output dataset containing a copy of the input data in an external format (such as XML). For details, see "Generating data in external format - XML representation" on page 239.
- Copy concatenated data sets with like or unlike attributes. Note that, under some conditions (with tape data sets), File Manager may not be able to detect unlike data set attributes and still invoke DFSORT for processing. Such invocation may fail as DFSORT does not allow for unlike concatenation of data sets. In such cases, you can disable the DFSORT with the NOSORT function to allow for successful processing of concatenated datasets with unlike attributes.
- Change ISPF packed format. File Manager can unpack existing packed members or sequential data sets, or can write members or sequential data sets in ISPF packed format.
- These changes are visible in the printed DSC BATCH processing report:
 - Member names are printed as specified on the CPYMBR list (when used).
 - Whenever a member name, its alias or new name (prompt value) contains unprintable characters, an additional line of output is printed below the regular output containing the hexadecimal values of the respective member names.

- Whenever a member was located in the input library and appeared on the CPYMBR list but was not selected for processing because of the member mask or advanced member selection criteria, then it is shown in the processing report as "Not selected" (as opposed to "Not found") and be counted in the "not copied" category (as opposed to "in error").

- For example, with these control cards:

```

$$$FILEM MEMBER=X'5C22',
$$$FILEM CPYMBR=(C'alloclx',
$$$FILEM           x'8289879784a222',
$$$FILEM           autotest,
$$$FILEM           X'84A282')
```

One would see this output:

Member	Newname	Alias	Status	Member Copy Report
alloclx			Not selected	
bigpds			Replaced	
X'8289879784A222'				
AUTOTEST			Not found	
dsb			Not selected	
FMN4688I 0 member(s) copied; 1 replaced; 2 not copied; 1 in error				

- **Member names containing lowercase or unprintable characters:**
 - Member names specified with the CPYMBR, MEMBER, MEMSTART, MEMEND, or MEMOUT keywords may contain lowercase or unprintable characters.
 - To specify a member name containing lowercase or mixedcase characters, use the character literal form of the name surrounded by quotes and preceded with character C. For example, C'aBc'.
 - To specify a member name containing unprintable characters, use the hexadecimal literal form of the name surrounded by quotes and preceded with character X. For example, X'81C283'. Mask characters (their hexadecimal value) may be included within the string.

Note: File Manager supports the copying of Load Modules, when the following conditions are met:

- Your input and output data sets are PDS(E)s.
- Your TSO environment is active (and you can use the TSO authorized program services), or you are running File Manager as program-authorized.
- You have not specified a REXX user procedure.
- You have not specified Start key, Skip or Copy counts.
- You are not using templates.
- You do not request member record counts.

Performance tips

- See "General tips about performance when you use File Manager functions" on page 809. The comments about File Manager using DFSORT technology when performing sequential file I/O are important to DSC performance.
- When you are using DSC to copy members of a PDS(E):

Function reference: DSC

- One DSC default is STATS=ON, which causes the ISPF statistics for each copied member to be updated. This can significantly increase I/O (EXCP) and CPU utilization. To improve performance, consider using STATS=OFF.
- File Manager attempts to use IEBCOPY or an equivalent product to copy members if it can. File Manager using IEBCOPY can significantly reduce I/O and CPU requirements, compared to File Manager not using IEBCOPY. If any File Manager processing of individual records is required, it cannot use IEBCOPY. For example, File Manager cannot use IEBCOPY if:
 - A proc (PROC=) is used.
 - A template or copybook is used.
 - Record counts are requested (RECCOUNTS=YES).
- File Manager does not use IEBCOPY when processing members of a PDS(E) when it detects any member names containing unprintable or lowercase characters since IEBCOPY is not capable of processing such member names. This may negatively affect the performance of the DSC operation.

Options

When you specify the PROC option, you are supplying a DFSORT or REXX procedure that controls the selection and formatting used during the copy function. For more information, see the *proc* parameter below.

Return codes

The default return codes from the DSC function have the following modified meanings:

- | | |
|----|--|
| 1 | No records copied for some of multiple members. |
| 2 | No records copied for any of multiple members. |
| 3 | REXX member selection is in effect but the procedure encountered a RETURN DROP, STOP or STOP IMMEDIATE string. This has been treated as a RETURN string with no arguments. |
| | OR |
| | REXX member selection is NOT in effect but the procedure encountered a RETURN DROP MEMBER, RETURN PROCESS MEMBER string. This has been treated as a RETURN string with no arguments. |
| 4 | No records copied because no records selected (for single member or data set) |
| 4 | No records copied because no members to process |
| 4 | No records copied because input empty |
| 4 | No records copied because member exists and "no replace" option specified |
| 4 | Input data set or member was skipped because it is in ISPF Packed Data format and the "PACK=SKIP" option was specified |
| 8 | REXX non-syntax error encountered while processing records |
| 8 | Line generated in an external format is greater than output record size. |
| 16 | No records copied because input and output physically the same (not applicable to a PDS member) |

- 16 Program Object specified - this is not supported
- 16 Output data set or member in use
- 16 Data set or member open error
- 16 Data set or member not found
- 16 Other input or output error occurred
- 16 Insufficient storage available
- 16 DSC abended
- 16 Input data appears ISPF packed but is not valid.
- 16 Other serious error that stops processing occurred

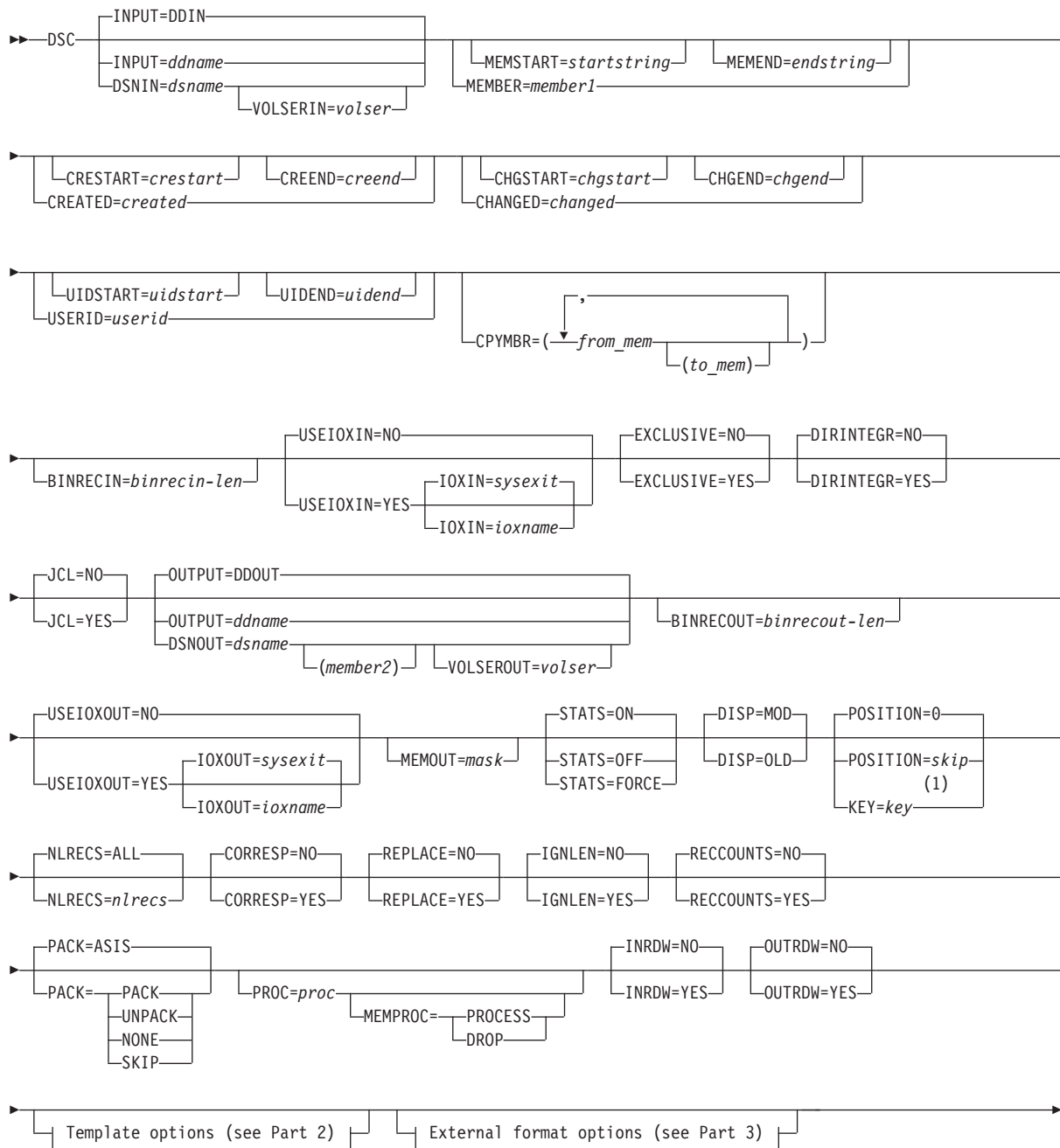
Note: Return codes can be customized during installation. If you receive return codes that do not match those listed above, your site might have customized the return codes in place for this function. File Manager may also issue the 999 abend, if the return code in batch is equal to or greater than the ABENDCC value. Please contact your File Manager systems administrator for details.

Related functions

- OS** Backup objects from an OAM database to a data set
- OV** Backup objects from an OAM database to a VSAM data set
- TS** Copy tape data to a data set
- SO** Copy a data set to an object database
- ST** Copy a data set to tape
- VO** Copy VSAM data to an object database

Function reference: DSC

Syntax: Part 1 of 4

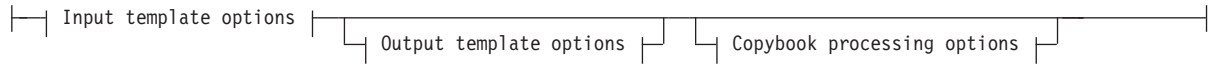


Notes:

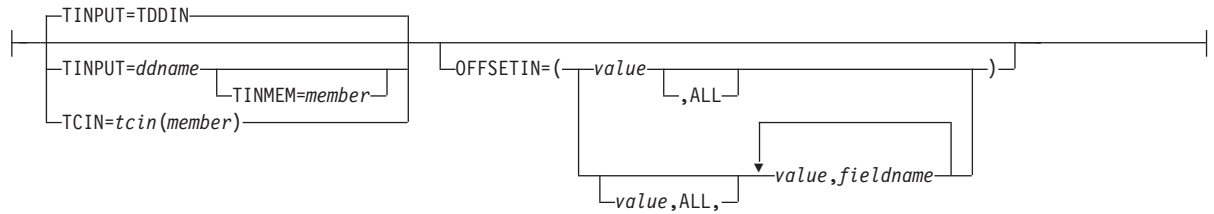
- 1 VSAM only.

Syntax: Part 2 of 4

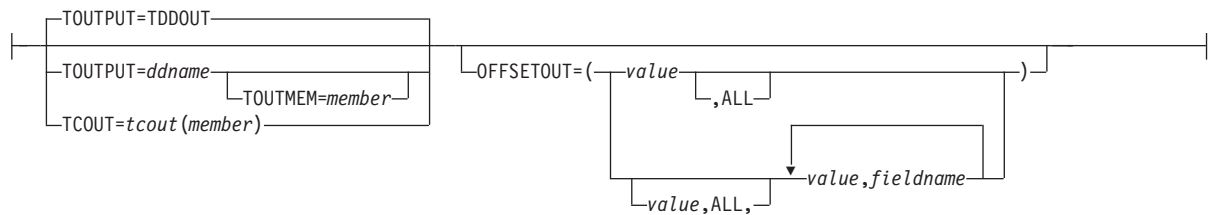
Template options (from Part 1):



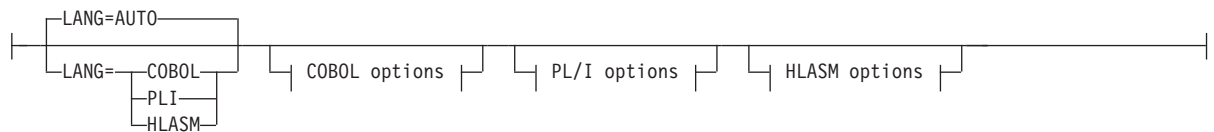
Input template options:



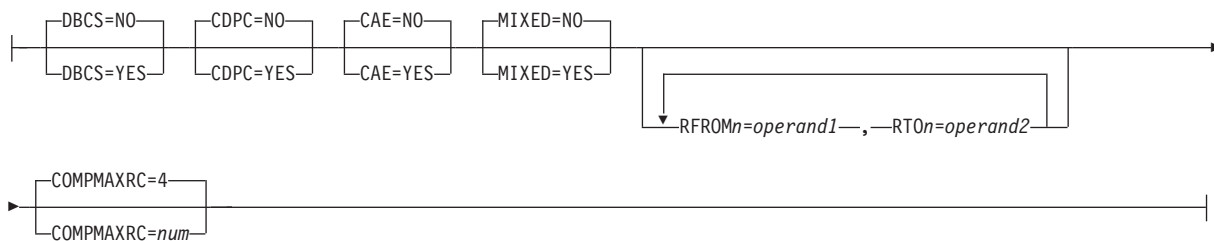
Output template options:



Copybook processing options:



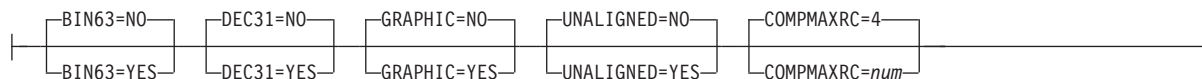
COBOL options:



Function reference: DSC

Syntax: Part 3 of 4

PL/I options:

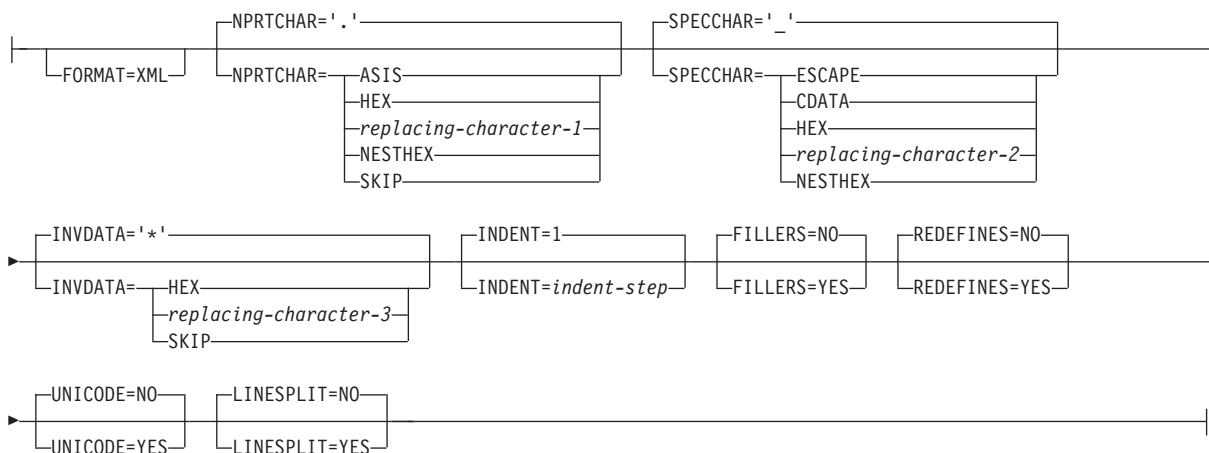


HLASM options:



Syntax: Part 4 of 4

External format options (from Part 1):



INPUT=ddname

Defines a reference to a DD or TSO ALLOC statement for the “From” data set or HFS file. The default is DDIN.

DSNIN=dsname

Defines the name of the “From” data set or an absolute path to the “From” HFS file (directory). If specified, any DD statement provided are not used. The name can include a member name in parenthesis. If the member is specified here, the associated Member parameter must be empty. An absolute path to an HFS file (directory) must be enclosed in apostrophes. If it does not fit on one line, you can split it over more than one line. You can further describe this data set, as follows:

VOLSERIN=volser

Volume serial number for a non-cataloged “From” data set.

MEMBER=member1

The name of a single member in a PDS, or a member name pattern representing one or more members in a PDS. If the input data set is a

PDS(E), you may specify this parameter, or a member name in the DD statement for *ddname*, or specify a member or members in the CPYMBR parameter, or specify a range of member names via the MEMSTART and/or MEMEND keywords.

A member name pattern can consist of any characters that are valid in a member name and two special pattern characters: the asterisk (*) and the percent symbol (%).

* represents any number of characters. As many asterisks as required can appear anywhere in a member name pattern. For example, if you enter a member name pattern of *d*, all members in the PDS whose name contains “d” are processed.

% is a place holding character that means a single character. As many percent symbols as necessary can appear anywhere in a member name pattern. For example, if you enter a member name pattern of %%%%, all members in the PDS whose name is four characters in length are processed.

member1 is ignored if the data set is not a PDS.

Note: See 854.

MEMSTART=*startstring*

Is used to specify the start of a range of member names to be included in the copy. If MEMSTART is specified but MEMEND is omitted, all members of the PDS(E) from the *startstring* value onwards are included. *startstring* can have the same values, including wild cards, as for the *member1* parameter of the MEMBER keyword.

Note: See 854.

MEMEND=*endstring*

Is used to specify the end of a range of member names to be included in the copy. If MEMEND is specified but MEMSTART is omitted, all members of the PDS(E) up to the *endstring* value onwards are included. *endstring* can have the same values, including wild cards, as for the *member1* parameter of the MEMBER keyword.

Note: See 854.

CREATED=*created*

The date on which a member was created, in YYYY/MM/DD format.

If the input data set is a PDS(E), you may specify this parameter, or specify a range of creation dates with the CRESTART and CREEND keywords.

You can specify an asterisk (*) as the last character to indicate a range of dates or a percent sign (%) in place of a single character to indicate a selection of dates.

created is ignored if the data set is not a PDS.

CRESTART=*crestart*

The start of a range of creation dates in YYYY/MM/DD format to be included in the copy.

If CRESTART is specified but CREEND is omitted, all members of the PDS(E) from the *crestart* value onwards are included.

Function reference: DSC

If omitted, or you do not enter a full date, or you specify an asterisk (*) as the last character, the unspecified portion of *crestart* defaults to the right as follows:

DD = 01
MM = 01
YYYY = 0000

No other wildcarding is allowed.

CREEND=*creend*

The end of a range of creation dates in YYYY/MM/DD format to be included in the copy.

If omitted, or you do not enter a full date, or you specify an asterisk (*) as the last character, the unspecified portion of *creend* defaults to the right as follows:

DD = 31
MM = 12
YYYY = 9999

No other wildcarding is allowed.

CHANGED=*changed*

The date on which a member was last modified, in YYYY/MM/DD format.

If the input data set is a PDS(E), you may specify this parameter, or specify a range of modification dates with the CHGSTART and CHGEND keywords.

You can specify an asterisk (*) as the last character to indicate a range of dates or a percent sign (%) in place of a single character to indicate a selection of dates.

changed is ignored if the data set is not a PDS.

CHGSTART=*chgstart*

The start of a range of modification dates in YYYY/MM/DD format to be included in the copy.

If CHGSTART is specified but CHGEND is omitted, all members of the PDS(E) from the *chgstart* value onwards are included.

If omitted, or you do not enter a full date, or you specify an asterisk (*) as the last character, the unspecified portion of *chgstart* defaults to the right as follows:

DD = 01
MM = 01
YYYY = 0000

No other wildcarding is allowed.

CHGEND=*chgend*

The end of a range of modification dates in YYYY/MM/DD format to be included in the copy.

If omitted, or you do not enter a full date, or you specify an asterisk (*) as the last character, the unspecified portion of *chgend* defaults to the right as follows:

DD = 31
MM = 12

YYYY = 9999

No other wildcarding is allowed.

USERID=userid

The TSO user ID by which the member was last updated.

If the input data set is a PDS(E), you may specify this parameter, or specify a range of user IDs with the UIDSTART and UIDEND keywords.

You can enter a generic user ID by using asterisks and percent signs.

userid is ignored if the data set is not a PDS.

UIDSTART=uidstart

The start of a range of user IDs to be included in the copy.

If UIDSTART is specified but UIDEND is omitted, all members of the PDS(E) from the *uidstart* value onwards are included.

If omitted, or you do not enter a full 7-character user ID, or you specify an asterisk (*) as the last character, File Manager replaces the asterisk and pads the unspecified portion of *uidstart* to the right with low values (X'00').

UIDEND=uidend

The end of a range of user IDs to be included in the copy.

If you omit this field, it defaults to high values (X'FF').

If you specify less than 7 characters (without an asterisk as the last character), File Manager pads *uidstart* to the right with low values (X'00'). If you specify an asterisk (*) as the last character, File Manager replaces the asterisk and pads the unspecified portion of *uidend* with high values (X'FF').

CPYMBR

Provides a means of selecting input members from a PDS(E) where no generic name pattern and no member name range has been specified. Also provides a means for renaming the selected members as they are copied to the output data set. If the CPYMBR keyword is specified, only those members included in the CPYMBR arguments are copied to the output data set. Members selected by the MEMBER=*member1* that are not included in the CPYMBR arguments are not copied.

The CPYMBR arguments also refine the member list specified in the MEMBER=*member1* parameter. If the MEMBER keyword is *not* specified, it is assumed to be MEMBER=*, and all members named in the CPYMBR list are processed. However, if the MEMBER keyword is specified, for example as MEMBER=TEST*, the members included in the CPYMBR list are selected from the TEST* subset of members. Any members named in the CPYMBR arguments that do not match the mask given in the MEMBER parameter are not copied.

from_mem

The name of the member to be copied. Generic name masks are not allowed.

to_mem

The name of the member after it has been copied to the output data set. If unspecified, the output member is not renamed.

Note: See 854.

Function reference: DSC

BINRECIN=*binrecin-len*

Specifies the record length used for processing the "From" HFS file. Valid range: 1 to 32760.

The file is processed in Binary mode (fixed-length records derived from the file, delimiters not distinguished). If you do not specify this parameter, the file is processed in Text mode (variable-length records, boundaries determined by delimiters).

USEIOXIN

Specifies whether to invoke a user I/O exit, to process the input data set.

NO Default. Do not invoke a user I/O exit.

YES Invoke a user I/O exit to process the input data set. This option is only available if the person who did the site customization for File Manager allowed user I/O exits on a site-wide basis.

IOXIN

Specifies the name of the user I/O exit used for the input data set. There are no restrictions on the programming language that you can use to write an exit. The exit must be provided to File Manager in the STEPLIB/ISPLLIB concatenation or their extensions (LINKLIST, LPA, and so on).

sysexit Default. If you specify USEIOXIN=YES and do not supply a user I/O exit name, File Manager uses the name of the exit provided in the installation customization options. If USEIOXIN has been set to YES and no installation default has been provided, you must specify IOXIN=*ioxname*.

Note: If you have selected batch processing in an online panel, the generated JCL statements use the default name provided in your Set System Processing Options panel.

ioxname

The name of a PDS(E) member of a data set that has been provided to File Manager in the STEPLIB concatenation.

EXCLUSIVE

Note: This option is supported for backward compatibility only.

Use the new DIRINTEGR option.

Determines the disposition of the From (input) data set.

NO Default. The data set is allocated with DISP=SHR, so that other users can obtain concurrent access to a PDS or PDSE during execution of DSC.

YES The data set is allocated with DISP=OLD, preventing concurrent access to the PDS or PDSE.

Note: If you pre-allocate the input data set with DISP=SHR and then specify EXCLUSIVE=YES in batch, I/O errors might occur during concurrent access to the data.

DIRINTEGR

Specifies whether to invoke a user I/O exit to process the input data set.

NO Default. File Manager uses a faster PDS(E) directory processing

method. This may cause I/O errors when multiple users are concurrently updating the directory of the data set being processed.

YES File Manager uses safer, but slower, PDS(E) directory processing method. This method allows for safe concurrent updates of the PDS(E) directory by multiple users.

JCL=NO

Treat the data set as a non-JCL data set.

JCL=YES

The data set contains JCL and the JCL syntax is to be preserved.

You cannot specify a template with this option.

OUTPUT=ddname

Defines a reference to a DD or TSO ALLOC statement for the “To” data set or HFS file. The default is DDOUT.

DSNOUT=*dsname*

Defines the name of the “To” data set or an absolute path to the “To” HFS file (directory). If specified, any DD statement provided are not used. The name may include a member name in parenthesis. If the member is specified here, the associated Member parameter must be empty. You can further describe this data set, as follows:

(*member2*)

Where DSNOUT=*dsname* specifies a PDS and you want to send the output to a specific member within this data set, this defines the output member name. An absolute path to an HFS file (directory) must be enclosed in apostrophes. If it does not fit on one line, you can split it over more than one line.

VOLSEROUT=*volser*

Volume serial number for a new or non-cataloged “To” data set.

BINRECOUT=*binrecout-len*

Specifies the record length used for processing the “To” HFS file. Valid range: 1 to 32760.

The file is processed in Binary mode (fixed-length records derived from the file, delimiters not distinguished). If you do not specify this parameter, the file is processed in Text mode (variable-length records, boundaries determined by delimiters).

Note: See 854.

USEIOXOUT

Specifies whether to invoke a user I/O exit, to process the output data set.

NO Default. Do not invoke a user I/O exit.

YES Invoke a user I/O exit to process the output data set. This option is only available if the person who did the site customization for File Manager allowed user I/O exits on a site-wide basis.

MEMOUT=*mask*

Where a number of input members have been specified, you can specify a member name pattern for the output members, allowing you to rename your members as they are copied. The member name pattern can consist of any characters that are valid in a member name and two special pattern characters: the asterisk (*) and percent sign (%).

Asterisk (*)

The asterisk is a place-holding character that means multiple characters with no change. Only one asterisk should appear in the mask. Any subsequent asterisk characters are treated as percent signs. For example, if you enter:

ABC*

The renamed members all begin with ABC followed by the remainder of the old member name.

Percent sign (%)

The percent sign is a place-holding character that means a single character with no change. As many percent symbols as necessary may appear anywhere in a member name. For example, if you enter:

%%A*

The 1st 3 characters of the renamed members remain unchanged, the 4th character is replaced with the letter "A" and the remainder of the old member name remains unchanged.

IOXOUT

Specifies the name of the user I/O exit used for the output data set. There are no restrictions on the programming language that you can use to write an exit. The exit must be provided to File Manager in the STEPLIB/ISPLLIB concatenation or their extensions (LINKLIST, LPA, and so on).

sysexit Default. If you specify USEIOXOUT=YES and do not supply a user I/O exit name, File Manager uses the name of the exit provided in the installation customization options. If USEIOXOUT has been set to YES and no installation default has been provided, you must specify IOXOUT=*ioxname*.

Note: If you have selected batch processing in an online panel, the generated JCL statements use the default name provided in your Set System Processing Options panel.

ioxname

The name of a PDS(E) member of a data set that has been provided to File Manager in the STEPLIB concatenation.

STATS=ON

Default. This updates the ISPF statistics (if already present) when a PDS or PDSE member has been changed.

STATS=OFF

The ISPF statistics are not updated when a PDS or PDSE member has been changed.

STATS=FORCE

The ISPF statistics that exist for members being processed are always updated and statistics for a member that previously did not have statistics are created.

DISP

Determines the disposition of the To (output) data set. Specify OLD or MOD.

OLD

Writes input records to the existing output data set, starting from the beginning.

MOD Default. Appends the input records to the end of the existing output data set.

Note: MOD is not available for PDS(E) member processing.

Note: SMS might modify the allocation of new data sets on your system. For details, contact your SMS Administrator.

POSITION=*skip*

Number of logical records to be skipped from the beginning of the data set. The default is 0.

KEY=*key* (**VSAM only**)

A key for KSDS records, or a slot number for RRDS records. The maximum key length is 30 characters. The first record with a key or slot value greater than or equal to *key* is the first record copied. If you omit the *key* and *skip* values, copying begins with the first record in the data set.

If the key contains lowercase characters, blanks, or commas, enclose it in quotation marks. You can also specify a key in hexadecimal format (for example, X'C1C2C3').

NLRECS

Number of records to be copied or ALL.

ALL If you specify ALL or omit the parameter, all the remaining records are copied.

nlrecs The maximum number is 99 999 999.

When you are coding a REXX procedure and NLRECS is specified, then this can affect the number of records presented to the REXX procedure. NLRECS only applies to the number of records written to the primary output data set. It does not apply to records written in the REXX procedure with the WRITE() function.

CORRESP

Specifies whether or not File Manager maps output fields to input fields with the corresponding name. The default is NO.

NO Instructs File Manager to use the existing field mapping in the TCOU member. If the TCOU member is a copybook, or no field mapping is supplied, then File Manager ignores this option and performs a corresponding copy (as if you had specified CORRESP=YES).

YES Instructs File Manager to map output fields to input fields with the corresponding name.

If you want to use existing mapping in the "To" template, specify CORRESP=NO.

REPLACE

Specifies whether or not File Manager replaces like-named members in an output partitioned data set. The default is NO.

NO Like-named members in the output partitioned data set are not replaced.

YES Like-named members in the output partitioned data set are replaced.

Function reference: DSC

IGNLEN

Specifies whether or not File Manager ignores length mismatches when selecting records for processing.

NO Do not ignore length mismatches. Records that are shorter than the matching structure length in the template are not selected for processing.

YES Use this option to ignore length mismatches.

When a field in the "From" template spans or is beyond the copied record's boundary, the corresponding field on the output record is initialized (since there is no data available from the from field). The exception is alphanumeric fields, where the portion of the field that exists on the input record is copied (a partial copy) and the remainder of the output field is padded with blanks.

RECCOUNTS

Controls whether or not the count of records for copied PDS(E) members and sequential/VSAM data sets is printed in the processing listing in batch.

NO Record counts are not reported.

YES Record counts are reported.

Note: This option affects PDS(E) processing ONLY. For sequential/VSAM data sets, the record counts are always provided. When the option is selected, it prevents the use of IEBCOPY for PDS(E) processing, which may affect the copy performance.

PACK Determines if File Manager should detect if the input data is in ISPF packed format and specifies if the output data is to be written in ISPF packed format. This keyword is ignored when processing VSAM data sets. When an I/O exit has been specified for either the input or output data set (or both), the only valid option is PACK=NONE.

ASIS Instructs File Manager to write the output in ISPF Packed format only if the input is in ISPF packed format.

PACK Instructs File Manager to write the output in ISPF packed format regardless of the input format.

UNPACK

Instructs File Manager to write the output without ISPF packing, regardless of the input format.

NONE

Instructs File Manager not to determine if the input data set is in ISPF packed format and writes the output records as they are read from the input data set (after any enhanced processing).

SKIP Instructs File Manager to determine if the input data set is in ISPF packed format and if so, to skip the copy processing.

PROC=*proc*

Member name of a REXX procedure that you want to use to process each record before it is copied, or an asterisk (*) to indicate the procedure is inline. If you specify a member name, you must define an FMNEXEC ddname that identifies the PDS containing the member. If you specify *, the procedure is read from SYSIN immediately following the control

statement for the current function. The inline procedure is terminated by a record containing a slash and a plus sign (/+) in columns 1–2.

For more information about using DFSORT or REXX procedures to process records before they are copied, see Chapter 13, “Enhancing File Manager processing,” on page 385.

If PROC=*proc* is specified, you can then choose to include a MEMPROC parameter:

MEMPROC

Specifies that REXX member selection is in effect. Records are read from the input member and then cached in memory until a decision is made, within the REXX procedure, on whether the member is to be copied or dropped. Once the decision has been made, the entire member is either copied or dropped, depending upon the RETURN string specified in the REXX procedure.

If the entire member is processed without encountering a RETURN DROP MEMBER or RETURN PROCESS MEMBER string, the member is processed according to the action specified by the parameter specified for MEMPROC. These are:

PROCESS

The member is to be included in the copy. The member is copied intact, subject to any specified template processing, which is performed before the user REXX proc is invoked.

This is the default action, if no parameter is specified with the MEMPROC keyword.

DROP The member is to be excluded from the copy. Processing continues with the next member.

INRDW

Controls whether or not to adjust the input start location when the specified start location takes into account the record descriptor word (RDW).

NO Does not adjust the input start location.

YES Subtracts 4 from all start locations that have been coded on external functions that refer to the input record.

OUTRDW

Controls whether or not to adjust the output start location when the specified start location takes into account the record descriptor word (RDW).

NO Does not adjust the output start location.

YES Subtracts 4 from all start locations that have been coded on external functions that refer to the output record.

Template options (Part 2 of syntax diagram)

The template options define which templates (if any) are used to describe the record structure in the “From” and “To” data sets, and how File Manager processes those templates.

TINPUT=*ddname*

Defines a reference to a DD or TSO ALLOC statement for the data sets which contain the copybook or template that describes the record structure of your input data. The default is TDDIN.

Function reference: DSC

If you specify a concatenated DD, then you must provide the member name, *member*.

Note: When you specify concatenated data sets in the template DD statement, and these data sets are vendor-managed copybook libraries, a maximum of 20 data sets are supported.

TINMEM=*member*

The name of the copybook or template member in the datasets identified by the TINPUT parameter if it has not been specified on the DD statement. This parameter must not be specified if the TCIN parameter is specified.

TCIN=*tcin(member)*

PDS and member name of the copybook or template that describes the record structure of your input data.

OFFSETIN

The length of the 01 field in the "From" template and the start locations of the fields within that 01 field are adjusted by the value provided.

value The offset value, which must be in the range -32760 to 32760, to be applied to the corresponding field identifier. If no field identifier is supplied and ALL is not used, the value is applied to the first Level 01 field in the "From" template.

ALL Where the template contains multiple record structures, this keyword applies the corresponding *value* to all Level 01 within the "From" template.

Note: You can specify a value for ALL and then override this value for individual layouts by providing subsequent *value* and *fieldname* combinations.

fieldname

The name of the Level 01 field to which *value* is to be applied. The default is the first Level 01 field in the "From" template.

TOUTPUT=*ddname*

Defines a reference to a DD or TSO ALLOC statement for the data sets which contain the copybook or template that describes the record structure of your output data. The default is TDDOUT.

If you specify a concatenated DD, then you must provide the member name, *member*.

TOUTMEM=*member*

The name of the copybook or template member in the datasets identified by the TOUTPUT parameter if it has not been specified on the DD statement. This parameter must not be specified if the TCOUT parameter is specified.

TCOUT=*tcout(member)*

PDS and member name of the copybook or template that describes the record structure of your output data.

OFFSETOUT

The length of the 01 field in the “To” template and the start locations of the fields within that 01 field are adjusted by the value provided.

value The offset value, which must be in the range -32760 to 32760, to be applied to the corresponding field identifier. If no field identifier is supplied and ALL is not used, the value is applied to the first Level 01 field in the “To” template.

ALL Where the template contains multiple record structures, this keyword applies the corresponding *value* to all Level 01 fields within the “To” template.

Note: You can specify a value for ALL and then override this value for individual layouts by providing subsequent *value* and *fieldname* combinations.

fieldname

The name of the Level 01 field to which *value* is applied. The default is the first Level 01 field in the “To” template.

Copybook processing (Part 2 of syntax diagram)

If you specify a copybook (instead of an existing template), then File Manager uses these processing options to compile the copybook into a template:

LANG

Determines whether File Manager automatically detects the copybook language or interprets the language as COBOL, PL/I, or HLASM.

AUTO

Automatically detect whether the copybook language is COBOL or PL/I, and invoke the appropriate compiler. If the compilation results in a return code greater than 4, then invoke the compiler for the other language. If the second compilation also results in a return code greater than 4, then retry the first compiler and report the compilation errors. If File Manager successfully creates a template (despite the compilation errors), then continue processing with the template.

COBOL

Invoke the COBOL compiler to create a template from the copybook. (Do not invoke the PL/I compiler, even if the COBOL compilation results in errors.)

PLI

Invoke the PL/I compiler to create a template from the copybook. (Do not invoke the COBOL compiler, even if the PL/I compilation results in errors.)

HLASM

Invoke the HLASM compiler to create a template from the copybook.

COBOL options

The following options are used to compile a COBOL copybook into a template:

Function reference: DSC

DBCS=YES

Use the DBCS compiler option.

DBCS=NO

Use the NODBCS compiler option.

For details on the effect of the DBCS and NODBCS compiler options, see the *IBM COBOL Programming Guide for OS/390 & VM*.

CDPC=NO

Do not use the COBOL SPECIAL-NAMES paragraph "Decimal-point is comma".

CDPC = YES

Use the COBOL SPECIAL-NAMES paragraph "Decimal-point is comma".

CAE=NO

Do not use the COBOL compile option ARITH(EXTEND).

CAE = YES

Use the COBOL compile option ARITH(EXTEND).

MIXED = NO

Field names stored in the template in uppercase.

MIXED = YES

Field names stored in the template in the original case as coded in the COBOL copybook.

RFROM1 RTO1 ... RFROM5 RTO5

Up to five pairs of "From" and "To" pseudo-text character strings for the COBOL REPLACE compiler-directing statement.

If your COBOL copybooks contain characters that you want to remove or replace with other characters before compiling the copybooks into templates, then use these replacing options.

For example, if your copybooks contain colon characters (:) that you want to remove before compiling, then specify '==:==' as *operand1* and '====' as *operand2*.

For details on specifying "From" and "To" strings for COBOL REPLACE, see the *IBM COBOL Language Reference*.

COMPMAXRC

Sets the maximum acceptable return code for a copybook compile. A return code higher than the specified level causes the function to stop. Default is 4.

PL/I options

The following options are used to compile a PL/I copybook into a template:

BIN63=YES Use the LIMITS(FIXEDBIN(63)) compiler option.

BIN63=NO Use the LIMITS(FIXEDBIN(31)) compiler option.

DEC31=YES Use the LIMITS(FIXEDDEC(31)) compiler option.

DEC31=NO Use the LIMITS(FIXEDDEC(15)) compiler option.

GRAPHIC=YES

Use the GRAPHIC compiler option.

GRAPHIC=NO

Use the NOGRAPHIC compiler option.

UNALIGNED=YES

Use the DEFAULT RANGE (*) UNALIGNED, language statement to change the default alignment.

UNALIGNED=NO

Use the PL/I default.

COMPMAXRC

Sets the maximum acceptable return code for a copybook compile. A return code higher than the specified level causes the function to stop. Default is 4.

For details on the effect of these compiler options, see the *IBM VisualAge PL/I for OS/390 Programming Guide*.

HLASM options

The following options are used to compile a HLASM copybook into a template:

DBCS=YES Use the DBCS compiler option.

DBCS=NO Use the NODBCS compiler option

NOALIGN=YES

Use the NOALIGN compiler option.

NOALIGN=NO

Use the ALIGN compiler option.

COMPMAXRC

Sets the maximum acceptable return code for a copybook compile. A return code higher than the specified level causes the function to stop. Default is 4.

For details on the effect of these compiler options, see the *HLASM V1R5 Programmer's Guide*.

External format processing (Part 3 of syntax diagram)

When generating the output in an external format, File Manager uses the following options:

FORMAT Specifies the external format to be used for output.

XML Indicates that XML format is used.

NPRTCHAR Determines how non-printable characters are to be represented in the output.

'.' (dot)

Default. Each non-printable character is replaced with ".".

ASIS Non-printable characters appear unchanged in the output.

HEX Any non-printable character is converted into its hexadecimal representation.

'replacing-character-1'

Each non-printable character is replaced with the replacing-character-1. The set of allowable

characters is limited to printable characters with exception of special characters. You may specify:

char A character, such as "?".

C'char' A character used without case translation.

X'cc' A character defined by its hexadecimal value.

NESTHEX

Each string of consecutive non-printable characters will be nested into content of element as *X'hex-representation-of-string-of-non-printable-characters'*

SKIP Any non-printable character causes the value to be skipped (data is represented by start and end tags, without any content).

SPECCHAR Determines how special characters are to be represented in the XML output.

'_' (underscore)

Default. Each special character is replaced with "_".

ESCAPE

Special characters are converted into escaped strings:

">" for ">"

"<" for "<"

"'" for "'"

""" for "\""

"&" for "&"

CDATA

Unchanged string containing special characters are enclosed into the CDATA section.

HEX Any special character will cause the XML value to be converted into its hexadecimal representation.

'replacing-character-2'

Each special character is replaced with the replacing-character-2. The set of allowable characters is limited to printable characters with exception of special characters. You may use:

char A character, such as "?".

C'char' A character used without case translation.

X'cc' A character defined by its hexadecimal value.

NESTHEX

Each string of consecutive special characters is nested into content of element as *X'hex-representation-of-string-of-special-characters'*

INVDATA Determines how invalid data is to be represented in the output.

	'*' (asterisk)	Default. Invalid data is represented by string of "*" with a length equal to the assumed length of the output value.
	HEX	Invalid data causes the output value to be a hexadecimal representation of the input value.
	'replacing-character-3'	Invalid data is represented by a string of <i>replacing-character-3</i> with a length equal to the assumed length of the output value. The set of allowable characters is limited to printable characters with exception of special characters. You can use: <ul style="list-style-type: none"> <i>char</i> A character, such as "?". <i>C'char'</i> A character used without case translation. <i>X'cc'</i> A character defined by its hexadecimal value.
	SKIP	Invalid data is skipped (data is represented by start and end tags, without any content).
INDENT		Specifies the number of blanks used to indent each level of XML tag corresponding to the nested level in the template or copybook. <ul style="list-style-type: none"> 1 Default. Each nested level causes an increase in indentation of each XML level by one blank. <i>indent-step</i> Any value from 0 to 9. INDENT=0 will force no indentation. Each positive number will cause an increase in indentation of each XML level by this number of blanks.
FILLERS		Indicates whether fillers (unnamed data elements) are to be included into the output or not. <ul style="list-style-type: none"> NO Default. Fillers are ignored (not represented in the output). YES Fillers are treated as named data elements and represented in the output.
REDEFINES		Indicates whether data elements redefining other data elements are to be included into the output or not. <ul style="list-style-type: none"> NO Default. Redefines are ignored (not represented in the output). YES Redefines are treated as other data elements and represented in the output.
UNICODE		Indicates whether the output is to be converted to Unicode or not. <ul style="list-style-type: none"> NO Default. The output is not converted. YES The output is converted to Unicode.
LINESPLIT		Indicates, whether the output lines resulting from

Function reference: DSC

processing an input record are spanned contiguously over multiple output records, or each output line must be included as the only line in an output record.

- NO** Default. Output line is contained, as the only output line, in one output record.
- YES** Output records are cut independently of external formatting. An output line can span multiple output records and not necessarily start from the beginning of the record. However, output representation of each input record starts from the new output record.

Note: You cannot specify different options for compiling “From” and “To” copybooks; the same copybook options are used for both.

Batch example

```
//DSC JOB (acct),'name'  
/* Copy data set to data set  
/*  
//FMBAT PROC  
//FMBAT EXEC PGM=FILEMGR  
//STEPLIB DD DSN=FMN.SFMNMOD1,DISP=SHR  
//SYSPRINT DD SYSOUT=*  
//SYSABEND DD SYSOUT=*  
//SYSUDUMP DD SYSOUT=*  
// PEND  
/*  
//STPSSEX EXEC FMBAT  
//SYS1INP DD DISP=SHR,  
// DSN=SYS1.PARMLIB(LNKLIST00)  
//SYSIN DD *  
$$FILEM VER  
$$FILEM DSC INPUT=SYS1INP,  
$$FILEM DSNOUT=FMNUSER.TEMP.LINKLIST,  
$$FILEM TCIN=FMNUSER.FMN.TEMPLATE(ODOTW001)  
$$FILEM EOJ  
/*
```

Batch example - renaming members

This example copies all members of the input PDS ('USERID.PLXIN') that match the input mask 'FMNE*' to the output PDS('USERID.PLXOUT'), renaming them using the rename mask JBG*. The members FMNEDIT, FMNEDIT1, FMNEDIT2, FMNEDIT3, FMNEDIT4, FMNEDIT5 and FMNEDIT6 are copied and renamed as JBGEDIT, JBGEDIT1, JBGEDIT2, JBGEDIT3, JBGEDIT4, JBGEDIT5 and JBGEDIT6 respectively.

```
//SYSIN DD *  
$$FILEM DSC DSNIN=USERID.PLXIN,  
$$FILEM MEMBER=FMNEDIT*,  
$$FILEM DISP=MOD,  
$$FILEM DSNOUT=USERID.PLXOUT,  
$$FILEM MEMOUT=JBG*
```

Batch example - copying PDS(E)s using a TSO environment

This example shows copying PDS(E)s using a TSO environment (required for load module processing when File Manager is not APF-authorized and recommended for better performance with PDS(E)s).

```
//FMBAT EXEC PGM=IKJEFT01,DYNAMNBR=100
//STEPLIB DD DSN=FMN.SFMNMOD1,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
CALL *(FMNMAIN)
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
$$FILEM DSC DSNIN=TEST.PDS,
$$FILEM MEMBER=*,
$$FILEM REPLACE=YES,
$$FILEM DISP=OLD,
$$FILEM DSNOUT=TEST2.PDS
```

Replacement pages for “DSM (Data Set Copy)”, “DSP (Data Set Print)”, “DSU (Data Set Update)” sections

DSM (Data Set Compare)

Purpose

Use the DSM function to:

- Compare data from any partitioned, sequential or VSAM data set, or HFS file to data in any other partitioned, sequential or VSAM data set, or HFS file.
- Perform a field level comparison. By using an “Old” copybook or template with a “New” copybook or template, you can compare selected fields with the result of the comparison reflecting the types of data in the fields.
- Perform a load module comparison. Load module and CSECT information from both the “Old” and “New” versions of the module is extracted and compared. By specifying compare criteria, you can see differences between specific properties of the load modules, such as load module size, link date, CSECT names, and compilers used.
- Produce a comparison report, showing information such as where insertions, deletions or changes have occurred in the “New” data set. The report’s content and structure reflects the various comparison options used.
- Create output data sets containing records identified as inserted, deleted, old and new changed records, and old and new matched records. (These data sets can only be created if the synchronization option, **SYNCH=READAHEAD**, has *not* been specified.)

Usage Notes

- Select the records to be compared, using:
 - The start key (VSAM only)
 - The skip field
 - The compare count field
 - Conditional expressions defined in the “Old” and/or “New” templates.
 - The “Number of differences to report” optionTo perform a field level comparison, you must provide an “Old” and a “New” copybook or template and use the **TYPE=FORMATTED** comparison option. You can use the field mapping specified in a “New” template (created online), or you can use the default mapping generated from the template or copybook contents, or you can specify the field mapping in the batch file, using the **FIELDOLD** and **FIELDNEW** keywords.
- Specify the way in which the comparison is performed, using:
 - The compare options
 - The synchronization optionsIf **SYNCH=KEYED** is used, up to sixteen key segments can be specified to create a single composite key.
- Specify the type of output produced and the way in which the output is displayed, using:
 - The listing type
 - The listing options

Performance tips

- DSM was designed with a focus on comparing data in fields using templates or copybooks. See the template performance tips in “General tips about performance when you use File Manager functions” on page 809.
- The ISPF utility SuperC may perform more efficiently when comparing ordinary text data sets, since the special features of File Manager are not required. For more details on SuperC, see the *z/OS ISPF User's Guide Vol II*.

Return codes

The default return codes from the DSM function have the following modified meanings:

- | | |
|---|---|
| 0 | The function was completed successfully and the compare sets match. |
| 1 | The function was completed successfully and the compare sets do not match. |
| 2 | One of the compare sets was empty, so no comparison was performed. |
| 4 | Both of the compare sets were empty, so no comparison was performed. |
| 4 | No comparison was performed because one of the input data sets or members in ISPF Packed Data format and the “PACK=SKIP” option was specified. |
| 4 | At least one record with an unmapped type was encountered. |
| 8 | A data error occurred, for example, a key sequence error was found when using a keyed comparison. The conditions that result in a return code of 8 are: |

For keyed synchronization:

key truncation error

A key truncation error occurs when a key segment falls outside the record.

key sequence error

A key sequence error occurs when the key for a record is found to be less than or equal to the key for the previous record.

For read-ahead synchronization:

read-ahead resynchronization failure

Read-ahead resynchronization fails when matching records cannot be found within the read-ahead limit.

- | | |
|----|---|
| 16 | No records compared because input and output physically the same. |
| 16 | Invalid data in template. |
| 16 | Data set or member in use. |
| 16 | Data set or member open error. |
| 16 | Data set or member not found. |
| 16 | Other input or output error occurred. |

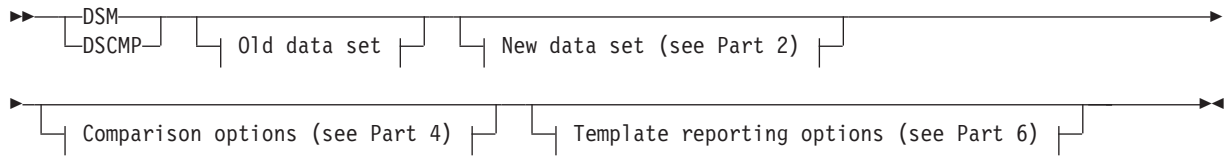
Function reference: DSM

- 16 Member name required and not specified.
- 16 Insufficient storage available.
- 16 DSM abended
- 16 Input data appears ISPF packed but is not valid.
- 16 Other serious error that stops processing occurred.
- 16 A severe error occurred, causing File Manager to terminate.

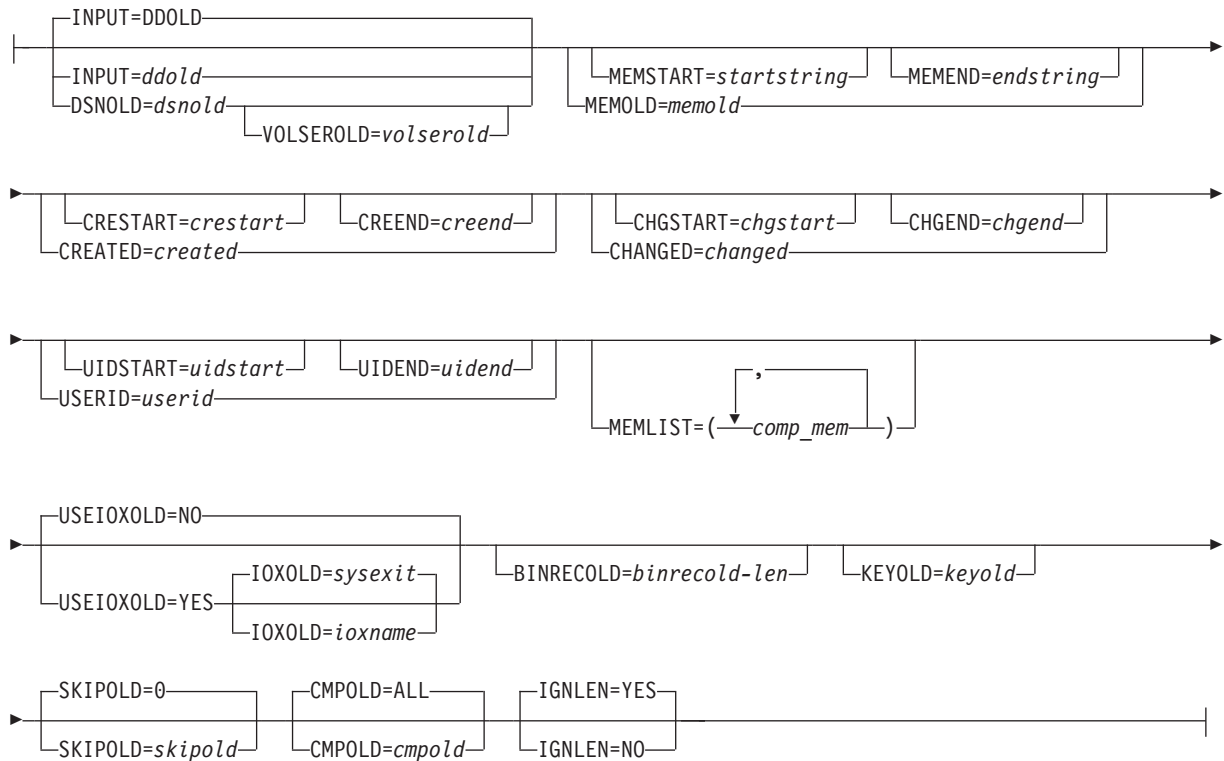
Tip: When handling return codes 0, 1, 2, and 4 in your JCL, you might choose to deal with each code separately or you might choose to deal with return codes 0 and 4 as a single result (the compare sets match) and 1 and 2 as another result (the compare sets do not match).

Note: Return codes can be customized during installation. If you receive return codes that do not match those listed above, your site might have customized the return codes in place for this function. File Manager may also issue the 999 abend, if the return code in batch is equal to or greater than the ABENDCC value. Please contact your File Manager systems administrator for details.

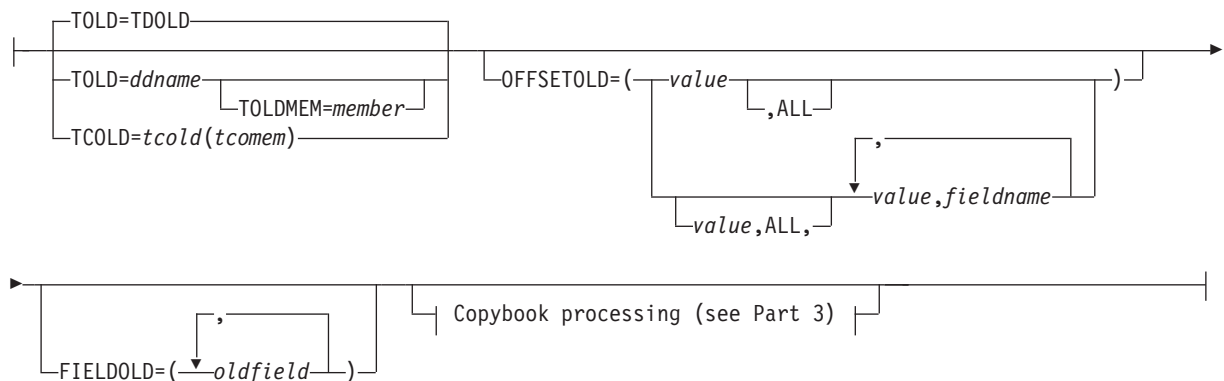
DSM Syntax: Part 1 of 7



Old data set:



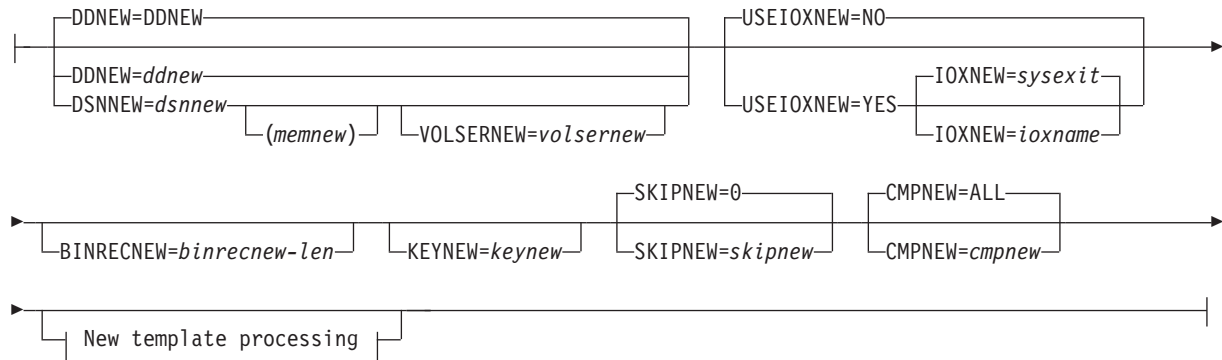
Old template processing:



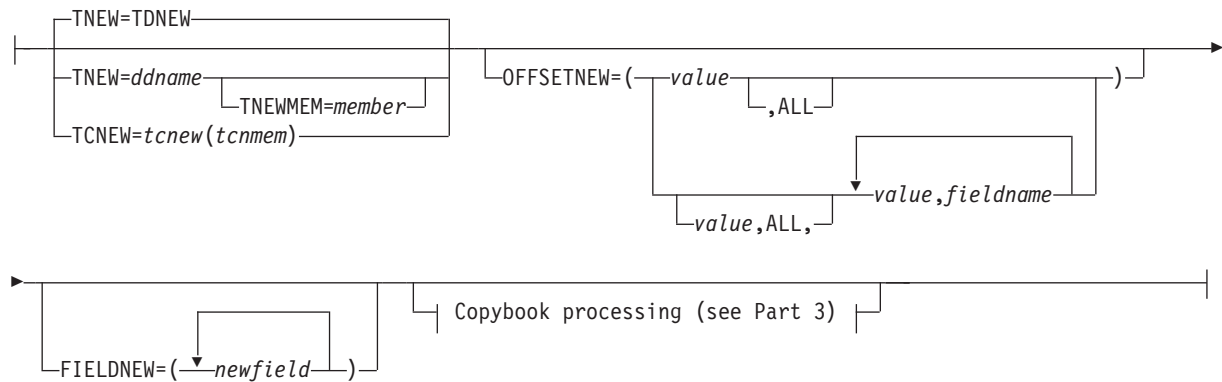
Function reference: DSM

DSM Syntax: Part 2 of 7

New data set (from Part 1):

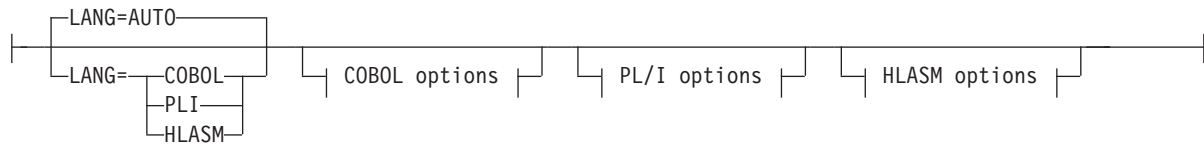


New template processing:

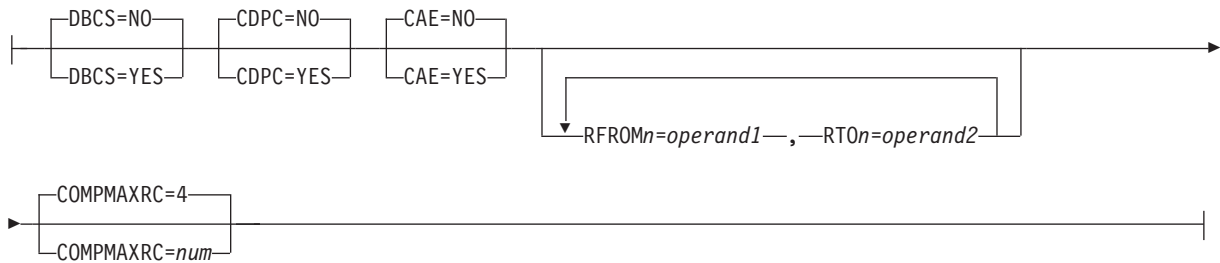


DSM Syntax: Part 3 of 7

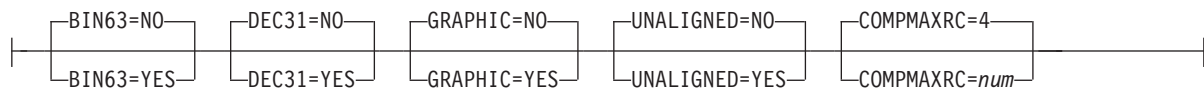
Copybook processing (from Part 1 and Part 2):



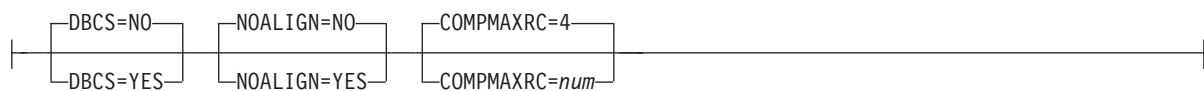
COBOL options:



PL/I options:



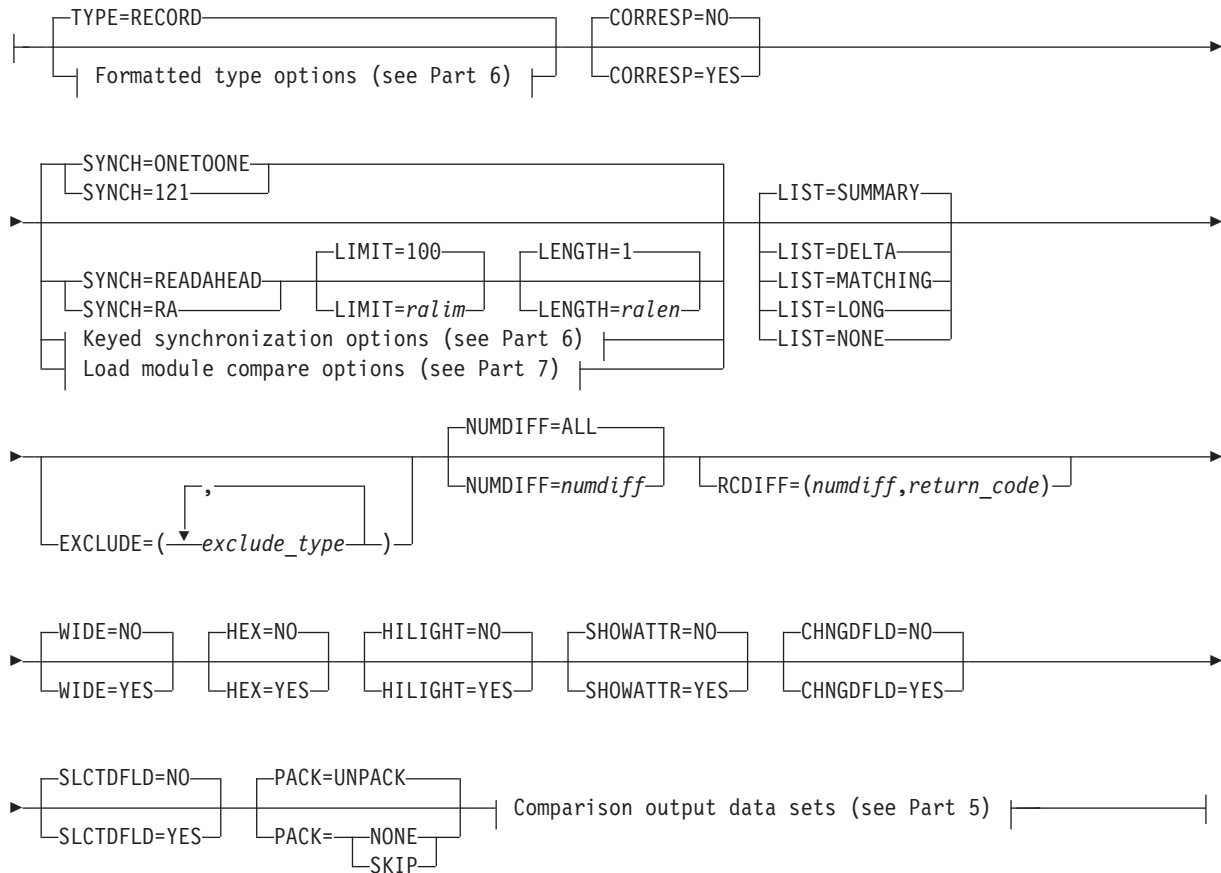
HLASM options:



Function reference: DSM

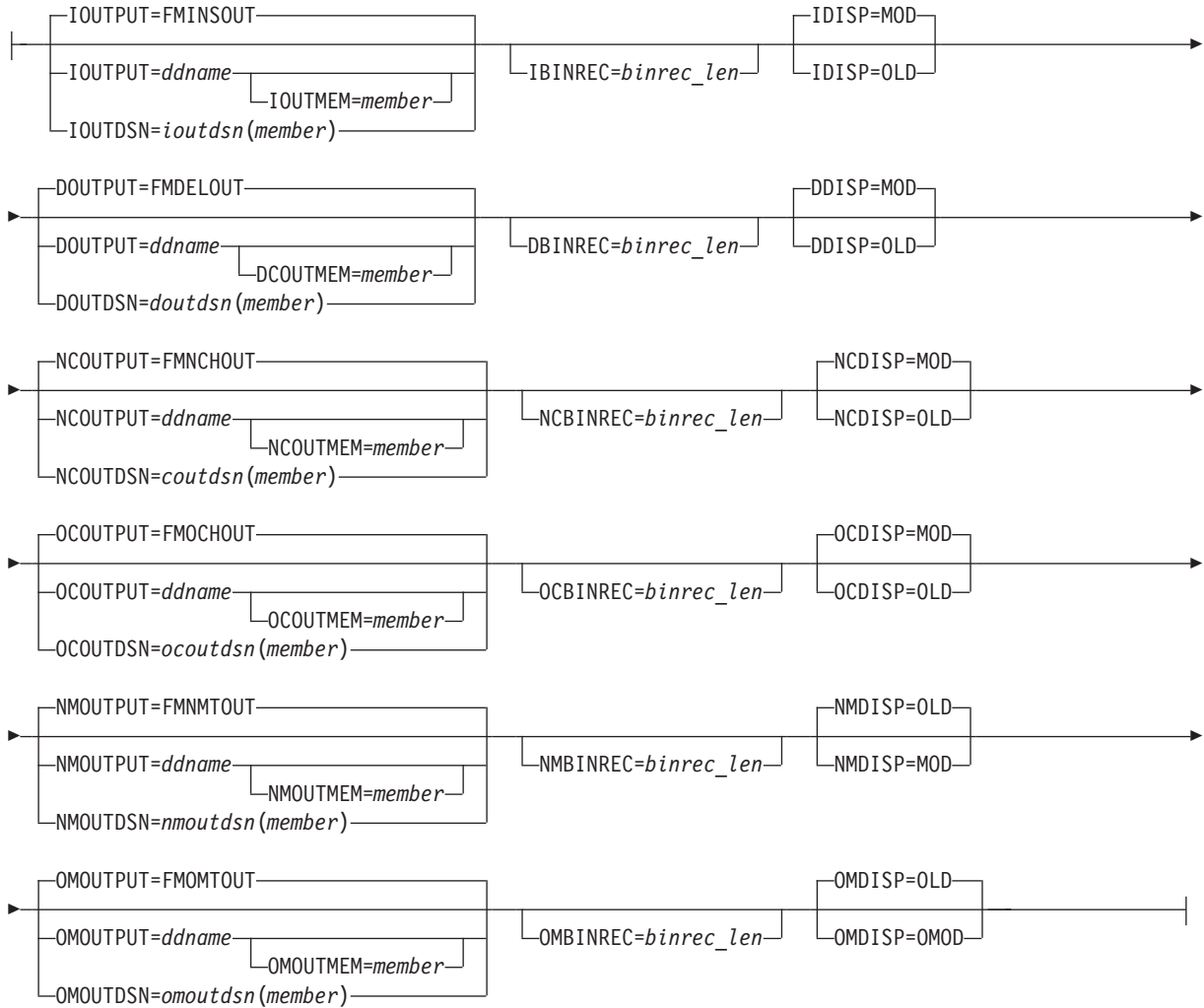
DSM Syntax: Part 4 of 7

Comparison options (from Part 1):



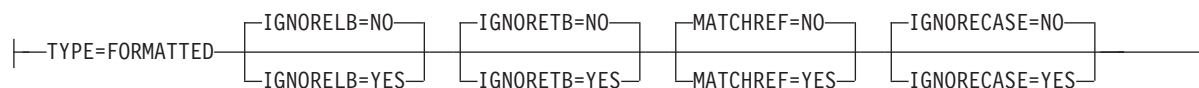
DSM Syntax: Part 5 of 7

Comparison output data sets (from Part 4):

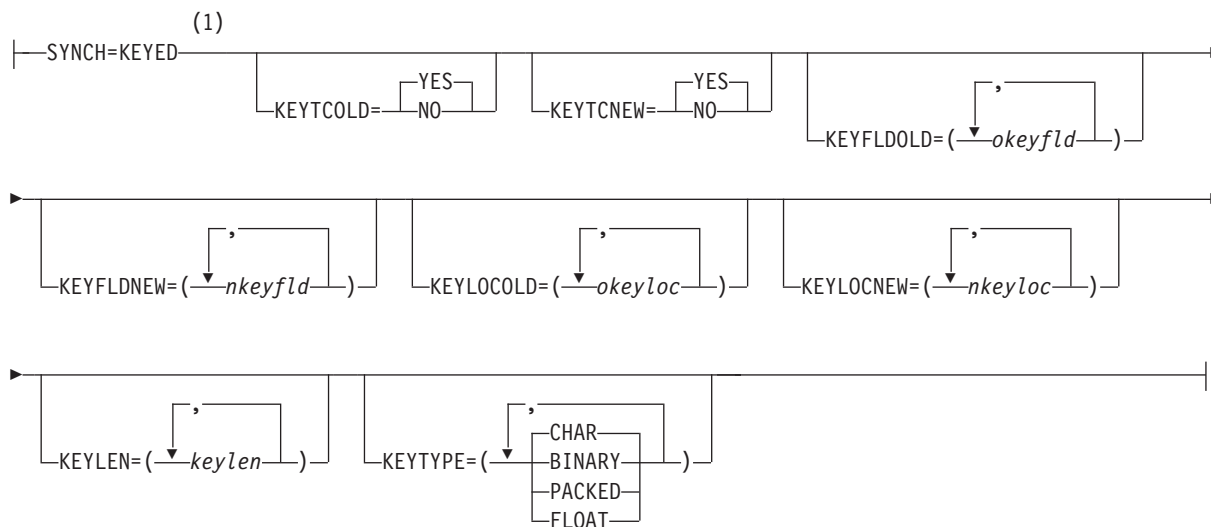


DSM Syntax: Part 6 of 7

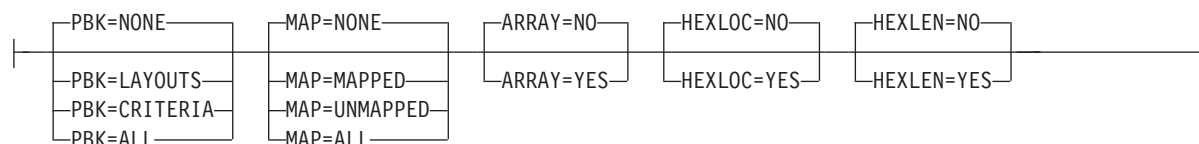
Formatted type options (from Part 4):



Keyed synchronization options (from Part 4):



Template reporting options (from Part 1):

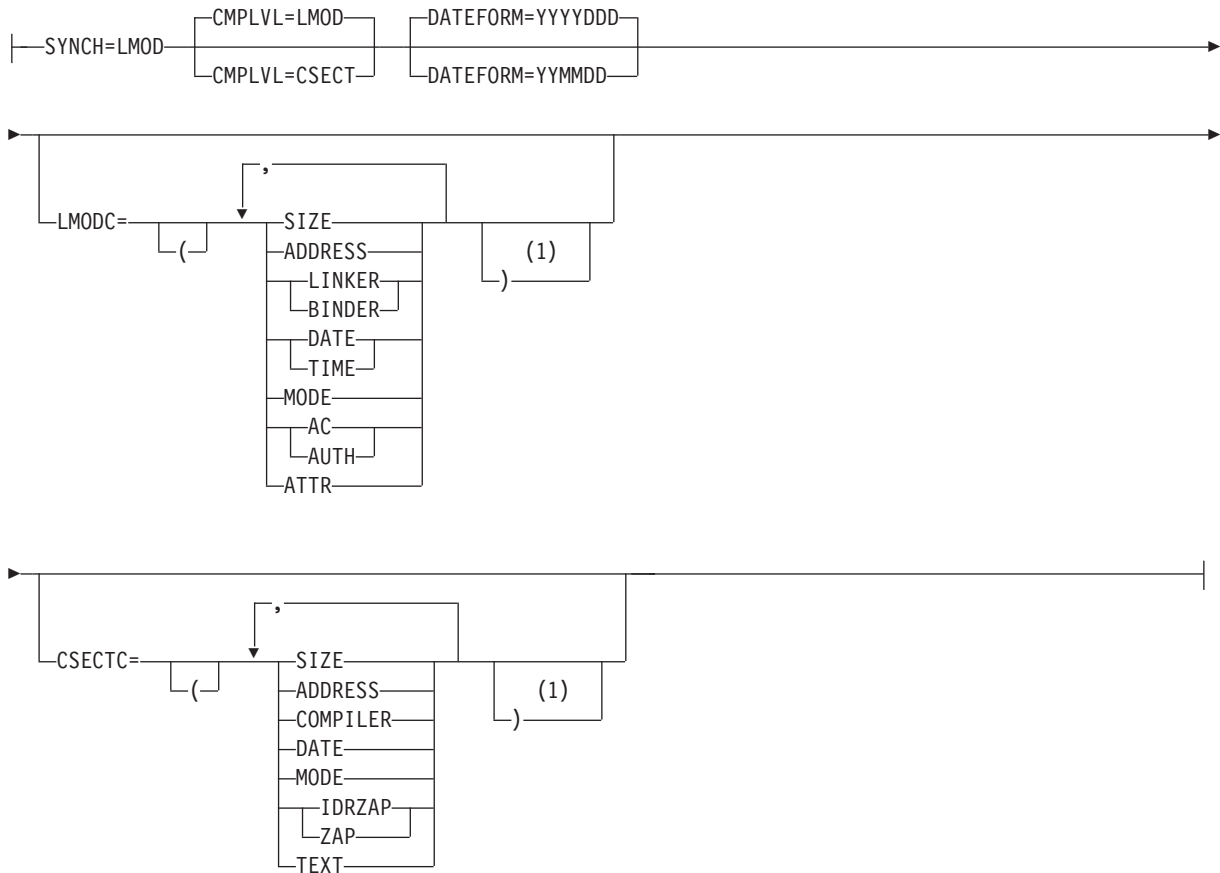


Notes:

- 1 All of the following keywords are shown as optional in the syntax, but in practice the actual requirements are dependent on a number of factors, as described in the SYNCH=KEYED definition below the syntax diagram.

DSM Syntax: Part 7 of 7

Load module compare options (from Part 4):



Notes:

- 1 Provide closing bracket when opening bracket has been used.

Old data set specifications (Part 1 of syntax diagram)

The "Old" data set can be specified as follows:

DDOLD=ddold

Defines a reference to a DD or TSO ALLOC statement for the "Old" data set or HFS file. The default is DDOLD.

DSNOLD=dsnold

Defines the name of the "Old" data set or an absolute path to the "Old" HFS file. If specified, any DD statements provided are not used. The name may include a member name in parenthesis. If the member is specified here, the associated MEMOLD parameter must be empty. An absolute path to an HFS file must be enclosed in apostrophes. If it does not fit on one line, you can split it over more than one line. To further describe the data set, use the following:

VOLSEROLD=*volserold*

The VOLUME serial number for a non-cataloged "Old" data set.

MEMOLD=*memold*

The name of a single member in a PDS, or a member name pattern representing one or more members in a PDS library. You can specify this parameter, or a member name in the DD statement for *ddname*, or specify a member or members in the MEMLIST parameter, or specify a range of member names with the MEMSTART and MEMEND keywords.

A member name pattern can consist of any characters that are valid in a member name, and two special pattern characters: the asterisk (*) and the percent symbol (%).

An * represents any number of characters. As many asterisks as required can appear anywhere in a member name pattern. For example, if you enter a member name pattern of *d*, all members in the PDS whose name contains "d" are processed.

A % is a place-holding character that represents a single character. As many percent symbols as necessary can appear anywhere in a member name pattern. For example, if you enter a member name pattern of %%%%, all members in the PDS with a 4-character name are processed.

MEMOLD is ignored if the data set is not a PDS.

MEMSTART=*startstring*

Is used to specify the start of a range of member names to be included in the compare. If MEMSTART is specified but MEMEND is omitted, all members of the PDS(E) from the *startstring* value onwards are included. *startstring* can have the same values, including wild cards, as for the *memold* parameter of the MEMOLD keyword.

MEMEND=*endstring*

Is used to specify the end of a range of member names to be included in the compare. If MEMEND is specified but MEMSTART is omitted, all members of the PDS(E) up to the *endstring* value onwards are included. *endstring* can have the same values, including wild cards, as for the *memold* parameter of the MEMOLD keyword.

CREATED=*created*

The date on which a member was created, in YYYY/MM/DD format.

If the "Old" data set is a PDS(E), you can specify this parameter, or specify a range of creation dates with the CRESTART and CREEND keywords.

You can specify an asterisk (*) as the last character to indicate a range of dates, or a percent sign (%) in place of a single character to indicate a selection of dates.

created is ignored if the data set is not a PDS.

CRESTART=*crestart*

The start of a range of creation dates in YYYY/MM/DD format to be included in the compare.

If CRESTART is specified but CREEND is omitted, all members of the PDS(E) from the *crestart* value onwards are included.

If omitted, or you do not enter a full date, or you specify an asterisk (*) as the last character, the unspecified portion of *crestart* defaults to the right as follows:

DD = 01
MM = 01
YYYY = 0000

No other wildcarding is allowed.

CREEND=*creend*

The end of a range of creation dates in YYYY/MM/DD format to be included in the compare.

If omitted, or you do not enter a full date, or you specify an asterisk (*) as the last character, the unspecified portion of *creend* defaults to the right as follows:

DD = 31
MM = 12
YYYY = 9999

No other wildcarding is allowed.

CHANGED=*changed*

The date on which a member was last modified, in YYYY/MM/DD format.

If the “Old” data set is a PDS(E), you can specify this parameter, or specify a range of modification dates with the CHGSTART and CHGENG keywords.

You can specify an asterisk (*) as the last character to indicate a range of dates, or a percent sign (%) in place of a single character to indicate a selection of dates.

changed is ignored if the data set is not a PDS.

CHGSTART=*chgstart*

The start of a range of modification dates in YYYY/MM/DD format to be included in the compare.

If CHGSTART is specified but CHGENG is omitted, all members of the PDS(E) from the *chgstart* value onwards are included.

If omitted, or you do not enter a full date, or you specify an asterisk (*) as the last character, the unspecified portion of *chgstart* defaults to the right as follows:

DD = 01
MM = 01
YYYY = 0000

No other wildcarding is allowed.

CHGENG=*chengend*

The end of a range of modification dates in YYYY/MM/DD format to be included in the compare.

If omitted, or you do not enter a full date, or you specify an asterisk (*) as the last character, the unspecified portion of *chengend* defaults to the right as follows:

Function reference: DSM

DD = 31
MM = 12
YYYY = 9999

No other wildcarding is allowed.

USERID=*userid*

The TSO user ID by which the member was last updated.

If the “Old” data set is a PDS(E), you can specify this parameter, or specify a range of user IDs with the UIDSTART and UIDEND keywords.

You can enter a generic user ID by using asterisks and percent signs.

userid is ignored if the data set is not a PDS.

UIDSTART=*uidstart*

The start of a range of user IDs to be included in the compare.

If UIDSTART is specified but UIDEND is omitted, all members of the PDS(E) from the *uidstart* value onwards are included.

If omitted, or you do not enter a full 7-character user ID, or you specify an asterisk (*) as the last character, File Manager replaces the asterisk and pads the unspecified portion of *uidstart* to the right with low values (X'00').

UIDEND=*uidend*

The end of a range of user IDs to be included in the compare.

If you omit this field, it defaults to high values (X'FF').

If you specify less than 7 characters (without an asterisk as the last character), File Manager pads *uidstart* to the right with low values (X'00'). If you specify an asterisk (*) as the last character, File Manager replaces the asterisk and pads the unspecified portion of *uidend* with high values (X'FF').

MEMLIST

Provides a means of selecting members from a PDS where no generic name pattern and no member name range has been specified. If the MEMLIST keyword is specified, only those members included in the MEMLIST arguments are compared with the corresponding members in the output data set. Members selected by the MEMBER=*memold* that are not included in the MEMLIST arguments are not compared.

The MEMLIST arguments also refine the member list specified in the MEMOLD=*memold* parameter. If the MEMOLD keyword is *not* specified, it is assumed to be MEMOLD=*, and all members named in the MEMLIST list are processed. However, if the MEMLIST keyword is specified, for example as MEMLIST=TEST*, the members included in the MEMLIST list are selected from the TEST* subset of members. Any members named in the MEMLIST arguments that do not match the mask given in the MEMOLD parameter are not compared.

comp_mem

The name of the member to be compared. Generic name masks are not allowed.

USEIOXOLD

Specifies whether to invoke a user I/O exit, to process the “Old” data set.

NO Default. Do not invoke a user I/O exit.

YES Invoke a user I/O exit to process the “Old” data set. This option is only available if the person who did the site customization for File Manager allowed user I/O exits on a site-wide basis.

IOXOLD

Specifies the name of the user I/O exit used for the “Old” data set. There are no restrictions on the programming language that you can use to write an exit. The exit must be provided to File Manager in the STEPLIB/ISPLLIB concatenation or their extensions (LINKLIST, LPA, and so on).

sysexit Default. If you specify USEIOXOLD=YES and do not supply a user I/O exit name, File Manager uses the name of the exit provided in the installation customization options. If USEIOXOLD has been set to YES and no installation default has been provided, you must specify IOXOLD=*ioxname*.

Note: If you have selected batch processing in an online panel, the generated JCL statements use the default name provided in your Set System Processing Options panel.

ioxname

The name of a PDS(E) member of a data set that has been provided to File Manager in the STEPLIB concatenation.

BINRECOLD=binrecold-len

Specifies the record length used for processing the “Old” HFS file. Valid range: 1 to 32760.

The file is processed in Binary mode (fixed-length records derived from the file, delimiters not distinguished). If you do not specify this parameter, the file is processed in Text mode (variable-length records, boundaries determined by delimiters).

KEYOLD=keyold

A key for KSDS records or a slot number for RRDS records, for the “Old” data set. The maximum key length is 30 characters. The first record with a key or slot value greater than or equal to *key* is the first record compared. If you omit the *keyold* and *skipold* values, the comparison begins with the first record in the data set.

If the key contains lowercase characters, blanks, or commas, enclose it in quotation marks. You can also specify a key in hexadecimal format (for example, X'C1C2C3').

SKIPOLD=skipold

Number of logical records to be skipped from the beginning of the “Old” data set. The default is 0.

COMPOLD=cmpold

Number of records from the “Old” data set to be compared. The

Function reference: DSM

maximum number is 99 999 999. If you specify ALL or omit the parameter, all the remaining records are compared.

IGNLEN

Specifies whether or not File Manager ignores length mismatches when selecting records for processing.

NO Do not ignore length mismatches. Records that are shorter than the matching structure length in the template are not selected for processing.

YES Use this option to ignore length mismatches.

Old template processing (Part 1 of syntax diagram)

Use these options to specify the "Old" copybook or template that describes the record structure of your "Old" data set.

TOLD=*ddname*

Defines a reference to a DD or TSO ALLOC statement for the data sets which contain the copybook or template that describes the record structure of your "Old" data set. The default is TDOLD.

If you specify a concatenated DD, then you must provide the member name, *member*.

TOLDMEM=*member*

The name of the copybook or template member in the datasets identified by the TOLD parameter if it has not been specified on the DD statement. This parameter must not be specified if the TCOLD parameter is specified.

TCOLD=*tcold(tcomem)*

PDS and member name of the "Old" copybook or template that describes the record structure of your "Old" data set.

OFFSETOLD

The length of the 01 field in the "Old" template and the start locations of the fields within that 01 field are adjusted by the value provided.

value The offset value, which must be in the range -32760 to 32760, to be applied to the corresponding field identifier. If no field identifier is supplied and ALL is not used, the value is applied to the first Level 01 field in the "Old" template.

ALL Where the template contains multiple record structures, this keyword applies the corresponding *value* to all Level 01 fields within the "Old" template.

Note: You can specify a value for ALL and then override this value for individual layouts by providing subsequent *value* and *fieldname* combinations.

fieldname

The name of the Level 01 field to which *value* is to be applied. The default is the first Level 01 field in the "Old" template.

FIELDOLD=(*oldfield1,oldfield2,...*)

Field name or names in the "Old" template, used to create a mapping for a formatted comparison. This keyword is used in

conjunction with the FIELDNEW keyword to define mapping within the batch file. The parentheses are optional when only one field is specified but mandatory when more than one field is included. The names are mapped in the order given, that is, *oldfield1* is mapped to *newfield1* and so on, overriding any default or existing mapping.

FIELDOLD and FIELDNEW can be used with or without an existing mapping in the template and with or without the CORRESP keyword setting, as follows:

Table 13. Batch mapping behavior

Specifications			Behavior
“New” template contains mapping?	CORRESP=?	FIELDxxx specified?	
no	NO	no	map corresponding fields
		yes	map specified fields
	YES	no	map corresponding fields
		yes	map corresponding fields then remap specified fields
yes	NO	no	use mapping in TCNEW template
		yes	use mapping in TCNEW template then remap specified fields
	YES	no	map corresponding fields
		yes	map corresponding fields then remap specified fields

Note: “Specified fields” refers to those fields specified in the FIELDOLD and FIELDNEW arguments.

Handling multiple 01s and duplicate field names

Field mapping specifications that are created via the FIELDOLD and FIELDNEW keywords are applied on a first match basis. For example, if the “Old” template came from a copybook containing:

```

01 OLD-TYPE01.
03 BINARY-X                PIC 999999999 USAGE BINARY.
01 OLD-TYPE02.
03 BINARY-1                PIC 999999999 USAGE BINARY.
01 OLD-TYPE03.
03 BINARY-1                PIC 999999999 USAGE BINARY.
    
```

and the “New” template came from a copybook containing:

```

01 NEW-TYPE01.
03 BINARY-2                PIC 999999999 USAGE BINARY.
01 NEW-TYPE02.
03 BINARY-2                PIC 999999999 USAGE BINARY.
05 DUP-FIELD.
07 BINARY-2                PIC 999999999 USAGE BINARY.
01 NEW-TYPE03.
03 BINARY-2                PIC 999999999 USAGE BINARY.
    
```

a mapping specification of:

```

$$FILEM FIELDOLD=BINARY-1,
$$FILEM FIELDNEW=BINARY-2,
    
```

Function reference: DSM

maps the 03 BINARY-2 field in NEW-TYPE02 to the BINARY-1 field in OLD-TYPE02. This is because the 03 BINARY-2 field in NEW-TYPE02 is the first “New” template field found named BINARY-2, where the corresponding 01-level, OLD-TYPE02, contains a field called BINARY-1. The 03 BINARY-2 field in NEW-TYPE01 is not mapped because the corresponding 01-level, OLD-TYPE01, does not contain a 03 BINARY-1 field.

You can override this default behavior to specify different field mappings by using a dot qualification. For example,

```
$$FILEM FIELDOLD=BINARY-1,  
$$FILEM FIELDNEW=DUPFIELD.BINARY-2,
```

would map the 07 BINARY-2 field in NEW-TYPE02 to the BINARY-1 field in OLD-TYPE02.

```
$$FILEM FIELDOLD=BINARY-1,  
$$FILEM FIELDNEW=NEW-TYPE03.BINARY-2,
```

would map the 03 BINARY-2 field in NEW-TYPE03 to the BINARY-1 field in OLD-TYPE03.

Notes:

1. This only affects the mapping. Normal record identification procedures must be understood and employed to ensure the correct 01 is in effect for each record comparison.
2. The qualifiers are resolved from left to right, skipping over levels not present in the qualification so that only enough information to uniquely identify a field need be provided.

New data set specifications (Part 2 of syntax diagram)

The “New” data set can be specified as follows:

DDNEW=*ddnew*

Defines a reference to a DD or TSO ALLOC statement for the “New” data set or HFS file. The default is DDNEW.

DSNNEW=*dsnnew*

Defines the name of the “New” data set or an absolute path to the “New” HFS file. If specified, any DD statement provided are not used. The name may include a member name in parenthesis. If the member is specified here, the associated MEMNEW parameter must be empty. An absolute path to an HFS file (directory) must be enclosed in apostrophes. If it does not fit on one line, you can split it over more than one line. To further describe the data set, use the following:

VOLSERNEW=*volsernew*

The VOLUME serial number for a non-cataloged “New” data set.

MEMNEW=*memnew*

The name of a single member in a PDS library, or a member name pattern representing one or more members in the library. You can specify this parameter, or a member name in the DD statement for *ddname*.

A member name pattern can consist of any characters that are valid in a member name, and two special pattern characters: the asterisk (*) and the percent symbol (%).

An * represents any number of characters. As many asterisks as required can appear anywhere in a member name pattern. For example, if you enter a member name pattern of *d*, all members in the library whose name contains "d" are processed.

A % is a place holding character that means a single character. As many percent symbols as necessary can appear anywhere in a member name pattern. For example, if you enter a member name pattern of %%%%, all members in the library with a 4-character name are processed.

Specification of MEMNEW (or a member in DSNNEW) depends on the parameters used in MEMOLD (or member used in DSNOLD). If MEMOLD (member in DSNOLD) specifies one member, MEMNEW (member in DSNNEW) must also point at one member. If MEMOLD (member in DSNOLD) contains a member name pattern, the specification of MEMNEW (member in DSNNEW) must use the same pattern or an "*".

MEMNEW is ignored if the data set is not a PDS.

USEIOXNEW

Specifies whether to invoke a user I/O exit, to process the "New" data set.

NO Default. Do not invoke a user I/O exit.

YES Invoke a user I/O exit to process the "New" data set. This option is only available if the person who did the site customization for File Manager allowed user I/O exits on a site-wide basis.

IOXNEW

Specifies the name of the user I/O exit used for the "New" data set. There are no restrictions on the programming language that you can use to write an exit. The exit must be provided to File Manager in the STEPLIB/ISPLLIB concatenation or their extensions (LINKLIST, LPA, and so on).

sysexit Default. If you specify USEIOXNEW=YES and do not supply a user I/O exit name, File Manager uses the name of the exit provided in the installation customization options. If USEIOXNEW has been set to YES and no installation default has been provided, you must specify IOXNEW=*ioxname*.

Note: If you have selected batch processing in an online panel, the generated JCL statements use the default name provided in your Set System Processing Options panel.

ioxname

The name of a PDS(E) member of a data set that has been provided to File Manager in the STEPLIB concatenation.

BINRECNEW=*binrecnew-len*

Specifies the record length used for processing the "New" HFS file. Valid range: 1 to 32760.

The file is processed in Binary mode (fixed-length records derived from the file, delimiters not distinguished). If you do not specify

this parameter, the file is processed in Text mode (variable-length records, boundaries determined by delimiters).

KEYNEW=*keynew*

A key for KSDS records or a slot number for RRDS records, for the "New" data set. The maximum key length is 30 characters. The first record with a key or slot value greater than or equal to *key* is the first record compared. If you omit the *keynew* and *skipnew* values, the comparison begins with the first record in the data set.

If the key contains lowercase characters, blanks, or commas, enclose it in quotation marks. You can also specify a key in hexadecimal format (for example, X'C1C2C3').

SKIPNEW=*skipnew*

Number of logical records to be skipped from the beginning of the "New" data set. The default is 0.

CMPNEW=*cmpnew*

Number of records from the "New" data set to be compared. The maximum number is 99 999 999. If you specify ALL or omit the parameter, all the remaining records are compared.

New template processing (Part 2 of syntax diagram)

Use these options to specify the "New" copybook or template that describes the record structure of your "New" data set.

TNEW=*ddname*

Defines a reference to a DD or TSO ALLOC statement for the data sets which contain the copybook or template member that describes the record structure of your "New" data set. The default is TDNEW.

If you specify a concatenated DD, then you must provide the member name, *member*.

TNEWMEM=*member*

The name of the copybook or template member in the datasets identified by the TNEW parameter if it has not been specified on the DD statement. This parameter must not be specified if the TCNEW parameter is specified.

TCNEW=*tcnew(tcnmem)*

PDS and member name of the "New" copybook or template that describes the record structure of your "New" data set.

OFFSETNEW

The length of the 01 field in the "New" template and the start locations of the fields within that 01 field are adjusted by the value provided.

value The offset value, which must be in the range -32760 to 32760, to be applied to the corresponding field identifier. If no field identifier is supplied and ALL is not used, the value is applied to the first Level 01 field in the "New" template.

ALL Where the template contains multiple record structures, this keyword applies the corresponding *value* to all Level 01 fields within the "New" template.

Note: You can specify a value for ALL and then override this value for individual layouts by providing subsequent *value* and *fieldname* combinations.

fieldname

The name of the Level 01 field to which *value* is to be applied. The default is the first Level 01 field in the “New” template.

FIELDNEW=(*newfield1,newfield2,...*)

Field name or names in the “New” template, used to create a mapping for a formatted comparison. This keyword is used in conjunction with the FIELDOLD keyword to define mapping within the batch file. The parentheses are optional when only one field is specified but mandatory when more than one field is included. The names are mapped in the order given, that is, *oldfield1* is mapped to *newfield1* and so on, overriding any default or existing mapping.

FIELDOLD and FIELDNEW can be used with or without an existing mapping in the template and with or without the CORRESP keyword setting. See Table 13 on page 925 for details.

Copybook processing (Part 3 of syntax diagram)

If you specify a copybook (instead of an existing template) for either TCOLD or TCNEW, then File Manager uses these processing options to compile the copybook into a template:

LANG

Determines whether File Manager automatically detects the copybook language or interprets the language as COBOL,PL/I, or HLASM.

AUTO

Automatically detect whether the copybook language is COBOL or PL/I, and invoke the appropriate compiler. If the compilation results in a return code greater than 4, then invoke the compiler for the other language. If the second compilation also results in a return code greater than 4, then retry the first compiler and report the compilation errors. If File Manager successfully creates a template (despite the compilation errors), then continue processing with the template.

COBOL

Invoke the COBOL compiler to create a template from the copybook. (Do not invoke the PL/I compiler, even if the COBOL compilation results in errors.)

PLI

Invoke the PL/I compiler to create a template from the copybook. (Do not invoke the COBOL compiler, even if the PL/I compilation results in errors.)

HLASM

Invoke the HLASM compiler to create a template from the copybook.

COBOL options (Part 3 of syntax diagram)

The following options are used to compile a COBOL copybook into a template:

Function reference: DSM

DBCS=YES

Use the DBCS compiler option.

DBCS=NO

Use the NODBCS compiler option.

For details on the effect of the DBCS and NODBCS compiler options, see the *IBM COBOL Programming Guide for OS/390 & VM*.

CDPC=NO

Do not use the COBOL SPECIAL-NAMES paragraph "Decimal-point is comma".

CDPC = YES

Use the COBOL SPECIAL-NAMES paragraph "Decimal-point is comma".

CAE=NO

Do not use the COBOL compile option ARITH(EXTEND).

CAE = YES

Use the COBOL compile option ARITH(EXTEND).

MIXED = NO

Field names stored in the template in uppercase.

MIXED = YES

Field names stored in the template in the original case as coded in the COBOL copybook.

RFROM1=RTO1 ... RFROM5=RTO5

Up to five pairs of "From" and "To" pseudo-text character strings for the COBOL REPLACE compiler-directing statement.

If your COBOL copybooks contain characters that you want to remove or replace with other characters before compiling the copybooks into templates, then use these replacing options.

For example, if your copybooks contain colon characters (:) that you want to remove before compiling, then specify '==:==' as *operand1* and '====' as *operand2*.

For details on specifying "From" and "To" strings for COBOL REPLACE, see the *IBM COBOL Language Reference*.

COMPMAXRC

Sets the maximum acceptable return code for a copybook compile. A return code higher than the specified level causes the function to stop. Default is 4.

PL/I options (Part 3 of syntax diagram)

The following options are used to compile a PL/I copybook into a template:

BIN63=YES Use the LIMITS(FIXEDBIN(63)) compiler option.

BIN63=NO Use the LIMITS(FIXEDBIN(31)) compiler option.

DEC31=YES Use the LIMITS(FIXEDDEC(31)) compiler option.

DEC31=NO Use the LIMITS(FIXEDDEC(15)) compiler option.

GRAPHIC=YES

Use the GRAPHIC compiler option.

GRAPHIC=NO

Use the NOGRAPHIC compiler option.

UNALIGNED=YES

Use the DEFAULT RANGE (*) UNALIGNED, language statement to change the default alignment.

UNALIGNED=NO

Use the PL/I default.

COMPMAXRC

Sets the maximum acceptable return code for a copybook compile. A return code higher than the specified level causes the function to stop. Default is 4.

For details on the effect of these compiler options, see the *IBM VisualAge PL/I for OS/390 Programming Guide*.

HLASM options

The following options are used to compile a HLASM copybook into a template:

DBCS=YES Use the DBCS compiler option.

DBCS=NO Use the NODBCS compiler option.

NOALIGN=YES

Use the NOALIGN compiler option.

NOALIGN=NO

Use the ALIGN compiler option.

COMPMAXRC

Sets the maximum acceptable return code for a copybook compile. A return code higher than the specified level causes the function to stop. Default is 4.

For details on the effect of these compiler options, see the *HLASM V1R5 Programmer's Guide*.

Note: You cannot specify different options for compiling "Old" and "New" copybooks; the same copybook options are used for both.

Comparison options (Part 5 of syntax diagram)**TYPE=RECORD**

Record comparison.

CORRESP=NO

Use this option if the field mapping you want to use is in the TCNEW member. If the TCNEW member is a copybook, or no field mapping is supplied, then File Manager ignores this option and performs a compare as if you had specified CORRESP=YES.

CORRESP=YES

This option instructs File Manager to map output fields to input fields with the corresponding name.

If you want to use the existing mapping in the "New" template, specify CORRESP=NO.

SYNCH=ONETOONE

One-to-one synchronization.

SYNCH=READAHEAD

Read-ahead synchronization. If specified, you can use the following option:

LIMIT=ralim

Limit for read-ahead synchronization.

LENGTH=ralen

The number of records that must match during read-ahead processing for synchronization to occur.

LIST=SUMMARY

Summary listing.

LIST=DELTA

Delta listing.

LIST=MATCHING

Matching listing.

LIST=LONG

Long listing.

LIST=NONE

No listing.

Listing Options

The following option takes effect if the LIST=LONG parameter is specified:

EXCLUDE=exclude_type

The specified compare result types are not reported.

exclude_type can have the following values:

INSERTED

Excludes inserted records from the report.

DELETED

Excludes deleted records from the report.

CHANGED

Excludes changed records from the report.

MATCHED

Excludes matched records from the report.

The following options take effect if the LIST keyword is not specified (that is, it defaults to SUMMARY) or is set to anything other than NONE:

NUMDIFF=numdiff

The number of differences after which the Compare Utility stops processing the data sets.

RCDIFF=(numdiff,return_code)

Sets the batch return code when a threshold of changes has been met, where:

numdiff

The minimum number of differences to trigger the return code.

return_code

The batch return code value that is set if the number of differences have been detected.

|
|
|
|
|
|
|
|
|
|
|

WIDE=NO

Narrow listing.

WIDE=YES

Wide listing. The WIDE listing is limited in width to approximately 32K bytes of (record) data. When working with records longer than 32K, the record data, and optional change highlighting, is truncated past 32K bytes of data because of the SYSPRINT output record limitation. The entire record length is used to perform the comparison so records are marked correctly as "changed" regardless of length.

HEX=NO

No hex formatting.

HEX=YES

Show hex formatting.

Note: The print processing option, DUMP, is ignored and the hexadecimal print output is in updown format.

HIGHLIGHT=NO

No highlighting of changed fields.

HIGHLIGHT=YES

Highlight changed fields.

SHOWATTR=NO

Suppress attribute information in headings (affects formatted comparisons only).

SHOWATTR=YES

Show attribute information in headings (affects formatted comparisons only).

CHNGDFLD=NO

Show all fields in the formatted comparison reports.

CHNGDFLD=YES

Show only changed fields in formatted comparison reports. This option has no effect if the 'Wide listing' (**WIDE=YES**) option has been selected, or for record type comparisons.

SLCTDFLD=NO

Fields selected in the template are not shown in addition to changed fields. This option has effect only if **CHNGDFLD=YES** is also specified.

SLCTDFLD=YES

Fields selected in the template are always shown. This option has effect only if **CHNGDFLD=YES** is also specified.

PACK=UNPACK

Allow detection of ISPF packed data and unpack the records if they are packed before passing to the processing routine or printing.

PACK=NONE

Omit the detection of ISPF packed data and process the records as they are.

Function reference: DSM

PACK=SKIP

Check for ISPF packed data and if packed, skip processing of this data set or member.

IOOUTPUT=ddname

Defines a reference to a DD or TSO ALLOC statement for the data set or HFS file the "inserted" records are to be written to.

IOUTMEM=member

The name of the member in the dataset identified by the COUTPUT parameter if it has not been specified on the DD statement.

IOUTDSN=ioutdsn(member)

The name of the output data set, or an absolute path to the output HFS file, the "inserted" records are to be written to. The name can include a member name in parenthesis.

An absolute path to an HFS file (directory) must be enclosed in apostrophes. If it does not fit on one line, you can split it over more than one line.

IBINREC=binrec_len

Specifies the record length to be used for processing the inserted HFS output data set. Valid range is 1–32760. The file is processed in binary mode. If you do not specify this parameter, the file is processed in text mode.

IDISP=MOD

Writes inserted records to the existing output data set, starting from the beginning.

IDISP=OLD

Appends inserted records to the existing output data set.

DOOUTPUT=ddname

Defines a reference to a DD or TSO ALLOC statement for the data set or HFS file the "deleted" records are to be written to.

DOUTMEM=member

The name of the member in the dataset identified by the COUTPUT parameter if it has not been specified on the DD statement.

DOUTDSN=doutdsn(member)

The name of the output data set, or an absolute path to the output HFS file, the "deleted" records are to be written to. The name can include a member name in parenthesis.

An absolute path to an HFS file (directory) must be enclosed in apostrophes. If it does not fit on one line, you can split it over more than one line.

DBINREC=binrec_len

Specifies the record length to be used for processing the deleted HFS output data set. Valid range is 1–32760. The file is processed in binary mode. If you do not specify this parameter, the file is processed in text mode.

DDISP=MOD

Writes deleted records to the existing output data set, starting from the beginning.

DDISP=OLD

Appends deleted records to the existing output data set.

NCOUTPUT=ddname

Defines a reference to a DD or TSO ALLOC statement for the data set or HFS file the new "changed" records are to be written to. The default is FMNCHOUT.

NCOUTMEM=member

The name of the member in the dataset identified by the NCOUTPUT parameter if it has not been specified on the DD statement.

NCOUTDSN=ncoutdsn(member)

The name of the output data set, or an absolute path to the output HFS file, the new "changed" records are to be written to. The name can include a member name in parenthesis.

An absolute path to an HFS file (directory) must be enclosed in apostrophes. If it does not fit on one line, you can split it over more than one line.

NCBINREC=binrec_len

Specifies the record length to be used for processing the New Change HFS output data set. Valid range is 1–32760. The file is processed in binary mode. If you do not specify this parameter, the file is processed in text mode.

NCDISP=MOD

Writes new changed records to the existing output data set, starting from the beginning.

NCDISP=OLD

Appends new changed records to the existing output data set.

OCOUTPUT=ddname

Defines a reference to a DD or TSO ALLOC statement for the data set or HFS file the new "changed" records are to be written to. The default is FMOCHOUT.

OCOUTMEM=member

The name of the member in the dataset identified by the OCOUTPUT parameter if it has not been specified on the DD statement.

OCOUTDSN=ocoutdsn(member)

The name of the output data set, or an absolute path to the output HFS file, the old "changed" records are to be written to. The name can include a member name in parenthesis.

An absolute path to an HFS file (directory) must be enclosed in apostrophes. If it does not fit on one line, you can split it over more than one line.

OCBINREC=binrec_len

Specifies the record length to be used for processing the New Change HFS output data set. Valid range is 1–32760. The file is processed in binary mode. If you do not specify this parameter, the file is processed in text mode.

OCDISP=MOD

Writes old changed records to the existing output data set, starting from the beginning.

OCDISP=OLD

Appends old changed records to the existing output data set.

NMOUTPUT=ddname

Defines a reference to a DD or TSO ALLOC statement for the data set or HFS file the new "matched" records are to be written to. The default is FMNMTOUT.

NMOUTMEM=member

The name of the member in the dataset identified by the COUTPUT parameter if it has not been specified on the DD statement.

NMOUTDSN=nmoutdsn(member)

The name of the output data set, or an absolute path to the matched HFS file, the new "matched" records are to be written to. The name can include a member name in parenthesis.

An absolute path to an HFS file (directory) must be enclosed in apostrophes. If it does not fit on one line, you can split it over more than one line.

NMBINREC=binrec_len

Specifies the record length to be used for processing the new matched HFS output data set. Valid range is 1–32760. The file is processed in binary mode. If you do not specify this parameter, the file is processed in text mode.

NMDISP=MOD

Writes new matched records to the existing output data set, starting from the beginning.

NMDISP=OLD

Appends new matched records to the existing output data set.

OMOUTPUT=ddname

Defines a reference to a DD or TSO ALLOC statement for the data set or HFS file the old "matched" records are to be written to. The default is FMOMTOUT.

OMOUTMEM=member

The name of the member in the dataset identified by the COUTPUT parameter if it has not been specified on the DD statement.

OMOUTDSN=omoutdsn(member)

The name of the output data set, or an absolute path to the matched HFS file, the old "matched" records are to be written to. The name can include a member name in parenthesis.

An absolute path to an HFS file (directory) must be enclosed in apostrophes. If it does not fit on one line, you can split it over more than one line.

OMBINREC=binrec_len

Specifies the record length to be used for processing the old

matched HFS output data set. Valid range is 1–32760. The file is processed in binary mode. If you do not specify this parameter, the file is processed in text mode.

OMDISP=MOD

Writes old matched records to the existing output data set, starting from the beginning.

OMDISP=OLD

Appends old matched records to the existing output data set.

Formatted type options (Part 6 of syntax diagram)

If you specify TYPE=FORMATTED, for a formatted comparison, you can use the following options:

IGNORELB=NO

Respect leading blanks when comparing alphanumeric fields.

IGNORELB=YES

Ignore leading blanks when comparing alphanumeric fields.

IGNORETB=NO

Respect trailing blanks when comparing alphanumeric fields.

IGNORETB=YES

Ignore trailing blanks when comparing alphanumeric fields.

MATCHREF=NO

Ignore leading blanks (unless IGNORELB=NO is also specified), trailing blanks (unless IGNORETB=YES also specified) and embedded blanks when comparing alphanumeric fields.

MATCHREF=YES

Respect leading blanks, trailing blanks and embedded blanks when comparing alphanumeric fields.

Note: Any setting of IGNORELB or IGNORETB is ignored if MATCHREF=YES is also specified.

IGNORECASE=NO

Respect case when comparing alphanumeric fields.

IGNORECASE=YES

Ignore case when comparing alphanumeric fields.

Keyed synchronization options (Part 6 of syntax diagram)

If you specify SYNCH=KEYED, you are requesting keyed synchronization.

If specified, you can define up to sixteen key segments to be concatenated to form a single key. A key segment is comprised of the key's location in the "Old" and "New" data sets and the key's length and data type. These values are built using one of the following sources or a combination of these elements:

- The intrinsic data set keys (where the data set is keyed, for example, VSAM KSDS).

If only one of the data sets is keyed, the location and length values for the first segment is initialized from the available data set key information, and the keywords corresponding to the initialized values are not required. However, you need to supply the location value and, optionally, the data type for the non-keyed data set. If you do not specify the data type, the default type of AN is used.

Function reference: DSM

If both data sets are keyed, you do not need to supply any of the keywords and the location values is initialized from the data set key. The length value is that of the shorter of the two keys.

If used in a multi-segment key, this defines the first segment. To prevent other segment sources from overriding the intrinsic key, leave the first argument in the **KEYxxx** keyfields as a void, for example, **KEYLOCOLD=(,keylocold)**.

- The key segment sequence information stored in your templates.
- The **KEYFLDOLD** and **KEYFLDNEW** keyfields (when a template has been specified)
- The **KEYLOCOLD**, **KEYLOCNEW**, **KEYLEN**, and (optionally) **KEYTYPE** keyfields.

In most situations, you would use only one of these sources to define your key segments, however, all of them can be intermixed to define the key segments, with the caveat that **KEYLOCOLD**, **KEYLOCNEW**, **KEYLEN**, and **KEYTYPE** override **KEYFLDOLD** and **KEYFLDNEW** when both specify data for the same segment.

Up to 16 values can be specified for each keyword, and the parentheses can be omitted if only one value is provided. The same number of key segments must be defined for both data sets, and the data type and length of corresponding segments must match. Each set of corresponding values then defines a key segment.

Notes:

1. **KEYFLDOLD** and **KEYFLDNEW** can only be specified if a template is specified for the corresponding data set.
2. Any segment whose presence is implied by a keyword must be fully specified, except that the data type for the segment can default as described in **KEYTYPE** below.

KEYTCOLD

Determines whether or not any key segment information stored in the “Old” template is used.

If unspecified, the default behavior is that key segment information stored in the “Old” template is only used when all of the following conditions are met:

- **TCOLD=tcold** has been specified.
- The “Old” template contains key segment information.
- No other keyed synchronization options have been specified for either the “Old” or the “New” data sets.

If any of these conditions are not met, key segment information in an “Old” template is ignored.

If specified, the setting given overrides the default behavior.

YES The key segment information in the “Old” template is loaded regardless of whether or not other **KEY...** keywords have been specified. Where present, the other **KEY...** keywords function as overrides to the template specifications.

NO The key segment information in the “Old” template is ignored regardless of whether or not other **KEY...** keywords have been specified.

KEYTCNEW

Determines whether or not any key segment information stored in the “New” template is used.

If unspecified, the default behavior is that key information stored in the “New” template is only used when all of the following conditions are met:

- TCNEW=*tcnew* has been specified.
- The “New” template contains key segment information.
- No other keyed synchronization options have been specified for either the “Old” or the “New” data sets.

If any of these conditions are not met, key segment information in an “New” template is ignored.

If specified, the setting given overrides the default behavior.

YES The key segment information in the “New” template is loaded regardless of whether or not other **KEY...** keywords have been specified. Where present, the other **KEY...** keywords function as overrides to the template specifications.

NO The key segment information in the “New” template is ignored regardless of whether or not other **KEY...** keywords have been specified.

KEYFLDOLD=(*keyfieldold*,...)

Specifies the name of the field or fields in the “Old” template to be used as key segments. When used in combination with other key segment sources, empty arguments must be included to indicate the position of the *keyfieldold* value or values in the concatenated key.

KEYFLDNEW=(*keyfieldnew*,...)

Specifies the name of the field or fields in the “New” template to be used as key segments. When used in combination with other key segment sources, empty arguments must be included to indicate the position of the *keyfieldnew* value or values in the concatenated key.

KEYLOCOLD=(*keylocold*,...)

Key locations in “Old” data set for keyed synchronization. When used in combination with other key segment sources, empty arguments must be included to indicate the position of the *keylocold* value or values in the concatenated key.

KEYLOCNEW=(*keylocnew*)

Key location in “New” data set for keyed synchronization. When used in combination with other key segment sources, empty arguments must be included to indicate the position of the *keylocnew* value or values in the concatenated key.

KEYLEN=*keylen*

Key length for keyed synchronization. When used in combination with other key segment sources, empty arguments must be included to indicate the position of the *keylen* value in the concatenated key.

KEYTYPE

Defines the data type of the key segment. Can be used to override

the data type of an existing template field. When used in combination with other key segment sources, empty arguments must be included to indicate the position of the *keylen* value in the concatenated key.

KEYTYPE is optional - any key segments defined by **KEYLOCxxx/KEYLEN** that do not have a corresponding **KEYTYPE** keyword is given type **CHAR**.

Key segments defined using **KEYFLDOLD** and **KEYFLDNEW** inherit the type of the template field on which they are based, and the expected record sequence is assumed to follow accordingly. In practice this is only likely to be significant for signed binary and packed decimal fields (though internal floating point is also supported). For example, if a key were defined on a two-byte field by location and length, then a value of '001C'x would be less than a value of '001D'x. However if the key were defined via a packed-decimal template field then '001D'x (-1) would be less than '001C'x (+1). Equivalent results can be obtained by using **KEYTYPE** with **KEYLOCxxx** and **KEYLEN**.

CHAR

This is equivalent to the internal C/AN data type. **CHAR** data type segments are synchronized using the normal EBCDIC collating sequence.

BINARY

This is equivalent to the internal B/BI data type. **BINARY** data type segments are synchronized as signed binary integers. They must have a length of 2, 4 or 8.

PACKED

This is equivalent to the internal P/PD data type. **PACKED** data type segments are synchronized as signed packed decimal integers. They must have a length less than or equal to 16.

FLOAT

This is equivalent to the internal FP data type. **FLOAT** data type segments are synchronized as signed floating point numbers. They must have a length less of 4 or 8.

Note: Template fields that are selected as key segments but do not have one of the above data types, is treated as **CHAR**. In particular, this means that the actual data lengths of varying fields is ignored.

Load module compare options (Part 7 of syntax diagram)

If you specify **SYNCH=LMOD**, you are requesting load module comparison.

Load module comparison has these specific options:

CMPLVL

Determines the level of load module comparison.

LMOD

Only information on the load module level is extracted and compared. **CSECT** information (and differences at **CSECT** level) is ignored. This results in a less detailed comparison.

CSECT

Information on both the load module and CSECT levels is extracted and compared. This results in a detailed comparison.

DATEFORM=YYYYDDD

Reported dates (link and compile dates) shown in YYYY.DDD format.

DATEFORM=YYMMDD

Reported dates (link and compile dates) shown in YY/MM/DD format.

LMODC

Determines what information at the load module level is to be included in the compare. The criteria correspond with load module properties; only those specified are compared and displayed. Each of the options below can be specified in any sequence, enclosed in parenthesis:

SIZE The load module size is compared.

ADDRESS

The entry point address of load module is compared.

LINKER | BINDER

The version of the linkage editor or binder used to prepare the load module is compared. LINKER and BINDER are mutually exclusive.

DATE | TIME

The load module link (bind) date and time are compared. DATE and TIME are mutually exclusive.

MODE

The AMODE and RMODE of the load module are compared.

AC | AUTH

The load module authorization code is compared. AC and AUTH are mutually exclusive.

ATTR The load module link (bind) attributes are compared.

CSECTC

Determines what information at the CSECT level is to be included in the compare. The criteria correspond with the CSECT properties; only those specified are compared and displayed. Each of the options below can be specified in any sequence, enclosed in parenthesis. If CMPLVL=LMOD, this parameter is ignored.

SIZE The CSECT size is compared.

ADDRESS

The address of the CSECT is compared.

COMPILER

The versions of the language compilers used to compile the CSECT are compared

DATE The date of the CSECT compile is compared.

MODE

The AMODE and RMODE of the CSECT are compared.

IDRZAP | ZAP

The AMSPZAP IDR data is compared. The IDR ZAP data is an extension of the CSECT information, but is formatted into separate records. ISRZAP and ZAP are mutually exclusive.

TEXT The CSECT content is compared. The CSECT content is an extension of the CSECT information, but is formatted into separate, 32-byte records shown in "memory dump" format (hexadecimal and character).

Template reporting options (Part 6 of syntax diagram)

PBK=NONE

Template layout and criteria information not reported.

PBK=LAYOUTS

Template layout information reported if TYPE=FORMATTED is also specified.

PBK=CRITERIA

Template criteria information reported. In this case, a terse layout report is also generated, containing only the fields referred to in the criteria expressions.

PBK=ALL

Template layout and criteria information reported. If TYPE=RECORD is specified, a terse layout report is generated, containing only the fields referred to in the criteria expressions. If TYPE=FORMATTED is specified, a full layout report is generated.

MAP=NONE

Template mapping information not reported.

MAP=MAPPED

Mapped template fields reported if TYPE=FORMATTED is also specified.

MAP=UNMAPPED

Unmapped template fields reported if TYPE=FORMATTED is also specified.

MAP=ALL

Both mapped and unmapped template fields reported if TYPE=FORMATTED is also specified.

ARRAY=YES

If field information is reported, all occurrences of any array elements are reported.

For COBOL OCCURS DEPENDING ON tables, the maximum occurrences are reported.

For PL/1 REFER arrays, the minimum occurrences are reported.

ARRAY=NO

Occurrences of array elements not reported.

HEXLOC=YES

If field information is reported, all field start and end positions are reported as hexadecimal offsets.

HEXLOC=NO

Field start and end positions not reported as hexadecimal offsets.

HEXLEN=YES

If field information is reported, all field lengths are reported in hexadecimal.

HEXLEN=NO

Field lengths not reported in hexadecimal.

Simple batch example: One-to-one synchronization

```
//DSM JOB (ACCT), 'NAME'
/* COMPARE DATA SETS
/*
//FMBAT PROC
//FMBAT EXEC PGM=FILEMGR
//STEPLIB DD DSN=FMN.SFMNMOD1,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSABEND DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
// PEND
/*
//STPSSEX EXEC FMBAT
//SYSIN DD *
$$FILEM VER
$$FILEM DSM TYPE=RECORD,
$$FILEM SYNCH=ONETOONE,
$$FILEM LIST=LONG,
$$FILEM WIDE=YES,
$$FILEM HILIGHT=YES,
$$FILEM DSNOLD=FMN.SFMNSAM1(FMNCDATA),
$$FILEM SKIPOLD=0,
$$FILEM CMPOLD=20,
$$FILEM DSNNEW=FMN.SFMNSAM1(FMNCDATA),
$$FILEM SKIPNEW=20,
$$FILEM CMPNEW=20
$$FILEM EOJ
/*
```

Complex batch example

```
$$FILEM DSCMP TYPE=RECORD,
$$FILEM SYNCH=KEYED,
$$FILEM LIST=LONG,
$$FILEM WIDE=YES,
$$FILEM HILIGHT=YES,
$$FILEM HEX=YES,
$$FILEM DSNOLD=USERID.COMPARE.KSDS1,1
$$FILEM KEYLOCOLD=(,11),2
$$FILEM DSNNEW=USERID.COMPARE.FLAT2,3
$$FILEM TCNEW=USERID.TEMPLATE(FLAT2),4
$$FILEM KEYTCNEW=YES,5
$$FILEM KEYFLDNEW=(,FIELD-3),6
$$FILEM KEYLEN=(,1),7
$$FILEM KEYTYPE=(,CHAR)8
$$FILEM EOJ
```

Notes:

1. DSNOLD is intrinsically keyed with key position 1 and length 10.
2. KEYLOCOLD provides a position for the second segment of the “Old” key.
3. DSNNEW is not intrinsically keyed.
4. TCNEW specifies a new template: the template specifies FIELD-1 (position 1, length 10, type AN) as key segment 1.

Function reference: DSM

5. KEYTCNEW forces the loading of the TCNEW key segment information, in spite of the presence of the other KEY... keywords, which act as overrides wherever they clash with the information from the template.
6. KEYFLDNEW specifies FIELD-3 (position 12, length 2, type BI) as the second segment of the “New” key
7. KEYLEN overrides the length of the second key segment
8. KEYTYPE overrides the type of the second key segment

The resulting key segments are shown in the following table:

Segment number	“Old” key position	“New” key position	Key length	Key type
1	1 ¹	1 ²	10 ³	AN ⁴
2	11 ⁵	12 ⁶	1 ⁷	AN ⁸

Source of key information:

1. DSNOLD catalog entry
2. Template field FIELD-1
3. DSNOLD catalog entry and template field FIELD-1 (must be consistent)
4. Template field FIELD-1 (consistent with AN default for key type)
5. KEYLOCOLD
6. Template field FIELD-3 via KEYFLDNEW
7. KEYLEN
8. KEYTYPE

DSP (Data Set Print)

Purpose

Use the DSP function to print sequential data sets, VSAM data sets, PDS members, or HFS files in a selected format. You can print data by:

- record
- block (non-VSAM)
- control interval (VSAM)
- field (if a template or copybook has been provided)

You can select records for printing using:

- Member name selection criteria
- Date created selection criteria
- Date last modified selection criteria
- User ID selection criteria
- the start key (VSAM only)
- skip and print count fields
- conditional expression defined in the provided template

The print function can be run in the foreground or as a batch job. The output for the print function is controlled by the SET options.

Usage notes

Choose between four data print formats:

- character
- hexadecimal
- single-record, using a template
- multiple-record (tabular), using a template

When you use a copybook or template, records are formatted field by field using the record layout defined in the copybook or template.

You can Print concatenated data sets with like or unlike attributes. Note that, under some conditions (with tape data sets), File Manager may not be able to detect unlike data set attributes and still invoke DFSORT for processing. Such invocation may fail as DFSORT does not allow for unlike concatenation of data sets. In such cases, you can disable DFSORT with the NOSORT function to allow for successful processing of concatenated datasets with unlike attributes.

Performance tips

- See “General tips about performance when you use File Manager functions” on page 809. The comments about File Manager using DFSORT technology when performing sequential file I/O are important to DSP performance.

Options

You can specify the following options:

- Whether to process logical records or physical blocks.
- The position of the first record to print.
- The number of records to print.
- The name of a DFSORT or REXX procedure, if you want the output records passed to a procedure for processing before they are printed.

Physical block processing is not compatible with SNGL or TABL print format, or with using templates, and SNGL or TABL print format requires you to use a template.

You can use the various SET processing options to control the print output:

- SET PRINTOUT defines the destination of the print output. If set to PRINTOUT=SYSOUT, you can use the PB (Print Browse) function to browse the accumulated output.
- When you specify CHAR or LHEX print format, SET RECLIMIT controls how many bytes of each record are printed, and SET DATAHDR determines whether header information, such as record number, is printed.
- The format of the print output also depends on the settings of SET PAGESIZE, SET PRINTLEN, and SET PRTRTRANS.
- Use SET DUMP to specify the dump format.

Note that additional formatting options are available when using formatted print in SNGL mode. Additional field information (redefined fields, field reference number, field type and length values, picture clause, start location, structure and numeric field justification) can be set and printed depending on the options selected using the keywords listed later in this section.

When you specify the PROC option, you are supplying a REXX procedure. For more information, see the *proc* parameter below.

Return codes

The default return codes from the DSP function have the following modified meanings:

- | | |
|---|--|
| 1 | No records printed for some of multiple members |
| 2 | Print error encountered |
| 3 | REXX member selection is in effect but the procedure encountered |

Function reference: DSP

a RETURN DROP, STOP or STOP IMMEDIATE string. This has been treated as a RETURN string with no arguments. OR REXX member selection is NOT in effect but the procedure encountered a RETURN DROP MEMBER or RETURN PROCESS MEMBER string. This has been treated as a RETURN string with no arguments.

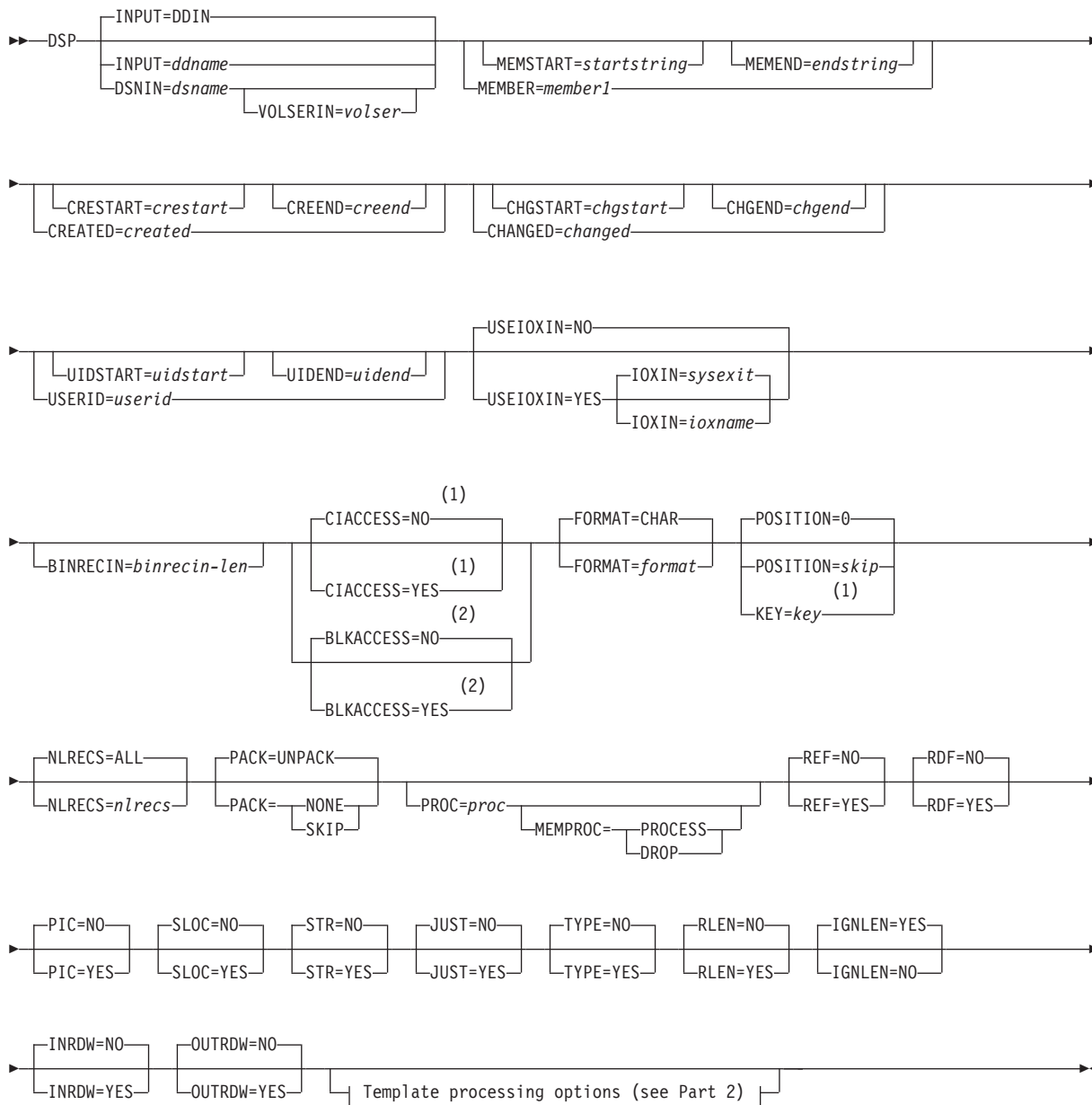
- 4 No records printed because no records selected
- 4 No records printed because no members to process
- 4 No records printed because input empty
- 4 No records printed because the input data set or member is in ISPF Packed Data format and the "PACK=SKIP" option was specified
- 8 REXX non-syntax error encountered while processing records
- 16 Data set or member open error
- 16 Data set not found
- 16 Input data appears ISPF packed but is not valid.
- 16 Other input or output error occurred
- 16 Insufficient storage available
- 16 DSP abended
- 16 Other serious error that stops processing occurred

Note: Return codes can be customized during installation. If you receive return codes that do not match those listed above, your site might have customized the return codes in place for this function. File Manager may also issue the 999 abend, if the return code in batch is equal to or greater than the ABENDCC value. Please contact your File Manager systems administrator for details.

Related functions

- DP Print physical disk records
- DVT Print VTOC entries

Syntax: Part 1 of 3



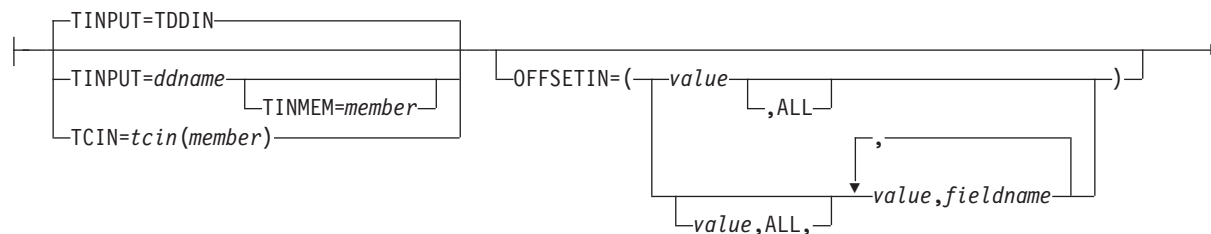
Notes:

- 1 VSAM only.
- 2 Non-VSAM only.

Function reference: DSP

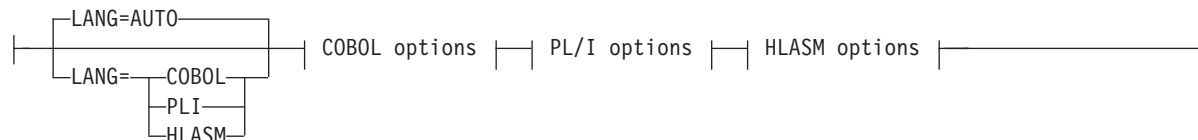
Syntax: Part 2 of 3

Template processing options (from Part 1):

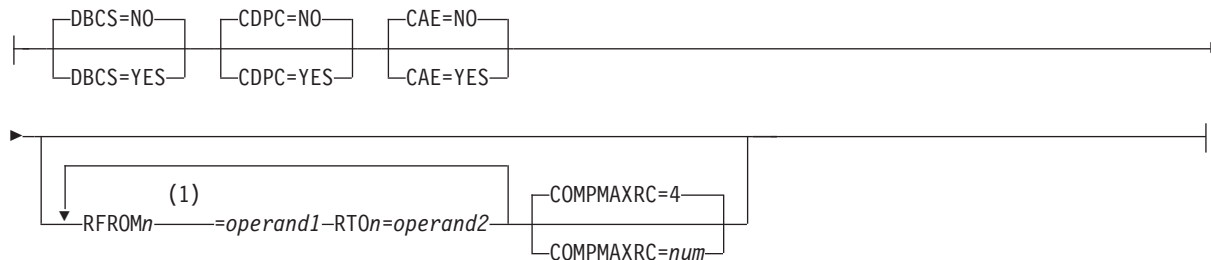


Copybook processing options

Copybook processing options:



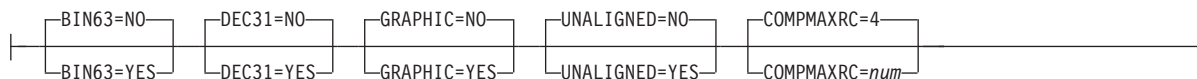
COBOL options:



Notes:

1 RFROM1 RTO1, RFROM2 RTO2, ... RFROM5 RTO5 ($n=1-5$).

Syntax: Part 3 of 3

PL/I options:**HLASM options:****INPUT=ddname**

Defines a reference to a DD or TSO ALLOC statement for the input data set or HFS file. The default is DDIN.

DSNIN=dsname

Defines the name of the input data set or an absolute path to the input HFS file (directory). If specified, any DD statement provided are not used. The name may include a member name in parenthesis. If the member is specified here, the associated Member parameter must be empty. An absolute path to an HFS file (directory) must be enclosed in apostrophes. If it does not fit on one line, you can split it over more than one line. You can further describe this data set, as follows:

VOLSERIN=volser

Volume serial number for a non-cataloged data set.

MEMBER=member1

The name of a single member in a PDS, or a member name pattern representing one or more members in a PDS. If the input data set is a PDS(E), you may specify this parameter, or a member name in the DD statement for *ddname*, or specify a range of member names with the MEMSTART and MEMEND keywords.

A member name pattern can consist of any characters that are valid in a member name and two special pattern characters: the asterisk (*) and the percent symbol (%).

- * represents any number of characters. As many asterisks as required can appear anywhere in a member name pattern. For example, if you enter a member name pattern of *d*, all members in the PDS whose name contains "d" are processed.
- % is a place holding character that means a single character. As many percent symbols as necessary can appear anywhere in a member name pattern. For example, if you enter a member name pattern of %%%, all members in the PDS whose name is four characters in length are processed.

member1 is ignored if the data set is not a PDS.

MEMSTART=startstring

Is used to specify the start of a range of member names to be included in the copy. If MEMSTART is specified but MEMEND is omitted, all members

Function reference: DSP

of the PDS(E) from the *startstring* value onwards are included. *startstring* can have the same values, including wild cards, as for the *member1* parameter of the MEMBER keyword.

MEMEND=*endstring*

Is used to specify the end of a range of member names to be included in the copy. If MEMEND is specified but MEMSTART is omitted, all members of the PDS(E) up to the *endstring* value onwards are included. *endstring* can have the same values, including wild cards, as for the *member1* parameter of the MEMBER keyword.

CREATED=*created*

The date on which a member was created, in YYYY/MM/DD format.

If the input data set is a PDS(E), you may specify this parameter, or specify a range of creation dates with the CRESTART and CREEND keywords.

You can specify an asterisk (*) as the last character to indicate a range of dates or a percent sign (%) in place of a single character to indicate a selection of dates.

created is ignored if the data set is not a PDS.

CRESTART=*crestart*

The start of a range of creation dates in YYYY/MM/DD format to be included in the copy.

If CRESTART is specified but CREEND is omitted, all members of the PDS(E) from the *crestart* value onwards are included.

If omitted, or you do not enter a full date, or you specify an asterisk (*) as the last character, the unspecified portion of *crestart* defaults to the right as follows:

DD = 01
MM = 01
YYYY = 0000

No other wildcarding is allowed.

CREEND=*creend*

The end of a range of creation dates in YYYY/MM/DD format to be included in the copy.

If omitted, or you do not enter a full date, or you specify an asterisk (*) as the last character, the unspecified portion of *creend* defaults to the right as follows:

DD = 31
MM = 12
YYYY = 9999

No other wildcarding is allowed.

CHANGED=*changed*

The date on which a member was last modified, in YYYY/MM/DD format.

If the input data set is a PDS(E), you may specify this parameter, or specify a range of modification dates with the CHGSTART and CHGEND keywords.

You can specify an asterisk (*) as the last character to indicate a range of dates or a percent sign (%) in place of a single character to indicate a selection of dates.

changed is ignored if the data set is not a PDS.

CHGSTART=*chgstart*

The start of a range of modification dates in YYYY/MM/DD format to be included in the copy.

If CHGSTART is specified but CHGEND is omitted, all members of the PDS(E) from the *chgstart* value onwards are included.

If omitted, or you do not enter a full date, or you specify an asterisk (*) as the last character, the unspecified portion of *chgstart* defaults to the right as follows:

DD = 01
MM = 01
YYYY = 0000

No other wildcarding is allowed.

CHGEND=*chgend*

The end of a range of modification dates in YYYY/MM/DD format to be included in the copy.

If omitted, or you do not enter a full date, or you specify an asterisk (*) as the last character, the unspecified portion of *chgend* defaults to the right as follows:

DD = 31
MM = 12
YYYY = 9999

No other wildcarding is allowed.

USERID=*userid*

The TSO user ID by which the member was last updated.

If the input data set is a PDS(E), you may specify this parameter, or specify a range of user IDs with the UIDSTART and UIDEND keywords.

You can enter a generic user ID by using asterisks and percent signs.

userid is ignored if the data set is not a PDS.

UIDSTART=*uidstart*

The start of a range of user IDs to be included in the copy.

If UIDSTART is specified but UIDEND is omitted, all members of the PDS(E) from the *uidstart* value onwards are included.

If omitted, or you do not enter a full 7-character user ID, or you specify an asterisk (*) as the last character, File Manager replaces the asterisk and pads the unspecified portion of *uidstart* to the right with low values (X'00').

UIDEND=*uidend*

The end of a range of user IDs to be included in the copy.

If you omit this field, it defaults to high values (X'FF').

If you specify less than 7 characters (without an asterisk as the last character), File Manager pads *uidstart* to the right with low values (X'00').

Function reference: DSP

If you specify an asterisk (*) as the last character, File Manager replaces the asterisk and pads the unspecified portion of *uidend* with high values (X'FF').

USEIOXIN

Specifies whether to invoke a user I/O exit, to process the input data set.

NO Default. Do not invoke a user I/O exit.

YES Invoke a user I/O exit to process the input data set. This option is only available if the person who did the site customization for File Manager allowed user I/O exits on a site-wide basis.

IOXIN

Specifies the name of the user I/O exit used for the input data set. There are no restrictions on the programming language that you can use to write an exit. The exit must be provided to File Manager in the STEPLIB/ISPLLIB concatenation or their extensions (LINKLIST, LPA, and so on).

sysexit Default. If you specify USEIOXIN=YES and do not supply a user I/O exit name, File Manager uses the name of the exit provided in the installation customization options. If USEIOXIN has been set to YES and no installation default has been provided, you must specify IOXIN=*ioxname*.

Note: If you have selected batch processing in an online panel, the generated JCL statements use the default name provided in your Set System Processing Options panel.

ioxname

The name of a PDS(E) member of a data set that has been provided to File Manager in the STEPLIB concatenation.

BINRECIN=*binrecin-len*

Specifies the record length used for processing the HFS file. Valid range: 1 to 32760.

The file is processed in Binary mode (fixed-length records derived from the file, delimiters not distinguished). If you do not specify this parameter, the file is processed in Text mode (variable-length records, boundaries determined by delimiters).

CIACCESS=NO

Process logical records.

CIACCESS=YES

Process control intervals.

BLKACCESS=NO

Process logical records.

BLKACCESS=YES

Process blocks.

FORMAT=*format*

The format of the output:

CHAR

Character format (the default).

HEX Hexadecimal format.

SNGL Single-record format (one field on each line). This option is available when viewing logical records.

TABL Tabular format (fields printed across the page). This option is available when viewing logical records.

POSITION=skip

Number of logical records to be skipped from the beginning of the data set. The default is 0.

KEY=key (VSAM only)

A key for KSDS records, or a slot number for RRDS records. The maximum key length is 30 characters. The first record with a key or slot value greater than or equal to *key* is the first record printed. If you omit the *key* and *skip* values, printing begins with the first record in the data set.

If the key contains lowercase characters, blanks, or commas, enclose it in quotation marks. You can also specify a key in hexadecimal format (for example, X'C1C2C3').

NLRECS

Number of records to be printed or ALL.

ALL If you specify ALL or omit the parameter, all the remaining records are copied.

nlrecs The maximum number is 99 999 999.

PACK Determines if File Manager should detect if the input data is in ISPF packed format. This keyword is ignored when processing VSAM data sets. When an I/O exit has been specified for either the input or output data set (or both), the only valid option is PACK=NONE.

UNPACK

Instructs File Manager to check if the input data set is in ISPF packed format and if so, to unpack it before print processing.

NONE

Instructs File Manager not to check if the input data set is in ISPF packed format.

SKIP Instructs File Manager to determine if the input data set is in ISPF packed format and if so, to skip the print processing.

PROC=proc

Member name of a REXX procedure that you want to use to process each record before it is printed, or an asterisk (*) to indicate the REXX procedure is inline. If you specify a member name, you must define an FMNEXEC ddname that identifies the PDS containing the member. If you specify *, the procedure is read from SYSIN immediately following the control statement for the current function. The inline procedure is terminated by a record containing a slash and a plus sign (/+) in columns 1–2.

For more information about using REXX procedures to process records before they are printed, see Chapter 13, "Enhancing File Manager processing," on page 385.

MEMPROC

Specifies that REXX member selection is in effect. Records are read from the input member and then cached in memory until a decision is made, within the REXX procedure, on whether the member is to be copied or dropped. Once the decision has been made, the entire member is either copied or dropped, depending upon the RETURN string specified in the REXX procedure. If the entire member is processed without encountering a

Function reference: DSP

RETURN DROP MEMBER or RETURN PROCESS MEMBER string, the member is processed according to the action specified by the parameter specified for MEMPROC. These are:

PROCESS

The member is to be included in the copy. The member is copied intact, subject to any specified template processing, which is performed before the user REXX proc is invoked. This is the default action, if no parameter is specified with the MEMPROC keyword.

DROP The member is to be excluded from the copy. Processing continues with the next member.

REF=YES

Show field reference number on SNGL print.

REF=NO

Do not show field reference number on SNGL print.

RDF=YES

Show redefined fields on SNGL or TABL print.

RDF=NO

Do not show redefined fields on SNGL or TABL print.

PIC=YES

Show picture clause on SNGL print.

PIC=NO

Do not show picture clause on SNGL print.

SLOC=YES

Show start location on SNGL print.

SLOC=NO

Do not show start location on SNGL print.

STR=YES

Show structure on SNGL print.

STR=NO

Do not show structure on SNGL print.

JUST=YES

Left-justify numeric fields on SNGL print.

JUST=NO

Do not left-justify numeric fields on SNGL print.

TYPE=YES

Show field type and length values on SNGL print.

TYPE=NO

Do not show field type and length values on SNGL print.

RLEN=YES

Print record length in TABL and SNGL formats.

RLEN=NO

Do not print record length in TABL and SNGL formats.

IGNLEN

Specifies whether or not File Manager ignores length mismatches when selecting records for processing.

NO Do not ignore length mismatches. Records that are shorter than the matching structure length in the template are not selected for processing.

YES Use this option to ignore length mismatches.

INRDW

Controls whether or not to adjust the input start location when the specified start location takes into account the record descriptor word (RDW).

NO Does not adjust the input start location.

YES Subtracts 4 from all start locations that have been coded on external functions that refer to the input record.

OUTRDW

Controls whether or not to adjust the output start location when the specified start location takes into account the record descriptor word (RDW).

NO Does not adjust the output start location.

YES Subtracts 4 from all start locations that have been coded on external functions that refer to the output record.

Template processing

Define which template (if any) is used to describe the record structure in the input data set, and how File Manager processes this template.

TINPUT=*ddname*

Defines a reference to a DD or TSO ALLOC statement for the data sets which contain the copybook or template that describes the record structure of your input data. The default is TDDIN.

If you specify a concatenated DD, then you must provide the member name, *member*.

TINMEM=*member*

The name of the copybook or template member in the datasets identified by the TINPUT parameter if it has not been specified on the DD statement. This parameter must not be specified if the TCIN parameter is specified.

TCIN=*tcin(member)*

PDS and member name of the copybook or template that describes the record structure of your input data.

OFFSETIN

The length of the 01 field in the template and the start locations of the fields within that 01 field are adjusted by the value provided.

value The offset value, which must be in the range -32760 to 32760, to be applied to the corresponding field identifier. If no field identifier is supplied and ALL is not used, the value is applied to the first Level 01 field in the template.

ALL Where the template contains multiple record structures, this keyword applies the corresponding *value* to all Level 01 fields within the template.

Note: You can specify a value for ALL and then override this value for individual layouts by providing subsequent *value* and *fieldname* combinations.

fieldname

The name of the Level 01 field to which *value* is to be applied. The default is the first Level 01 field in the template.

Copybook processing

If you specify a copybook (instead of an existing template), then File Manager uses these processing options to compile the copybook into a template:

LANG

Determines whether File Manager automatically detects the copybook language or interprets the language as COBOL, PL/I, or HLASM.

AUTO

Automatically detect whether the copybook language is COBOL or PL/I, and invoke the appropriate compiler. If the compilation results in a return code greater than 4, then invoke the compiler for the other language. If the second compilation also results in a return code greater than 4, then retry the first compiler and report the compilation errors. If File Manager successfully creates a template (despite the compilation errors), then continue processing with the template.

COBOL

Invoke the COBOL compiler to create a template from the copybook. (Do not invoke the PL/I compiler, even if the COBOL compilation results in errors.)

PLI Invoke the PL/I compiler to create a template from the copybook. (Do not invoke the COBOL compiler, even if the PL/I compilation results in errors.)

HLASM

Invoke the HLASM compiler to create a template from the copybook.

COBOL options

The following options are used to compile a COBOL copybook into a template:

DBCS=YES

Use the DBCS compiler option.

DBCS=NO

Use the NODBCS compiler option.

For details on the effect of the DBCS and NODBCS compiler options, see the *IBM COBOL Programming Guide for OS/390 & VM*.

CDPC=NO

Do not use the COBOL SPECIAL-NAMES paragraph "Decimal-point is comma".

CDPC = YES

Use the COBOL SPECIAL-NAMES paragraph "Decimal-point is comma".

CAE=NO

Do not use the COBOL compile option ARITH(EXTEND).

CAE = YES

Use the COBOL compile option ARITH(EXTEND).

MIXED = NO

Field names stored in the template in uppercase.

MIXED = YES

Field names stored in the template in the original case as coded in the COBOL copybook.

RFROM1 RTO1 ... RFROM5 RTO5

Up to five pairs of "From" and "To" pseudo-text character strings for the COBOL REPLACE compiler-directing statement.

If your COBOL copybooks contain characters that you want to remove or replace with other characters before compiling the copybooks into templates, then use these replacing options.

For example, if your copybooks contain colon characters (:) that you want to remove before compiling, then specify '==:==' as *operand1* and '====' as *operand2*.

For details on specifying "From" and "To" strings for COBOL REPLACE, see the *IBM COBOL Language Reference*.

COMPMAXRC

Sets the maximum acceptable return code for a copybook compile. A return code higher than the specified level causes the function to stop. Default is 4.

PL/I options

The following options are used to compile a PL/I copybook into a template:

BIN63=YES Use the LIMITS(FIXEDBIN(63)) compiler option.

BIN63=NO Use the LIMITS(FIXEDBIN(31)) compiler option.

DEC31=YES Use the LIMITS(FIXEDDEC(31)) compiler option.

DEC31=NO Use the LIMITS(FIXEDDEC(15)) compiler option.

GRAPHIC=YES

Use the GRAPHIC compiler option.

GRAPHIC=NO

Use the NOGRAPHIC compiler option.

UNALIGNED=YES

Use the DEFAULT RANGE (*) UNALIGNED, language statement to change the default alignment.

UNALIGNED=NO

Use the PL/I default.

COMPMAXRC

Sets the maximum acceptable return code for a copybook compile. A return code higher than the specified level causes the function to stop. Default is 4.

For details on the effect of these compiler options, see the *IBM VisualAge PL/I for OS/390 Programming Guide*.

HLASM options

The following options are used to compile a HLASM copybook into a template:

DBCS=YES Use the DBCS compiler option.

DBCS=NO Use the NODBCS compiler option.

NOALIGN=YES
Use the NOALIGN compiler option.

NOALIGN=NO
Use the ALIGN compiler option.

COMPMAXRC

Sets the maximum acceptable return code for a copybook compile. A return code higher than the specified level causes the function to stop. Default is 4.

For details on the effect of these compiler options, see the *HLASM V1R5 Programmer's Guide*.

PACK Determines if File Manager should detect if the input data sets are in ISPF packed format. This keyword is ignored when processing VSAM data sets. When an I/O exit has been specified for either data set (or both), the only valid option is PACK=NONE.

UNPACK

Instructs File Manager to check if the input data sets are in ISPF packed format and if they are, to unpack them before the comparison.

NONE

Instructs File Manager not to check if the input data sets are in ISPF packed format.

SKIP Instructs File Manager to check if the input data set is in ISPF packed format and if so, to skip the compare processing.

Batch example 1

```
//DSP JOB (acct),'name' Print QSAM Data
//*
//FMBAT PROC
//FMBAT EXEC PGM=FILEMGR
//STEPLIB DD DSN=FMN.SFMNMOD1,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSABEND DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
// PEND
//*
//STPSPEX EXEC FMBAT
//SYSIN DD *
$$FILEM VER
$$FILEM DSP DSNIN=SYS1.PROCLIB,MEMBER=COBUCLG
$$FILEM EOJ
/*
```

Batch example 2

```
//DSPJPN JOB (acct),'name' Print with DBCS characters
//JAPEF96 OUTPUT DUPLEX=NORMAL,CHARS=(GT15,EF96),PRMODE=SOSI1
//FILEMGR EXEC PGM=FILEMGR
//SYSPRINT DD SYSOUT=*,OUTPUT=(*.JAPEF96)
//FMNTSPRT DD SYSOUT=*
//SYSTEM DD SYSOUT=*
//SYSIN DD *
$$FILEM SET LANGUAGE=JAPANESE
$$FILEM DSP FORMAT=SNGL,
$$FILEM TCIN=h1q.TEMPLATE(member),
$$FILEM DSNIN=h1q.DBCSDATA
```

Batch example 3

```
//FMNUSR3 JOB (FMNUSER),'USER',USER=FMNUSER,NOTIFY=FMNUSER,
// TIME=(5),CLASS=A,MSGLEVEL=(1,1),MSGCLASS=H
//*
/* TEST PRINTING USING JAPANESE CHARACTER SETS WITH FMT
/*
//JAPEF96 OUTPUT DUPLEX=NORMAL,CHARS=(GT15,EF96),PRMODE=SOSI1
//FILEMGR EXEC PGM=FMNMAIN
//STEPLIB DD DSN=FMN.V2RIM0.SFMNMOD1,DISP=SHR
//SYSPRINT DD SYSOUT=H,OUTPUT=(*.JAPEF96),DEST=(PTHMVS8,QAPT22Q1)
//SYSTEM DD SYSOUT=*
//SYSIN DD *
$$FILEM SET LANGUAGE=JAPANESE
$$FILEM FMT SET,FLD=(11,20,DB),FLD=(31,40,DB),FLD=(51,60,DB),
$$FILEM FLD=(71,80,DB)
$$FILEM DSP DSNIN=FMNUSER.JPN.TESTDATA,MEMBER=$FMTDATA
/*
```

DSU (Data Set Update) — batch only**Purpose**

Update disk data set records.

Function reference: DSU

Usage notes

Use this function to update logical records in a single sequential disk data set, a single VSAM data set, or one or more members of a PDS.

Note: DFSMS-compressed datasets are not supported (for use with DSU).

You can select the records to be processed using:

- Member name selection criteria
- Date created selection criteria
- Date last modified selection criteria
- User ID selection criteria

Records in the data set are read sequentially. After each record is read, File Manager invokes the REXX procedure specified in the PROC parameter, and passes the contents of the record to the exec. The contents are passed in two File Manager-defined REXX variables, INREC and OUTREC. When the exec is invoked, the contents of the two variables are identical. The INREC variable is intended to be used as a reference variable. Any changes made to it are ignored by File Manager. The OUTREC variable can be updated by the exec. After the REXX procedure has processed the record, if the data in OUTREC has changed, the record is updated in the data set using the contents of OUTREC.

You cannot add records or delete records using DSU. If you need to add or delete records, you can use one of the File Manager data set copy functions. You cannot change the length of records in a data set using DSU. If the REXX procedure increases the length of the data in OUTREC, the data is truncated to its original length before the record is updated. If the REXX procedure decreases the length of the data in OUTREC, the data is padded to its original length using the pad value specified in the PAD processing option. If no pad value has been specified, the contents of the record are unpredictable.

Performance tips

- When you use DSU to update members of a PDS(E):
One DSU default is STATS=ON, which causes the ISPF statistics for each updated member to be updated. This can significantly increase I/O (EXCP) and CPU utilization. To improve performance, consider using STATS=OFF.

Options

When you specify the PROC option, you are supplying a REXX procedure. For more information, see the *proc* parameter below.

Return codes

The default return codes from the DSU function have the following modified meanings:

- | | |
|---|--|
| 1 | One or more members not updated |
| 2 | Change failed (for example invalid key change) |
| 4 | No records updated |
| 4 | No records processed because no members to process |
| 4 | No records processed because input empty |
| 4 | No records processed because input is in ISPF Packed Data format and the "PACK=STOP" option was specified. |

- 8 REXX non-syntax error encountered while processing records
- 16 Program Object specified - this is not supported
- 16 Data set or member in use
- 16 Data set or member open error
- 16 Data set not found
- 16 Other input or output error occurred
- 16 Insufficient storage available
- 16 DSU abended
- 16 Other serious error that stops processing occurred

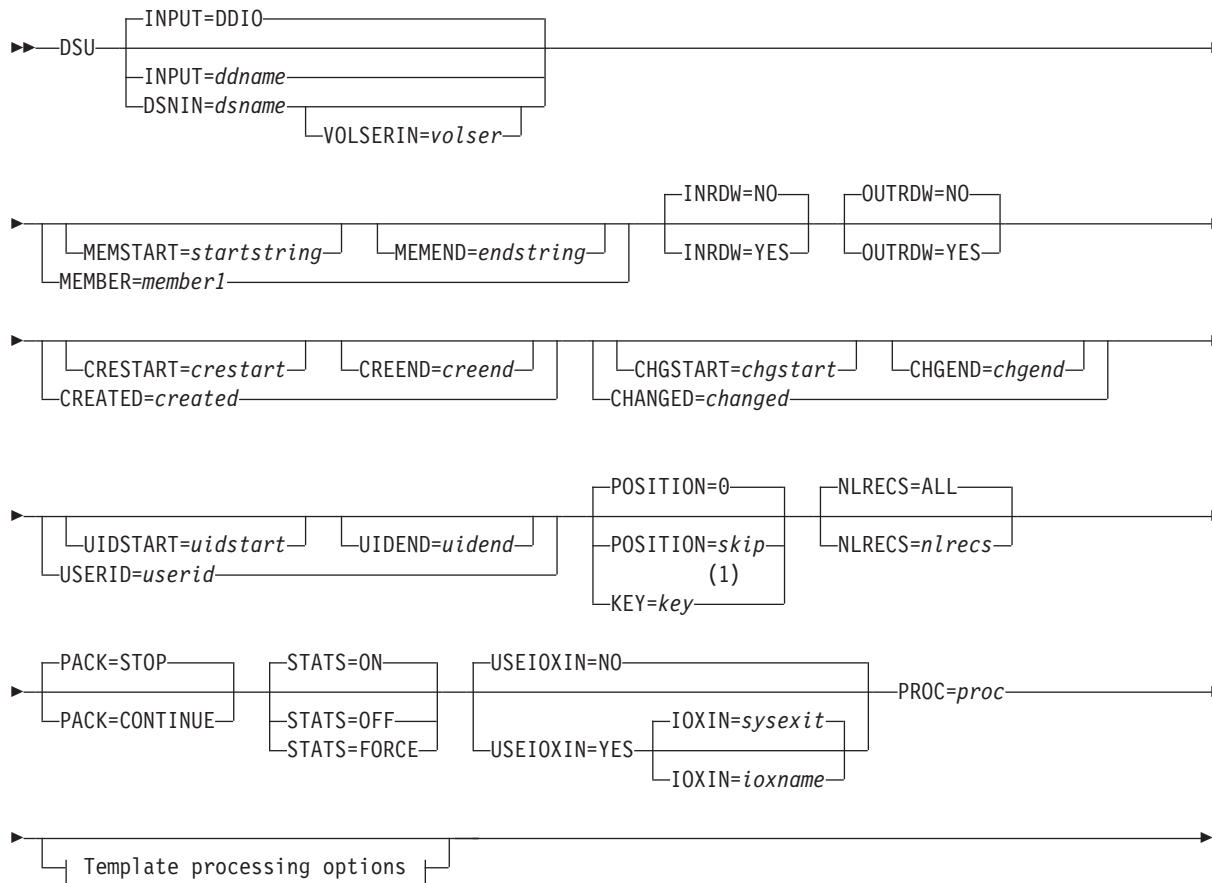
Note: Return codes can be customized during installation. If you receive return codes that do not match those listed above, your site might have customized the return codes in place for this function. File Manager may also issue the 999 abend, if the return code in batch is equal to or greater than the ABENDCC value. Please contact your File Manager systems administrator for details.

Related functions

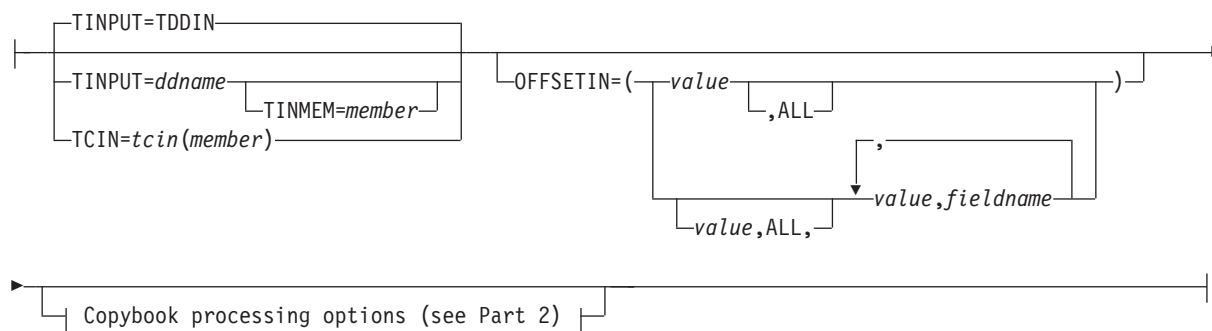
- DSEB** Edit a data set via batch job processing.
- DSX** Display the extents of a data set.

Function reference: DSU

Syntax: Part 1 of 2



Template processing options:

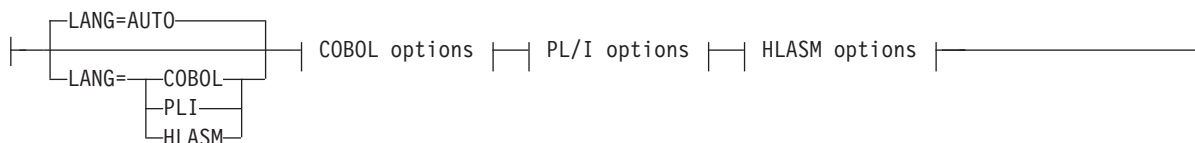


Notes:

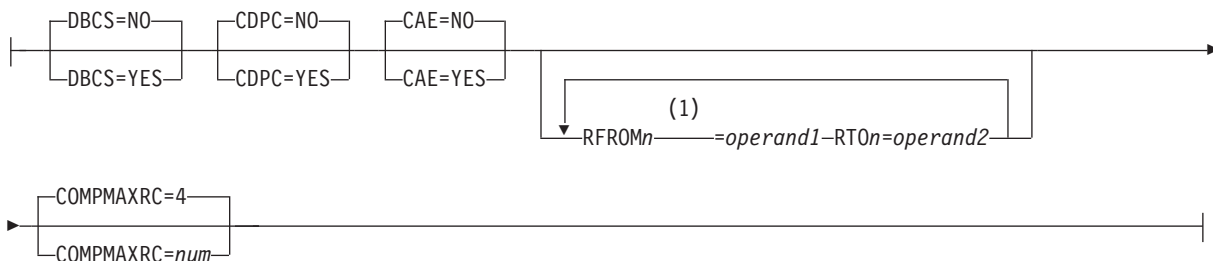
- 1 VSAM only.

Syntax: Part 2 of 2

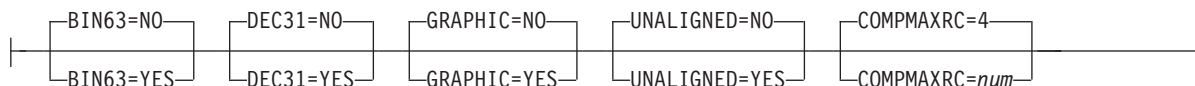
Copybook processing options (from Part 1):



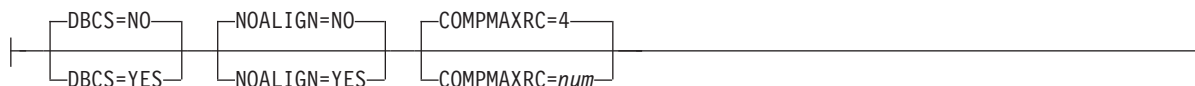
COBOL options:



PL/I options:



HLASM options:



Notes:

- 1 RFROM1 RTO1, RFROM2 RTO2, ... RFROM5 RTO5 ($n=1-5$).

INPUT=*ddname*

Defines a reference to a DD or TSO ALLOC statement for the input data set. The default is DDIN.

DSNIN=*dsname*

Defines the name of the input data set. If specified, any DD statement provided are not used. The name may include a member name in parenthesis. If the member is specified here, the associated Member parameter must be empty. You can further describe this data set, as follows:

VOLSERIN=*volser*

Volume serial number for a non-cataloged data set.

MEMBER=*member1*

The name of a single member in a PDS, or a member name pattern

Function reference: DSU

representing one or more members in a PDS. If the input data set is a PDS(E), you may specify this parameter, or a member name in the DD statement for *ddname*, or specify a range of member names with the MEMSTART and MEMEND keywords.

A member name pattern can consist of any characters that are valid in a member name and two special pattern characters: the asterisk (*) and the percent symbol (%).

* represents any number of characters. As many asterisks as required can appear anywhere in a member name pattern. For example, if you enter a member name pattern of **d**, all members in the PDS whose name contains "d" are processed.

% is a place holding character that means a single character. As many percent symbols as necessary can appear anywhere in a member name pattern. For example, if you enter a member name pattern of *%%%*, all members in the PDS whose name is four characters in length are processed.

member1 is ignored if the data set is not a PDS.

MEMSTART=*startstring*

Is used to specify the start of a range of member names to be included in the copy. If MEMSTART is specified but MEMEND is omitted, all members of the PDS(E) from the *startstring* value onwards are included. *startstring* can have the same values, including wild cards, as for the *member1* parameter of the MEMBER keyword.

MEMEND=*endstring*

Is used to specify the end of a range of member names to be included in the copy. If MEMEND is specified but MEMSTART is omitted, all members of the PDS(E) up to the *endstring* value onwards are included. *endstring* can have the same values, including wild cards, as for the *member1* parameter of the MEMBER keyword.

INRDW

Controls whether or not to adjust the input start location when the specified start location takes into account the record descriptor word (RDW).

NO Does not adjust the input start location.

YES Subtracts 4 from all start locations that have been coded on external functions that refer to the input record.

OUTRDW

Controls whether or not to adjust the output start location when the specified start location takes into account the record descriptor word (RDW).

NO Does not adjust the output start location.

YES Subtracts 4 from all start locations that have been coded on external functions that refer to the output record.

CREATED=*created*

The date on which a member was created, in YYYY/MM/DD format.

If the input data set is a PDS(E), you may specify this parameter, or specify a range of creation dates with the CRESTART and CREEND keywords.

You can specify an asterisk (*) as the last character to indicate a range of dates or a percent sign (%) in place of a single character to indicate a selection of dates.

created is ignored if the data set is not a PDS.

CRESTART=*crestart*

The start of a range of creation dates in YYYY/MM/DD format to be included in the copy.

If CRESTART is specified but CREEND is omitted, all members of the PDS(E) from the *crestart* value onwards are included.

If omitted, or you do not enter a full date, or you specify an asterisk (*) as the last character, the unspecified portion of *crestart* defaults to the right as follows:

DD = 01
MM = 01
YYYY = 0000

No other wildcarding is allowed.

CREEND=*creend*

The end of a range of creation dates in YYYY/MM/DD format to be included in the copy.

If omitted, or you do not enter a full date, or you specify an asterisk (*) as the last character, the unspecified portion of *creend* defaults to the right as follows:

DD = 31
MM = 12
YYYY = 9999

No other wildcarding is allowed.

CHANGED=*changed*

The date on which a member was last modified, in YYYY/MM/DD format.

If the input data set is a PDS(E), you may specify this parameter, or specify a range of modification dates with the CHGSTART and CHGEND keywords.

You can specify an asterisk (*) as the last character to indicate a range of dates or a percent sign (%) in place of a single character to indicate a selection of dates.

changed is ignored if the data set is not a PDS.

CHGSTART=*chgstart*

The start of a range of modification dates in YYYY/MM/DD format to be included in the copy.

If CHGSTART is specified but CHGEND is omitted, all members of the PDS(E) from the *chgstart* value onwards are included.

If omitted, or you do not enter a full date, or you specify an asterisk (*) as the last character, the unspecified portion of *chgstart* defaults to the right as follows:

DD = 01
MM = 01
YYYY = 0000

Function reference: DSU

No other wildcarding is allowed.

CHGEND=*chgend*

The end of a range of modification dates in YYYY/MM/DD format to be included in the copy.

If omitted, or you do not enter a full date, or you specify an asterisk (*) as the last character, the unspecified portion of *chgend* defaults to the right as follows:

DD = 31
MM = 12
YYYY = 9999

No other wildcarding is allowed.

USERID=*userid*

The TSO user ID by which the member was last updated.

If the input data set is a PDS(E), you may specify this parameter, or specify a range of user IDs with the UIDSTART and UIDEND keywords.

You can enter a generic user ID by using asterisks and percent signs.

userid is ignored if the data set is not a PDS.

UIDSTART=*uidstart*

The start of a range of user IDs to be included in the copy.

If UIDSTART is specified but UIDEND is omitted, all members of the PDS(E) from the *uidstart* value onwards are included.

If omitted, or you do not enter a full 7-character user ID, or you specify an asterisk (*) as the last character, File Manager replaces the asterisk and pads the unspecified portion of *uidstart* to the right with low values (X'00').

UIDEND=*uidend*

The end of a range of user IDs to be included in the copy.

If you omit this field, it defaults to high values (X'FF').

If you specify less than 7 characters (without an asterisk as the last character), File Manager pads *uidstart* to the right with low values (X'00').

If you specify an asterisk (*) as the last character, File Manager replaces the asterisk and pads the unspecified portion of *uidend* with high values (X'FF').

POSITION=*skip*

Number of logical records to be skipped from the beginning of the data set. The default is 0.

KEY=*key* (VSAM only)

A key for KSDS records, or a slot number for RRDS records. The maximum key length is 30 characters. The first record with a key or slot value greater than or equal to *key* is the first record updated. If you omit the *key* and *skip* values, updating begins with the first record in the data set.

If the key contains lowercase characters, blanks, or commas, enclose it in quotation marks. You can also specify a key in hexadecimal format (for example, X'C1C2C3').

NLRECS

Number of records to be printed or ALL.

ALL If you specify ALL or omit the parameter, all the remaining records are copied.

nrecs The maximum number is 99 999 999.

PACK Determines if File Manager should detect if the input data is in ISPF packed format.

STOP Default. File Manager detects whether the input data is in ISPF packed format, and if it is, stops the processing.

CONTINUE

File Manager does not detect whether the input data is in ISPF packed format and continues processing.

STATS=ON

Default. This updates the ISPF statistics (if already present) when a PDS or PDSE member has been changed.

STATS=OFF

The ISPF statistics is not updated when a PDS or PDSE member has been changed.

STATS=FORCE

The ISPF statistics that exist for members being processed are always updated and statistics for a member that previously did not have statistics are created.

USEIOXIN

Specifies whether to invoke a user I/O exit, to process the input data set.

NO Default. Do not invoke a user I/O exit.

YES Invoke a user I/O exit to process the input data set. This option is only available if the person who did the site customization for File Manager allowed user I/O exits on a site-wide basis.

IOXIN

Specifies the name of the user I/O exit used for the input data set. There are no restrictions on the programming language that you can use to write an exit. The exit must be provided to File Manager in the STEPLIB/ISPLLIB concatenation or their extensions (LINKLIST, LPA, and so on).

sysexit Default. If you specify USEIOXIN=YES and do not supply a user I/O exit name, File Manager uses the name of the exit provided in the installation customization options. If USEIOXIN has been set to YES and no installation default has been provided, you must specify IOXIN=*ioxname*.

Note: If you have selected batch processing in an online panel, the generated JCL statements use the default name provided in your Set System Processing Options panel.

ioxname

The name of a PDS(E) member of a data set that has been provided to File Manager in the STEPLIB concatenation.

PROC=*proc*

Member name of a REXX procedure that you want to use to process each record before it is updated, or an asterisk (*) to indicate the REXX procedure is inline. If you specify a member name, you must define an FMNEXEC ddname that identifies the PDS containing the member. If you

Function reference: DSU

specify *, the procedure is read from SYSIN immediately following the control statement for the current function. The inline procedure is terminated by a record containing a slash and a plus sign (/+) in columns 1-2.

For more information about using REXX procedures to process records before they are updated, see Chapter 13, "Enhancing File Manager processing," on page 385.

Template processing

Define which template (if any) is used to describe the record structure in the input data set, and how File Manager processes this template.

TINPUT=*ddname*

Defines a reference to a DD or TSO ALLOC statement for the data sets which contain the copybook or template that describes the record structure of your input data. The default is TDDIN.

If you specify a concatenated DD, then you must provide the member name, *member*.

TINMEM=*member*

The name of the copybook or template member in the datasets identified by the TINPUT parameter if it has not been specified on the DD statement. This parameter must not be specified if the TCIN parameter is specified.

TCIN=*tcin(member)*

PDS and member name of the copybook or template that describes the record structure of your input data.

Note: If you specify a template for DSEB and DSU, it is ignored, except for calls to the external REXX function PRINT specifying TABL or SNGL format.

OFFSETIN

The length of the 01 field in the template and the start locations of the fields within that 01 field are adjusted by the value provided.

value The offset value, which must be in the range -32760 to 32760, to be applied to the corresponding field identifier. If no field identifier is supplied and ALL is not used, the value is applied to the first Level 01 field in the template.

ALL Where the template contains multiple record structures, this keyword applies the corresponding *value* to all Level 01 fields within the template.

Note: You can specify a value for ALL and then override this value for individual layouts by providing subsequent *value* and *fieldname* combinations.

fieldname

The name of the Level 01 field to which *value* is to be applied. The default is the first Level 01 field in the template.

Copybook processing

If you specify a copybook (instead of an existing template), then File Manager uses these processing options to compile the copybook into a template:

LANG

Determines whether File Manager automatically detects the copybook language or interprets the language as COBOL, PL/I, or HLASM.

AUTO

Automatically detect whether the copybook language is COBOL or PL/I, and invoke the appropriate compiler. If the compilation results in a return code greater than 4, then invoke the compiler for the other language. If the second compilation also results in a return code greater than 4, then retry the first compiler and report the compilation errors. If File Manager successfully creates a template (despite the compilation errors), then continue processing with the template.

COBOL

Invoke the COBOL compiler to create a template from the copybook. (Do not invoke the PL/I compiler, even if the COBOL compilation results in errors.)

PLI

Invoke the PL/I compiler to create a template from the copybook. (Do not invoke the COBOL compiler, even if the PL/I compilation results in errors.)

HLASM

Invoke the HLASM compiler to create a template from the copybook.

COBOL options

The following options are used to compile a COBOL copybook into a template:

DBCS=YES

Use the DBCS compiler option.

DBCS=NO

Use the NODBCS compiler option.

For details on the effect of the DBCS and NODBCS compiler options, see the *IBM COBOL Programming Guide for OS/390 & VM*.

CDPC=NO

Do not use the COBOL SPECIAL-NAMES paragraph "Decimal-point is comma".

CDPC = YES

Use the COBOL SPECIAL-NAMES paragraph "Decimal-point is comma".

CAE=NO

Do not use the COBOL compile option ARITH(EXTEND).

CAE = YES

Use the COBOL compile option ARITH(EXTEND).

MIXED = NO

Field names stored in the template in uppercase.

MIXED = YES

Field names stored in the template in the original case as coded in the COBOL copybook.

RFROM1 RTO1 ... RFROM5 RTO5

Up to five pairs of “From” and “To” pseudo-text character strings for the COBOL REPLACE compiler-directing statement.

If your COBOL copybooks contain characters that you want to remove or replace with other characters before compiling the copybooks into templates, then use these replacing options.

For example, if your copybooks contain colon characters (:) that you want to remove before compiling, then specify '=:==' as *operand1* and '====' as *operand2*.

For details on specifying “From” and “To” strings for COBOL REPLACE, see the *IBM COBOL Language Reference*.

COMPMAXRC

Sets the maximum acceptable return code for a copybook compile. A return code higher than the specified level causes the function to stop. Default is 4.

PL/I options

The following options are used to compile a PL/I copybook into a template:

BIN63=YES Use the LIMITS(FIXEDBIN(63)) compiler option.

BIN63=NO Use the LIMITS(FIXEDBIN(31)) compiler option.

DEC31=YES Use the LIMITS(FIXEDDEC(31)) compiler option.

DEC31=NO Use the LIMITS(FIXEDDEC(15)) compiler option.

GRAPHIC=YES

Use the GRAPHIC compiler option.

GRAPHIC=NO

Use the NOGRAPHIC compiler option.

UNALIGNED=YES

Use the DEFAULT RANGE (*) UNALIGNED, language statement to change the default alignment.

UNALIGNED=NO

Use the PL/I default.

COMPMAXRC

Sets the maximum acceptable return code for a copybook compile. A return code higher than the specified level causes the function to stop. Default is 4.

For details on the effect of these compiler options, see the *IBM VisualAge PL/I for OS/390 Programming Guide*.

HLASM options

The following options are used to compile a HLASM copybook into a template:

DBCS=YES Use the DBCS compiler option.

DBCS=NO Use the NODBCS compiler option.

NOALIGN=YES

Use the NOALIGN compiler option.

NOALIGN=NO

Use the ALIGN compiler option.

COMPMAXRC

Sets the maximum acceptable return code for a copybook compile. A return code higher than the specified level causes the function to stop. Default is 4.

For details on the effect of these compiler options, see the *HLASM V1R5 Programmer's Guide*.

Batch example

```
//DSU JOB (acct),'name' PDS Member Update
//*
//FMBAT PROC
//FMBAT EXEC PGM=FILEMGR
//STEPLIB DD DSN=FMN.SFMNMOD1,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSABEND DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
// PEND
//*
//FILEMGR EXEC FMBAT
//FMNTSPRT DD SYSOUT=*
//JCLPDS DD DSN=FMNUSER.FMOS390.JCL,DISP=SHR
//SYSIN DD *
$$FILEM DSU INPUT=JCLPDS,MEMBER=*,PROC=*
/* Translate all records to uppercase */
Upper outrec
Return
/+
$$FILEM EOJ
/*

//DSU JOB (acct),'name' Fix post code
//*
//FMBAT PROC
//FMBAT EXEC PGM=FILEMGR
//STEPLIB DD DSN=FMN.SFMNMOD1,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSABEND DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
// PEND
//*
//FILEMGR EXEC FMBAT
//FMNTSPRT DD SYSOUT=*
//SYSIN DD *
$$FILEM DSU DSNIN=FMNUSER.FMOS390.TRANRECS,
$$FILEM PROC=*
/* Locate name and address record for James */
/* Browne and change postcode, stored in */
/* packed decimal, from 6011 to 6194 */
If Substr(inrec,1,1) == 'A' &
   Substr(inrec,32,5) == 'James' &
   Substr(inrec,57,6) == 'Browne' then
   outrec = Change(outrec,'06011F'x,'06194F'x,1,125,3)
Return
/+
$$FILEM EOJ
/*
```

Replacement pages for “FCH (Find/Change)” section

FCH (Find/Change)

Purpose

The FCH function allows you to:

- Search for, and optionally change, strings in a PDS, VSAM data set, or sequential data set.
- Search for strings in HFS files.

Usage notes

You can select the records to be processed using:

- Member name selection criteria
- Date created selection criteria
- Date last modified selection criteria
- User ID selection criteria

You can either specify a REXX procedure with the *proc* parameter, or enter a FIND or CHANGE command in the Command line. For information about the primary commands, see “Finding and changing data in multiple PDS members” on page 246. The LOCATE primary command is ignored in batch jobs. When working with compressed non-VSAM extended format data sets (compressed PSE data sets), the CHANGE command is not supported, however, the FIND command can be used.

Multiple command processing

There is no limit on the number of FIND or CHANGE commands that can be processed in one pass of the file, but each FIND or CHANGE command must start on a new line. Be careful when using overlapping change commands such as C cat dog and C catapult crossbow. A string is only matched against the first command with a matching search argument. Therefore, you must place the longer change first. For example, if you specify the following change commands:

```
C Cat Dog  
C Catapult Crossbow
```

The second command would never get processed. Reversing the command order would ensure any occurrences of “Catapult” were changed.

After a change is made, FCH processing continues for the same data record. The point where processing resumes is immediately following the most recently changed string. As a result, changes are not recursive, so that C cat cow and C cow dog does not change “cat” to “dog”, unless separate runs are done. Furthermore, if the search argument is found but the change fails, subsequent FIND or CHANGE commands that match that string are not done.

Performance tips

- When you use FCH to update members of a PDS(E):
One FCH default is STATS=ON, which causes the ISPF statistics for each changed member to be updated. This can significantly increase I/O (EXCP) and CPU utilization. To improve performance, consider using STATS=OFF.
- Using JCL processing (JCL=YES) is more CPU intensive than JCL=NO. Only use JCL=YES if necessary.

- You can improve concurrent read access by other users or jobs to the target data set when there is a user PROC, by using the NOUPDATE=YES option when the PROC will not be performing any updates.

Options

When you specify the PROC option, you are supplying a REXX procedure. For more information, see the *proc* parameter below.

Return codes

The default return codes from the FCH function have the following modified meanings:

- | | |
|----|---|
| 1 | One or more FIND or CHANGE commands successful but one or more FIND or CHANGE commands unsuccessful because no strings found. |
| 2 | One or more strings found but one or more CHANGE commands could not be performed (no space available or invalid key change). |
| 4 | No FIND or CHANGE command successful because no strings found (no matches). |
| 4 | No FIND or CHANGE command successful because no members to process. |
| 4 | No FIND or CHANGE command successful because input empty. |
| 8 | Bad FIND/CHANGE command(s) supplied. |
| 8 | Too many FIND/CHANGE commands supplied. |
| 8 | REXX error encountered. |
| 8 | Job step interrupted/cancelled. |
| 16 | Program Object not supported (but specified). |
| 16 | Data set in use. |
| 16 | Member in use. |
| 16 | Data set/member open error. |
| 16 | Data set not found / allocation error. |
| 16 | Insufficient storage available. |
| 16 | Input data appears ISPF packed but is not valid. |
| 16 | FCH abended. |
| 16 | Other serious error that stops processing occurred (for example an input/output error). |

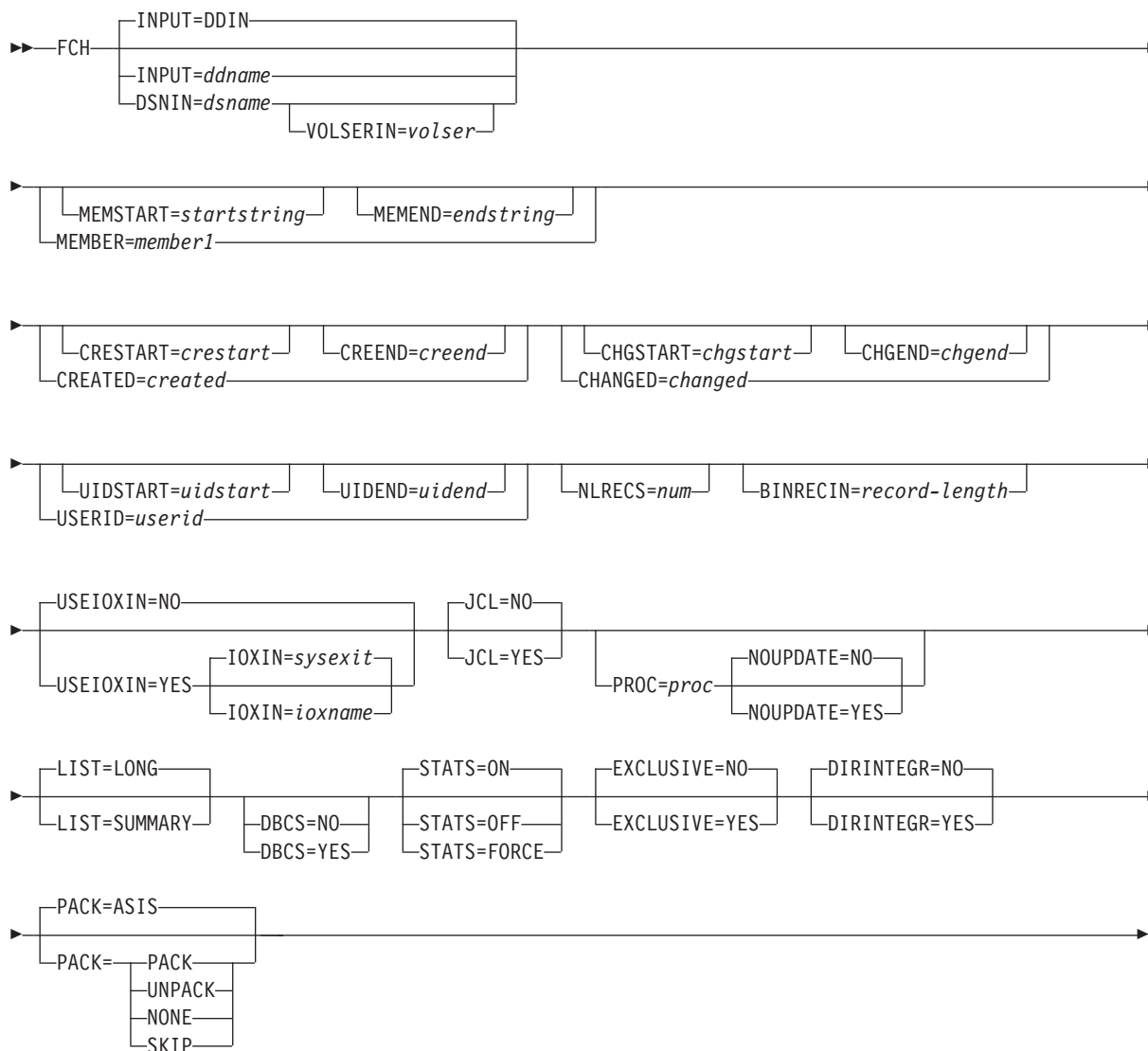
Note: Return codes can be customized during installation. If you receive return codes that do not match those listed above, your site might have customized the return codes in place for this function. File Manager may also issue the 999 abend, if the return code in batch is equal to or greater than the ABENDCC value. Please contact your File Manager systems administrator for details.

Related functions

- | | |
|-------------|--|
| DSEB | Edit a data set via batch job processing |
| TRS | Locate data within a tape file |

Function reference: FCH

Syntax



INPUT=ddname

Defines a reference to a DD or TSO ALLOC statement for the input data set, or HFS file. The default is DDIN.

DSNIN=dsname

Defines the name of the input data set, or an absolute path to a HFS file (directory). . If specified, any DD statement provided are not used. The name may include a member name in parenthesis. If the member is specified here, the associated Member parameter must be empty. You can further describe this data set, as follows:

VOLSERIN=volser

Volume serial number for a non-cataloged data set.

An absolute path to a HFS file (directory) must be enclosed in apostrophes. If it does not fit on one line, it can be split into several lines.

MEMBER=*member1*

The name of a single member in a PDS, or a member name pattern representing one or more members in a PDS. If the input data set is a PDS(E), you may specify this parameter, or a member name in the DD statement for *ddname*, or specify a range of member names with the MEMSTART and MEMEND keywords.

A member name pattern can consist of any characters that are valid in a member name and two special pattern characters: the asterisk (*) and the percent symbol (%).

- * represents any number of characters. As many asterisks as required can appear anywhere in a member name pattern. For example, if you enter a member name pattern of *d*, all members in the PDS whose name contains "d" are processed.
- % is a place holding character that means a single character. As many percent symbols as necessary can appear anywhere in a member name pattern. For example, if you enter a member name pattern of %%%%, all members in the PDS whose name is four characters in length are processed.

member1 is ignored if the data set is not a PDS.

MEMSTART=*startstring*

Is used to specify the start of a range of member names to be included in the copy. If MEMSTART is specified but MEMEND is omitted, all members of the PDS(E) from the *startstring* value onwards are included. *startstring* can have the same values, including wild cards, as for the *member1* parameter of the MEMBER keyword.

MEMEND=*endstring*

Is used to specify the end of a range of member names to be included in the copy. If MEMEND is specified but MEMSTART is omitted, all members of the PDS(E) up to the *endstring* value onwards are included. *endstring* can have the same values, including wild cards, as for the *member1* parameter of the MEMBER keyword.

CREATED=*created*

The date on which a member was created, in YYYY/MM/DD format.

If the input data set is a PDS(E), you may specify this parameter, or specify a range of creation dates with the CRESTART and CREEND keywords.

You can specify an asterisk (*) as the last character to indicate a range of dates or a percent sign (%) in place of a single character to indicate a selection of dates.

created is ignored if the data set is not a PDS.

CRESTART=*crestart*

The start of a range of creation dates in YYYY/MM/DD format to be included in the copy.

If CRESTART is specified but CREEND is omitted, all members of the PDS(E) from the *crestart* value onwards are included.

Function reference: FCH

If omitted, or you do not enter a full date, or you specify an asterisk (*) as the last character, the unspecified portion of *crestart* defaults to the right as follows:

DD = 01
MM = 01
YYYY = 0000

No other wildcarding is allowed.

CREEND=*creend*

The end of a range of creation dates in YYYY/MM/DD format to be included in the copy.

If omitted, or you do not enter a full date, or you specify an asterisk (*) as the last character, the unspecified portion of *creend* defaults to the right as follows:

DD = 31
MM = 12
YYYY = 9999

No other wildcarding is allowed.

CHANGED=*changed*

The date on which a member was last modified, in YYYY/MM/DD format.

If the input data set is a PDS(E), you may specify this parameter, or specify a range of modification dates with the CHGSTART and CHGEND keywords.

You can specify an asterisk (*) as the last character to indicate a range of dates or a percent sign (%) in place of a single character to indicate a selection of dates.

changed is ignored if the data set is not a PDS.

CHGSTART=*chgstart*

The start of a range of modification dates in YYYY/MM/DD format to be included in the copy.

If CHGSTART is specified but CHGEND is omitted, all members of the PDS(E) from the *chgstart* value onwards are included.

If omitted, or you do not enter a full date, or you specify an asterisk (*) as the last character, the unspecified portion of *chgstart* defaults to the right as follows:

DD = 01
MM = 01
YYYY = 0000

No other wildcarding is allowed.

CHGEND=*chgend*

The end of a range of modification dates in YYYY/MM/DD format to be included in the copy.

If omitted, or you do not enter a full date, or you specify an asterisk (*) as the last character, the unspecified portion of *chgend* defaults to the right as follows:

DD = 31
MM = 12

YYYY = 9999

No other wildcarding is allowed.

USERID=userid

The TSO user ID by which the member was last updated.

If the input data set is a PDS(E), you may specify this parameter, or specify a range of user IDs with the UIDSTART and UIDEND keywords.

You can enter a generic user ID by using asterisks and percent signs.

userid is ignored if the data set is not a PDS.

UIDSTART=uidstart

The start of a range of user IDs to be included in the copy.

If UIDSTART is specified but UIDEND is omitted, all members of the PDS(E) from the *uidstart* value onwards are included.

If omitted, or you do not enter a full 7-character user ID, or you specify an asterisk (*) as the last character, File Manager replaces the asterisk and pads the unspecified portion of *uidstart* to the right with low values (X'00').

UIDEND=uidend

The end of a range of user IDs to be included in the copy.

If you omit this field, it defaults to high values (X'FF').

If you specify less than 7 characters (without an asterisk as the last character), File Manager pads *uidstart* to the right with low values (X'00').

If you specify an asterisk (*) as the last character, File Manager replaces the asterisk and pads the unspecified portion of *uidend* with high values (X'FF').

NLRECS=num

Specifies the number of records to be processed in each data set or member.

BINRECIN=record-length

Specifies the record length used for processing a HFS file. Valid range: 1 to 32760. The file is processed in binary mode (fixed-length records derived from the file, delimiters not distinguished). If you do not specify this parameter, the file is processed in text mode (variable-length records, boundaries determined by delimiters).

USEIOXIN

Specifies whether to invoke a user I/O exit, to process the input data set.

NO Default. Do not invoke a user I/O exit.

YES Invoke a user I/O exit to process the input data set. This option is only available if the person who did the site customization for File Manager allowed user I/O exits on a site-wide basis.

IOXIN

Specifies the name of the user I/O exit used for the input data set. There are no restrictions on the programming language that you can use to write an exit. The exit must be provided to File Manager in the STEPLIB/ISPLLIB concatenation or their extensions (LINKLIST, LPA, and so on).

sysexit Default. If you specify USEIOXIN=YES and do not supply a user I/O exit name, File Manager uses the name of the exit provided in

Function reference: FCH

the installation customization options. If USEIOXIN has been set to YES and no installation default has been provided, you must specify IOXIN=*ioxname*.

Note: If you have selected batch processing in an online panel, the generated JCL statements use the default name provided in your Set System Processing Options panel.

ioxname

The name of a PDS(E) member of a data set that has been provided to File Manager in the STEPLIB concatenation.

JCL=NO

Treat the data set as a non-JCL data set.

JCL=YES

The data set contains JCL and the JCL syntax is to be preserved.

The columns searched are set to 3 through 71, unless the statement is not a JCL statement. A statement is considered to be a JCL statement if it begins with the strings `"/*` or `"/`. If the statement does not begin with either of these strings, it is not considered to be a JCL statement in which case any column range specified on the FIND (or CHANGE, respectively) command or preset using the BOUNDS command is honored. If no column range has been specified, the full record is searched.

If not successful at maintaining the number and size of records, File Manager attempts to rewrite the file:

- More errors are possible in this case. For example, a PDS(E) may run out of room.
- If a logical line is changed and requires more physical records, the file is rewritten. The data in columns 73–record length for new physical records is copied from the last related original physical record. Data past column 72 is treated as non-changeable sequence numbers or comments.

PROC=*proc*

Member name of a REXX procedure that you want to use to process each record, or an asterisk (*) to indicate the procedure is inline. If you specify a member name, you must define an FMNEXEC ddname that identifies the PDS containing the member. If you specify *, the procedure is read from SYSIN immediately following the control statement for the current function. The inline procedure is terminated by a record containing a slash and a plus sign (/+) in columns 1–2.

Whether or not a record appears in the FCH report is determined by the return code from the REXX procedure for that record. (If no PROC statement is specified, one is assumed at the end of the \$FILEM control statements.) If the REXX procedure ends with a RETURN DROP statement, then the current record is considered to be “not selected” (not one of the records you wanted to find), and does not appear in the FCH report. If the REXX procedure ends normally, or with an explicit RETURN (without the DROP keyword), then the current record is considered to be “selected”, and is included in the FCH report. Records that have been selected without being changed by the REXX procedure are marked in the FCH report by an “s” suffix attached to their record number, while records that have been selected and changed are marked by a “c”.

In a REXX procedure for FCH, explicitly code a RETURN statement when you identify a record that you want to select. To ensure that other records are not selected, on the last line of the REXX procedure, code a RETURN DROP statement.

For more information about using REXX procedures to process records, see Chapter 13, “Enhancing File Manager processing,” on page 385.

NOUPDATE

Allows you to specify that you intend no updates to the FCH data set while executing the utility. This option is valid only when a REXX procedure has been specified and is ignored otherwise.

NO Updates to the data are honored.

YES Forces the allocation of the data set as input only. All updates to the data are ignored.

LIST=LONG

Default. This prints each record where the string was found as well as a summary report.

LIST=SUMMARY

This produces a summary report only.

DBCS=YES

(Default for LANGUAGE=JAPANESE). This processes and preserves DBCS shiftin and shiftout characters within the data records.

DBCS=NO

(Default for LANGUAGE=ENGLISH). This ignores DBCS shiftin and shiftout characters within the data records.

STATS=ON

Default. This updates the ISPF statistics (if already present) when a PDS or PDSE member has been changed.

STATS=OFF

The ISPF statistics is not updated when a PDS or PDSE member has been changed.

STATS=FORCE

The ISPF statistics that exist for members being processed are always updated and statistics for a member that previously did not have statistics are created.

EXCLUSIVE=NO

Note: This option is supported for backward compatibility only.

Use the new DIRINTEGR option.

Default. The data set is allocated with DISP=SHR, so that other users can obtain concurrent access to a PDS or PDSE during execution of FCH.

EXCLUSIVE=YES

Forces an override of the PDS(E) member processing method which allows for safe concurrent updates by other users. This option has significant performance impact. When set to YES, the member processing is performed much faster but may be affected by PDS(E) directory updates, possibly causing I/O errors if the data set is concurrently updated. This option overrides the processing method selected by File Manager (EXCLUSIVE=NO, default, unless the input data set has been allocated

Function reference: FCH

OLD by the user), which always assumes concurrent safe processing when the data set is allocated to multiple users.

DIRINTEGR

Specifies whether to invoke a user I/O exit to process the input data set.

NO Default. File Manager uses a faster PDS(E) directory processing method. This may cause I/O errors when multiple users are concurrently updating the directory of the data set being processed.

YES File Manager uses safer, but slower, PDS(E) directory processing method. This method allows for safe concurrent updates of the PDS(E) directory by multiple users.

PACK Determines if File Manager should detect if the input data is in ISPF packed format and specifies if the output data is to be written in ISPF packed format. This keyword is ignored when processing VSAM data sets. When an I/O exit has been specified for either the input or output data set (or both), the only valid option is PACK=NONE.

ASIS Instructs File Manager to write the output in ISPF Packed format only if the input is in ISPF packed format.

PACK Instructs File Manager to write the output in ISPF packed format regardless of the input format.

UNPACK

Instructs File Manager to write the output without ISPF packing, regardless of the input format.

NONE

Instructs File Manager not to determine if the input data set is in ISPF packed format and writes the output records as they are read from the input data set (after any enhanced processing).

SKIP Instructs File Manager to determine if the input data set is in ISPF packed format and if so, to skip the find/change processing.

See "Finding and changing data in multiple PDS members" on page 246 for details of the FCH commands.

Note: It is not possible to use multiple FINDNOT commands in the input stream for batch processing. Similarly, it is not possible to combine FINDNOT commands with FIND and/or CHANGE commands in the batch input stream.

Batch example

```
//FMUSRFCH JOB (@TS2,MVS6),'FMUSER',NOTIFY=FMUSER,
// CLASS=A,MSGLEVEL=(1,1),MSGCLASS=H
//FMNBAT EXEC PGM=FILEMGR
//STEPLIB DD DSN=FMUSER.FMN110.TSTLOAD,DISP=SHR
// DD DSN=FMN.V1R1M0.TSTLOAD,DISP=SHR
// DD DSN=FMN.V1R1M0.SFMNMOD1,DISP=SHR
// DD DSN=FMN.IGYV1R20.SIGYCOMP,DISP=SHR
//SYSPRINT DD SYSOUT=*
//FMNEXEC DD DSN=FMN.EXEC,DISP=SHR
//FMNTSPRT DD SYSOUT=*
//FMNIN DD DSN=FMUSER.JCL.TESTING,DISP=SHR
//FMNOUT DD SYSOUT=*
//SYSTEM DD SYSOUT=*
//SYSIN DD *
$$FILEM FCH ,
$$FILEM INPUT=FMNIN,MEMBER=J*
F 'rights reserved'
C 'Copyright 2001-2002' ,
'Copyright 2001-2003' 1 71
C 'Alpha Company Ltd' ,
'Alpha Beta Company Ltd' 1 71
C 'Beta Company Ltd' ,
'Alpha Beta Company Ltd' 1 71
/+
/*
```


Replacement pages for “External REXX functions” section

External REXX functions

This section describes the REXX external functions that you can use when writing REXX procedures to enhance a File Manager function. In addition, File Manager provides access to the REXX internal functions described in the *z/OS TSO/E REXX Reference*.

CHANGE

Changes a specified character string.

CHG_OUT

(Can be used in FASTREXX procedures.) Changes a character string in the output record.

CHG_VAR

(Can be used in FASTREXX procedures.) Changes one or more occurrences of an old string in a variable to a new string.

CONTAINS

Checks for character values in a specified string.

FLD (Can be used in FASTREXX condition expressions or internally processed criteria expressions.) Refers to a field from the current input record.

FLD_CO

(Can be used in FASTREXX condition expressions or internally processed criteria expressions.) Searches a field in the input record for one or more occurrences of a string, or tests a field in the input record for one more numeric values.

FLD_OUT

(Can be used in FASTREXX procedures.) Overlays the output record with a field from the input record.

FLD_TM

(Can be used in FASTREXX condition expressions or internally processed criteria expressions.) Tests selected bits of a field in the input record.

FLD_TYPE

(Can be used in FASTREXX condition expressions or internally processed criteria expressions.) Tests the data type of a field in the input record.

FLDI (Can be used in FASTREXX procedures.) Performs a conditional test against an input record field.

FLDO (Can be used in FASTREXX procedures.) Performs a conditional test against an output record field.

I_LENGTH

(Can be used in FASTREXX condition expressions or internally processed criteria expressions.) Returns the length of the input record.

MOD_DATE

(Can be used in FASTREXX procedures.) Sets, increments, or decrements a date field using year, month, or day values.

NCONTAIN

Checks for numeric values in a specified string.

O_LENGTH

(Can be used in FASTREXX condition expressions or internally processed criteria expressions.) Returns the current length of the output record.

Function reference: External REXX functions

OFLD_CO

(Can be used in FASTREXX condition expressions or internally processed criteria expressions.) Searches a field in the output record for one or more occurrences of a string, or tests a field in the output record for one or more numeric values, and resets the current output relative position (OUTPOS) accordingly.

OVLY_OUT

(Can be used in FASTREXX procedures.) Overlays the output record with a literal (constant) or variable value.

OVLY_VAR

(Can be used in FASTREXX procedures.) Overlays the named character variable with a string.

PRINT

Prints a record.

PRTCOUNT

(Can be used in FASTREXX condition expressions or internally processed criteria expressions.) Returns the count of records printed.

RECSIN

(Can be used in FASTREXX condition expressions or internally processed criteria expressions.) Returns the count of records read.

RECSOUT

(Can be used in FASTREXX condition expressions or internally processed criteria expressions.) Returns the count of records written to a given data set.

RSTR_OUT

(Can be used in FASTREXX condition expressions.) Restores the most recently saved copy of the output buffer.

SAVE_OUT

(Can be used in FASTREXX condition expressions.) Saves a copy of the current output buffer.

SET_OLEN

(Can be used in FASTREXX procedures.) Sets the length of the output record.

SETC

(Can be used in FASTREXX procedures.) Defines or changes a character variable.

SETN

(Can be used in FASTREXX procedures.) Defines or changes a numeric variable.

TESTC

(Can be used in FASTREXX procedures.) Performs a conditional test against a character variable.

TESTN

(Can be used in FASTREXX procedures.) Performs a conditional test against a numeric variable.

TALLY

(Can be used in FASTREXX procedures.) Totals a field value and reports the total.

TFLD

(Can be used in FASTREXX condition expressions or internally processed criteria expressions.) Searches a field in the input record for one or more

occurrences of a string, or tests a field in the input record for one more numeric values. For dimensioned fields, you can search any or all of the elements of the array.

TM Tests a string for a bit value.

VAR_OUT

(Can be used in FASTREXX procedures.) Overlays the output record with a field from a variable.

VAR_TM

(Can be used in FASTREXX procedures.) Tests selected bits of a field in a variable.

WRITE

(Can be used in FASTREXX procedures.) Writes a record.

The following REXX external functions can only be used with DSEB (Data Set Edit Batch):

BOT Move to the last record

DOWN

Move down (forwards) a specified number of records

FINDNEXT

Search for a string from the current record forwards

FINDPREV

Search for a string from the current record backwards

RECCUR

Return the current record number

TOP Move to the first record

UP Move up (backwards) a specified number of records

UPDATE

Replace the current input record with the value in OUTREC

Note: You can only use these File Manager-specific REXX external functions, and the INREC and OUTREC variables, in a REXX procedure specified by the PROC parameter of a File Manager function (or, when using panels, by the **Use REXX proc** field). You cannot use these functions and variables in REXX procedures outside of this File Manager environment.

Absolute and relative positioning in external REXX functions

All File Manager external REXX functions that refer to positions within the input or output record can use absolute values to determine the byte location within the record. For example, the FLD function syntax is:

```
FLD(start_column,length,type)
```

where *start_column* can be an integer that refers to a specific byte in the input record.

However, some external REXX functions also allow positions to be specified as an offset value, relative to the “current position” within the input or output record. The current position is initialized as each record is processed but can be modified by these functions.

Function reference: External REXX functions

For input records, this allows you to perform tasks such as searching for a string in the input record and then testing or copying the contents of a field relative to the located string.

For output records, this allows you to easily append fields or constants at positions relative to the most recently updated output record field. For example, you could append a number of constants and fields, one after the other in the output record, without needing to keep track of the exact current starting position or manually updating the starting position with the length of added fields.

Note: You cannot use the REXX external functions to write data beyond the logical record limitations of the output data set. For example, when using a Fixed Block data set that has an LRECL of 80, you cannot write data to position 81 or beyond.

The following functions support relative positioning:

- CHG_OUT
- CHG_VAR
- FLD
- FLD_CO
- FLD_OUT
- FLD_TM
- FLD_TYPE
- FLDI
- FLDO
- MOD_DATE
- OVLY_OUT
- OVLY_VAR
- SETC
- SETN
- TESTC
- TESTN
- VAR_OUT
- VAR_TM

To maintain the current position within the input and output records, these functions use two internal variables, INPOS and OUTPOS. These values have not been externalized to the REXX environment and can only be accessed or altered indirectly, by using the functions listed above or SET_OLEN (which does not support relative position arguments but does alter the value of OUTPOS in some circumstances).

INPOS

For each new record that is processed, INPOS is set to 1. INPOS is then changed for the current input record whenever the FLD_CO function or FLDI function (with contains operator) executes a successful search for a *needle* that is a string (type C or U). If a *needle* is found, INPOS is set to the first byte of the located *needle*. If no *needle* is found, INPOS is unchanged.

OUTPOS

For each new record that is processed, OUTPOS is set to 1 greater than the length of the current output record. The output record is initially the same length as the input record, unless templates have been used to reformat the record. OUTPOS is then modified (as a side effect) when the following functions are used:

CHG_OUT

CHG_OUT sets OUTPOS to one byte past the end of the last changed field in the output record.

FLD_OUT

FLD_OUT sets OUTPOS to one byte past the end of the field being overlaid in the output record.

OVLY_OUT

OVLY_OUT sets OUTPOS to one byte past the end of the field being overlaid in the output record.

SET_OLEN

SET_OLEN only changes OUTPOS if it truncates the output record so that the existing OUTPOS becomes greater than the reduced record length. In this case, OUTPOS is reset to the reduced length plus 1.

FLDO with contains operator

If a *needle* is found, OUTPOS is set to the first byte of the located *needle*. If no *needle* is found, OUTPOS is unchanged.

Specifying relative positions

Relative positioning is specified when you use a special character string, in the form of <type><offset>, in place of an integer in the *start* argument of a supported function. The relative position can be specified in the following ways:

- IP x** The start position is taken from the current INPOS (the “I” in <type>) and then offset a positive (the “P” in <type>) number of bytes, as specified by x . For example, if the INPOS was currently 20 and you specified a *start* argument of IP5, the start position would be 25.
- IN x** The start position is taken from the current INPOS and then offset a negative (the “N” in <type>) number of bytes, as specified by x . For example, if the INPOS was currently 20 and you specified a *start* argument of IN5, the start position would be 15.
- OP x** The start position is taken from the current OUTPOS (the “O” in <type>) and then offset a positive number of bytes, as specified by x . For example, if the OUTPOS was currently 20 and you specified a *start* argument of OP5, the start position would be 25.
- ON x** The start position is taken from the current OUTPOS and then offset a negative number of bytes, as specified by x . For example, if the OUTPOS was currently 20 and you specified a *start* argument of ON5, the start position would be 15.

When the start argument’s natural target is the input record, **IP x** and **IN x** can be abbreviated to **P x** or **N x** .

When the argument’s natural target is the output record, **OP x** and **ON x** can be abbreviated to **P x** or **N x** .

When the argument’s natural target is a variable, **IP x** , **IN x** , **OP x** , **ON x** , can be used to denote the current relative variable position. They can be abbreviated to **P x** or **N x** .

For example, `FLD(start_column,length,type)` reads from the input record, so you could specify *start_column* as IP5 or P5 and get the same result. On the other hand,

Function reference: External REXX functions

if you wanted to use the current value of OUTPOS to specify the *start_column* in the input record, you must specify the <type> in full, that is, as OP5.

Using FASTREXX variables

File Manager supports these variables for FASTREXX processing:

- System numeric variables. See Table 16 on page 1069 below for descriptions.
- System character variables. See Table 15 below for descriptions.
- User character variables.
- User numeric variables.
- Tally registers.

System and tally variables are maintained by File Manager and are read-only to user procedures.

User variables are defined by SETC or SETN functions (there is an implied definition in TESTC and TESTN functions for variables that do not exist). These variables can be referenced and modified by these functions:

Table 14. Batch update status and action

Function	Reference	Modify
CHG_VAR	Y	Y
CHG_OUT	Y	N
FLDI	Y	N
FLDO	Y	N
OVLY_VAR	Y	Y
OVLY_OUT	Y	N
SETC	Y	Y
SETN	Y	Y
TESTC	Y	N
TESTN	Y	N
VAR_OUT	Y	N
VAR_TM	Y	N

A user variable persists from its creation by a procedure to the end of the File Manager invocation. This allows any number of procedures run within the same File Manager invocation to refer to the same variables.

Table 15. System character variables

Name	Description
ZINREC	Input record
ZOUTREC	Output record
ZMEMBERI	Input member name
ZMEMBERO	Output member name
ZDSNIN	Input data set name
ZDSNOUT	Output data set name

Table 16. System numeric variables

Name	Description
ZRECSIN	Input record count
ZRECSOUT	Output record count

Tally register for external REXX functions

Table 17 shows the functions that support a tally register that allows you to report on the function activity.

Table 17. Functions supporting a TALLY register

Function name	Counts number of	Sample coding for tally literal
CHANGE	Strings changed	(fld(1,'a','c',0,,,'Change 'a' to 'c' '))
CHG_OUT	Strings changed	chg_out('a','c',0,,,,'Change 'a' to 'c' ')
CHG_VAR	Strings changed	chg_var(myvar,'a','c',0,,,,'Change 'a' to 'c' ')
CONTAINS	True results	co(fld(1,2),'aa','bb','cc',,'Contains 'aa','bb','cc''')
FLD_CO	True results	fld_co(1,2,c,'aa','bb','cc',,'Contains 'aa','bb','cc''')
FLD_OUT	Invocations	fld_out(1,2,3,2,,,'Move Columns 1,2 to Columns 3,4 ')
FLD_TM	True results	fld_tm(1,'01'x,,,'Test under mask column 1 for '01'x ')
FLD_TYPE	True results	fld_type(36,1,Z,'Check Column 36 for valid zoned')
FLDI	True results	fldi(1,4,b,'>',64,'People over 64')
FLDO	True results	fldo(1,4,b,'>',64,'People over 64')
NCONTAIN	True results	nco(fld(36,1),1,4,3,2,,,'Column 36 contains 1,4,3,2''')
OFLD_CO	True results	ofld_co(1,2,c,'aa','bb','cc',,'Output contains 'aa','bb','cc''')
OVLY_OUT	Invocations	ovly_out('**',1,2,,,'Overlay columns 1,2 with '**' ')
OVLY_VAR	Invocations	ovly_var(myvar,'**',1,2,,,'Overlay columns 1,2 with '**' ')
SET_OLEN	Invocations	Set_olen(84,'b',,'Change output record length to 84')
SETC	True results	setc(myvar,'abc',,'Set myvar to abc')
SETN	True results	setn(mynum,'+2',,'Add 2 to mynum')
TESTC	True results	testc(myname,'cu','Smith','Jones',,'Common surnames')
TESTN	True results	testn(varage,'>',64,'People over 64')
TFLD	True results	tfld('Age','>',64,'People over 64') tfld('Age','NN','Non-Numeric Age fields') tfld('Age','RG',21,75,'People between 21 and 75 ') tfld('Name','CU','Smith','Jones',,'Common surnames')
TM	True results	tm(fld(1,1),'01'x,,,'Test under mask column 1 for '01'x ')
VAR_OUT	Invocations	var_out(myvar,1,2,3,2,,,'Move Columns 1,2 to Columns 3,4 ')
VAR_TM	True results	var_tm(myvar,1,'01'x,,,'Test under mask column 1 for '01'x ')

Specifying your tally register

The tally register is defined when you provide a literal value as an additional operand to the functions in Table 17. For functions that have a fixed number of operands, the literal operand is the next positional operand beyond the defined operands for the given function. For functions that have an indefinite number of operands, a null operand is required to delimit the function operands and to denote the next operand as a tally literal.

Function reference: External REXX functions

Example 1

```
IF FLD_CO(1,8,c,'a',,'Number of records with "a"') then
  chg_out('a','c',0,,,'Number of strings changed from "a" to "c"')
```

produces this tally report:

```
TALLY summary report
-----
Number of records with "a"                4
Number of strings changed from "a" to "c" 32
```

Note: FLD_CO can have an indefinite number of search literals. As a result, the tally register is specified by ,,'Number of records with "a"'.

For CHG_OUT, no null positional delimiter is required and the TALLY literal must be the seventh operand.

Example 2

```
*FASTREXX
if fld_tm(1,'01'x) then do;
  OVLY_OUT('**',1,2,,,'Count of first 2 chars set to "**"')
  return
end;
```

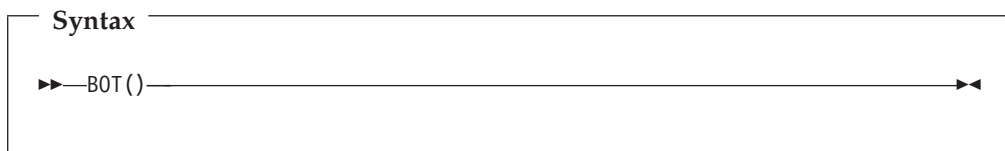
produces this tally report:

```
TALLY summary report
-----
Count of first 2 chars set to "**"        4
```

Using a tally register

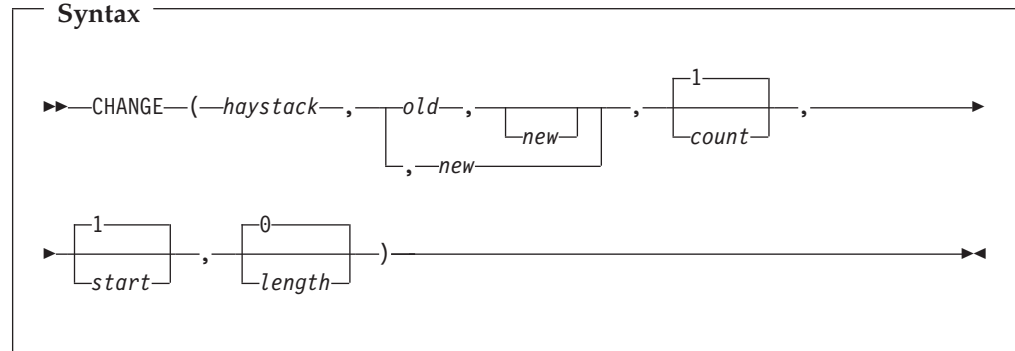
Each tally register is defined by the literal description. If you code the same literal description in a number of functions, then the same tally registered is incremented as determined by the function being invoked.

BOT (DSEB only)



Moves to the last input record.

CHANGE



Note: Commas following the last specified argument can be omitted.

Searches *haystack* and changes one or more occurrences of *old* to *new*.

Returns

Returns *haystack*, with up to *count* occurrences of the substring *old* changed to the substring *new*.

haystack

The string that you want to search.

old

Old string to change. If this argument is omitted, the new string is inserted at the *start* location.

new

New string. If this argument is omitted, then *count* occurrences of *old* are deleted.

count

Maximum number of occurrences of *old* to change. Must be a non-negative integer. Default value is 1. A value of 0 indicates that all occurrences should be changed, unless the *old* string field is omitted, in which case it is equivalent to a value of 1.

start

Position in *haystack* in bytes at which to start searching for occurrences of *old*. Must be a positive integer. The default value is 1. If *start* is greater than the current length of the output record, the function has no effect.

length

Number of bytes within *haystack* to search for occurrences of *old*. Must be a non-negative integer. A value of 0 indicates that the remainder of *haystack* from *start* should be searched. If *length* is less than the length of *old*, the function has no effect.

Example 1

```
CHANGE('abcabcabc','abc','DeF')      →  'DeFabcbc'
/* 1 (default) occurrence of old changed */
```

Example 2

```
CH('abcabcabc','abc','DeF',2)       →  'DeFDeFabc'
/* 2 occurrences of old changed */
```

Example 3

```
CHANGE('abcabcabc','abc','DeF',0)    →  'DeFDeFDeF'
/* count = 0, all occurrences of old changed */
```

Example 4

Function reference: External REXX functions

```
CH('abcabcabc','abc','DeF',,4)      → 'abcDeFabc'
/* 1 (default) occurrences of old changed, */
/* starting at position 4                */
```

Example 5

```
CHANGE('aaaaaaaa','a','A',0,3,2)    → 'aaAAaaaa'
/* all occurrences of old changed, starting at */
/* position 3 for a length of 2                */
```

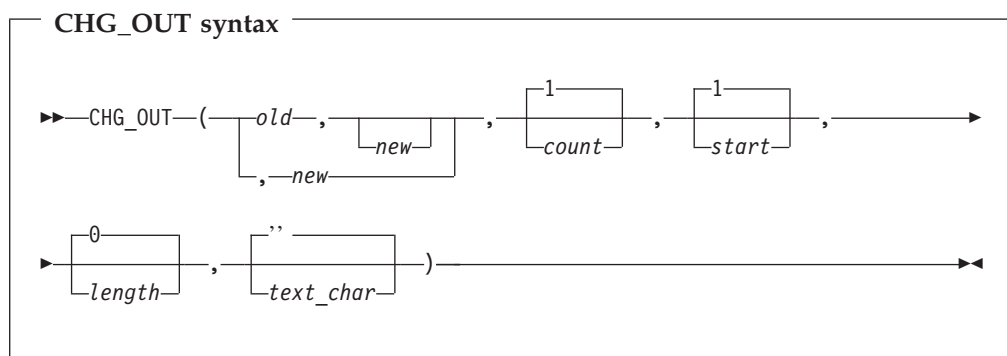
Example 6

```
CH('abcabcabc','a',,0)              → 'bcabc'
/* new omitted, count = 0,                  */
/* all occurrences of old deleted            */
```

Example 7

```
CHANGE('abc',,'def',,2)              → 'defbc'
/* old omitted, new inserted, starting at */
/* position 2                               */
```

CHG_OUT



Note: Commas following the last specified argument can be omitted.

Can be used in FASTREXX procedures.

Changes one or more occurrences of an *old* string in the output record to a *new* string. On successful execution, also updates the value of OUTPOS to one byte past the end of the last changed field in the output record.

Returns

A single blank.

old Old string to change. If this argument is omitted, the new string is inserted at the *start* location.

You can substitute a character or numeric variable or tally literal by specifying an *&varname* where *varname* matches an existing variable name.

Notes:

1. Numeric values are converted to display form with leading zeros removed.
2. If a variable name is not found, then the string is interpreted as literal.

new New string. If this argument is omitted, then *count* occurrences of *old* are deleted.

You can substitute a character or numeric variable or tally literal by specifying an *&varname* where *varname* matches an existing variable name.

Notes:

1. Numeric values are converted to display form with leading zeros removed.
2. If a variable name is not found, then the string is interpreted as literal.

count Maximum number of occurrences of *old* to change. Must be a non-negative integer. Default value is 1. A value of 0 indicates that all occurrences should be changed, unless the *old* string field is omitted, in which case it is equivalent to a value of 1.

start Position, in bytes, in the output record at which to start searching for occurrences of *old*. Can be specified as:

Absolute position

Must be a positive integer. Default value is 1. If *start* is greater than the current length of the output record, the function has no effect.

Relative to current INPOS

Must be specified as IP*x* or IN*x*. If this resolves to a value of less than or equal to zero, the function results in an error. If this resolves to a value that is greater than the current length of the output record, the function has no effect.

Relative to current OUTPOS

Can be specified as OP*x* or ON*x*, or as P*x* or N*x*. If this resolves to a value of less than or equal to zero, the function results in an error. If this resolves to a value that is greater than the current length of the output record, the function has no effect.

length Amount, in bytes, of the output record to search for occurrences of *old*. Must be a non-negative integer. Default value is 0. A value of 0 indicates that the remainder of the output from *start* should be searched. If *length* is less than the length of *old*, the function has no effect.

text_char

Can be a null string or a single character.

Specifying a null string (the default), indicates CHG_OUT should behave without text sensitivity.

Specifying a single character defines the special text-sensitive character and indicates that text-sensitive change behaviour is required. When a character has been specified for *text_char*, CHG_OUT behaves as follows:

- If the new and old strings are the same length, CHG_OUT behaves as if text-sensitive change behaviour had not been requested.
- If the new string is shorter than the old string, then when a replacement is made, CHG_OUT searches for the first *text_char* character following the end of the replaced string. Note that the entire record is searched. If the *text_char* character is found, additional *text_char* characters are inserted at the point of this first subsequent character, to make up the difference in length between new and old. If the *text_char* character is not found in the remainder of the record, then no insertion takes place and the record is reduced in length. If the record is fixed-length and no subsequent action occurs to make up the shortfall, then the File Manager record padding process fills out the record when it is written.

Function reference: External REXX functions

The intended effect is that text on multiple lines, if aligned in columns separated by the text char character, continues to align in columns after replacement. This is useful for updating files with sequence numbers on the right, such as COBOL or JCL.

- When the new string is longer than the old string, then when a replacement is made, CHG_OUT searches to the right of the replaced string for two consecutive *text_char* characters. Note that the entire record is searched. If two consecutive *text_char* characters are found, they are replaced with a single *text_char* character. This process is repeated, starting from the remaining single character (and including that character) until the length difference between old and new is accounted for. Using this algorithm, multiple *text_char* characters can be reduced to a single *text_char* character, but a single *text_char* character between other characters is never eliminated.

The intended effect is to try to use existing “blank” areas in the string, to leave text on the right unchanged as much as possible. There is no guarantee that there will be an adequate number of *text_char* characters to accomplish the goal. If there are not enough *text_char* characters, then the rest of the record is shifted right and possibly truncated when it is written, if fixed in length. This is useful for modifying files like COBOL or JCL source, where a sequence number may exist to the right.

- If you have specified multiple string replacements (*count* is greater than 1), then the intent remains the same. The search proceeds from left to right, firstly checking for the search argument, and secondly *text_char* characters to expand or collapse. If the search argument is found, it is replaced and the search continues immediately following the replaced string.
- Note that the string replacement may be limited to byte positions by the *length* argument. However, the search for *text char* characters to add or remove continues past that limit to the end of the record if required.

Example 1

Assuming that the current output record contains 'abcabcabcabcabcabc', then:

```
CHG_OUT('abc','DeF',0)
/* All occurrences of old within the          */
/* output record are changed                   */
```

The output record becomes 'DeFDeFDeFDeFDeFDeFDeFDeF'.

Example 2

Assuming that the current output record contains 'abcabcabcabcabcabc', then:

```
CHG_OUT('abc','DeF',,4)
/* 1 (default) occurrences of old changed,    */
/* starting at position 4 within the output record */
```

The output record becomes 'abcDeFabcabcabcabc'.

Example 3

Assuming that the current output record contains 'aaaaaaaaa', then:

```
CHG_OUT('a','A',0,3,2)
/* all occurrences of old changed, starting at */
/* position 3 in the output record, for a length of 2 */
```

The output record becomes 'aaAaaaaaa'.

Example 4

Assuming that the current output record contains 'abcabcabcabcabcabc', and that INPOS is currently set to 13 and OUTPOS is currently set to 4, then:

```
CHG_OUT('abc','DeF',1,P3)
/* 1 occurrence of old changed,          */
/* uses OUTPOS as the default target,     */
/* starts at position 7 within the output record */
```

The output record becomes 'abcabcDeFabcbcabcbcabcb' and OUTPOS is set to 10 (INPOS remains unchanged).

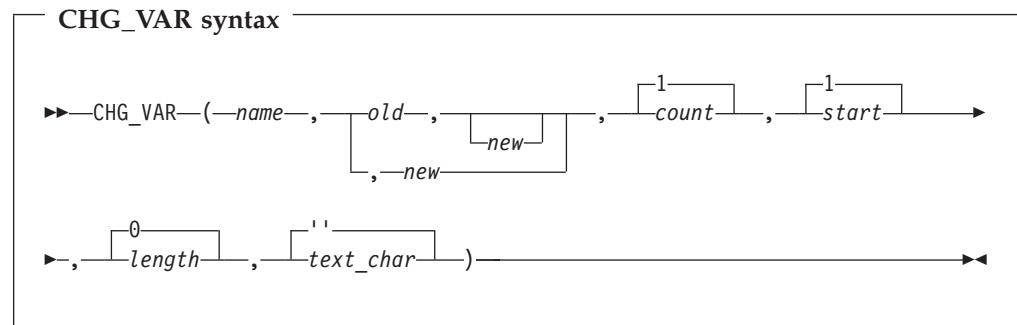
Example 5

Assuming that the current output record contains 'abcabcabcabcabcabc', and that INPOS is currently set to 13 and OUTPOS is currently set to 4, then:

```
CHG_OUT('abc','DeF',1,IN3)
/* 1 occurrence of old changed,          */
/* forces start to use INPOS value,     */
/* starts at position 10 within the output record */
```

The output record becomes 'abcabcabcDeFabcbcabcb' and OUTPOS is set to 13 (INPOS remains unchanged).

CHG_VAR



(Can be used in FASTREXX procedures.)

Note: Commas following the last specified argument can be omitted.

Changes one or more occurrences of an old string in the variable to a new string. On successful execution, also updates the value of OUTPOS to one byte past the end of the last changed field in the variable.

Returns

A single blank.

name 1–256 character variable identifier. Variable name matching is not case sensitive. If the name is not found, a severe error occurs and the procedure is terminated. Cannot be a system character variable or a system numeric variable. See “Using FASTREXX variables” on page 1068.

old Old string to change. If this argument is omitted, the new string is inserted

Function reference: External REXX functions

at the start location. You can substitute a character or numeric variable or tally literal by specifying an *&varname* where *varname* matches an existing variable name.

Notes:

1. Numeric values are converted to display form with leading zeros removed.
2. If a variable name is not found, then the string is interpreted as a literal.

new New string. If this argument is omitted, then *count* occurrences of *old* are deleted. You can substitute a character or numeric variable or tally literal by specifying an *&varname* where *varname* matches an existing variable name.

Notes:

1. Numeric values are converted to display form with leading zeros removed.
2. If a variable name is not found, then the string is interpreted as a literal.

count Maximum number of occurrences of *old* to change. Must be a non-negative integer. Default value is 1. A value of 0 indicates that all occurrences should be changed.

start Position, in bytes, in the variable at which to start searching for occurrences of *old*. Can be specified as:

Absolute position

Must be a positive integer. Default value is 1. If *start* is greater than the current length of the variable, the function has no effect.

Relative to current variable position

Can be specified as *OPx* or *ONx*, or as *Px* or *Nx*, or as *IPx* or *INx*. If this resolves to a value of less than or equal to zero, the function results in an error. If this resolves to a value that is greater than the current length of the variable, the function has no effect.

length Amount, in bytes, of the variable to search for occurrences of *old*. Must be a non-negative integer. Default value is 0. A value of 0 indicates that the remainder of the output from *start* should be searched. If *length* is less than the length of *old*, the function has no effect.

text_char

Can be a null string or a single character.

Specifying a null string (the default), indicates CHG_VAR should behave without text sensitivity.

Specifying a single character defines the special text-sensitive character and indicates that text-sensitive change behaviour is required. When a character has been specified for *text_char*, CHG_VAR behaves in this way:

- If the new and old strings are the same length, CHG_VAR behaves as if text-sensitive change behaviour had not been requested.
- If the new string is shorter than the old string, then when a replacement is made, CHG_VAR searches for the first *text_char* character following the end of the replaced string. Note that the entire record is searched. If the *text_char* character is found, additional *text_char* characters are inserted at the point of this first subsequent character, to make up the difference in length between *new* and *old*. If the *text_char* character is not

found in the remainder of the record, then no insertion takes place and the record is reduced in length. If the record is fixed-length and no subsequent action occurs to make up the shortfall, then the File Manager record padding process fills out the record when it is written.

The intended effect is that text on multiple lines, if aligned in columns separated by the text char character, continues to align in columns after replacement. This is useful for updating files with sequence numbers on the right, such as COBOL or JCL.

- When the new string is longer than the old string, then when a replacement is made, CHG_VAR searches to the right of the replaced string for two consecutive *text_char* characters. Note that the entire record is searched. If two consecutive *text_char* characters are found, they are replaced with a single *text_char* character. This process is repeated, starting from the remaining single character (and including that character) until the length difference between *old* and *new* is accounted for. Using this algorithm, multiple *text_char* characters can be reduced to a single *text_char* character, but a single *text_char* character between other characters is never eliminated.

The intended effect is to try to use existing "blank" areas in the string, to leave text on the right unchanged as much as possible. There is no guarantee that there will be an adequate number of *text_char* characters to accomplish the goal. If there are not enough *text_char* characters, then the rest of the record is shifted right and possibly truncated when it is written, if fixed in length. This is useful for modifying files like COBOL or JCL source, where a sequence number may exist to the right.

- If you have specified multiple string replacements (*count* is greater than 1), then the intent remains the same. The search proceeds from left to right, firstly checking for the search argument, and secondly *text_char* characters to expand or collapse. If the search argument is found, it is replaced and the search continues immediately following the replaced string.
- Note that the string replacement may be limited to byte positions by the length argument. However, the search for *text_char* characters to add or remove continues past that limit to the end of the record if required.

Example 1

Assuming that the current variable contains 'abcabcabcabcabcabc', then:

```
CHG_VAR(MYVAR,'abc','DeF',0)
/* All occurrences of old within the          */
/* variable are changed                        */
```

The variable becomes 'DeFDeFDeFDeFDeFDeF'.

Example 2

Assuming that the current variable contains 'abcabcabcabcabcabc', then:

```
CHG_VAR(MYVAR,'abc','DeF',,4)
/* 1 (default) occurrences of old changed,    */
/* starting at position 4 within the variable */
```

The variable becomes 'abcDeFabcabcabcabcab'.

Example 3

Function reference: External REXX functions

```
|           Assuming that the current variable contains 'aaaaaaaa', then:  
| CHG_VAR(MYVAR, 'a', 'A', 0, 3, 2)  
| /* all occurrences of old changed, starting at */  
| /* position 3 in the variable, for a length of 2 */
```

| The variable becomes 'aaAaaaaaa'.

Example 4

| Assuming that the current variable contains 'abcabcabcabcabcabc', and that
| current variable position is 4, then:

```
| CHG_VAR(MYVAR, 'abc', 'DeF', 1, P3)  
| /* 1 occurrence of old changed, */  
| /* uses current position as the default target, therefore */  
| /* starts at position 7 within the variable */
```

| The variable becomes 'abcabcDeFabcbcabcbcabcb' and the variable position is set to
| 10.

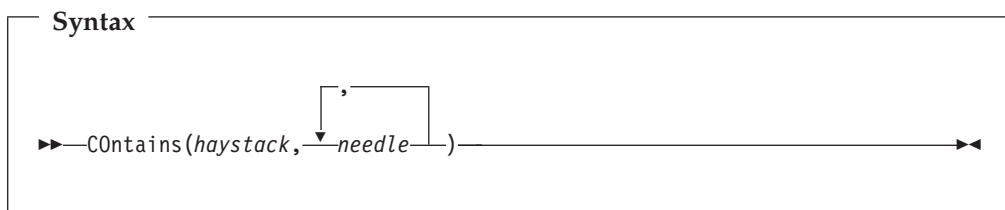
Example 4

| Assuming that the current variable contains 'abcabcabcabcabcabc', and that
| current variable position is currently set to 13, then:

```
| CHG_VAR(MYVAR, 'abc', 'DeF', 1, IN3)  
| /* 1 occurrence of old changed, */  
| /* forces start to use current position value, therefore */  
| /* starts at position 10 within the variable */
```

| The variable becomes 'abcabcabcDeFabcbcabcbcabcb' and the variable position is set to
| 13.

CONTAINS



Checks the contents of *haystack* for one or more occurrences of *needle*.

Returns

If the *haystack* string contains one or more of the *needle* strings, then CONTAINS returns 1. Otherwise, CONTAINS returns 0.

CONTAINS is case-sensitive: it only returns 1 if the *haystack* contains a string with the same mix of uppercase and lowercase as a *needle*.

haystack

The string that you want to search.

needle The string that you are attempting to find within *haystack*. You can search for up to 20 strings at a time.

For a similar function that matches numeric values, see "NCONTAIN" on page 1096. For a FASTREXX-eligible equivalent, see "FLD_CO" on page 1082.

Example 1

If the current input record contains “Michael”, “Mick” or “Mike” in the first ten columns, then print the record.

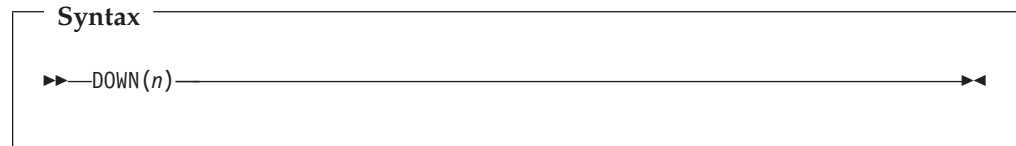
```
If CO(FLD(1,10), 'Michael', 'Mick', 'Mike') Then
  PRINT(inrec, 'CHAR')
```

Example 2

If the current input record contains “USA”, “Australia” or “England”, then drop the record from processing.

```
If CONTAINS(inrec, 'USA', 'Australia', 'England') Then
  Return 'DROP'
```

DOWN (DSEB only)

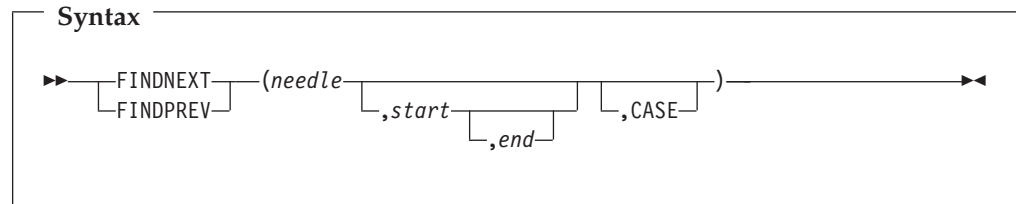


Moves down *n* number of records, or to the last record, if there are less than *n* records below the current record.

Returns

If, after moving, the current record is the last record, then the DOWN function returns the string value “EOF” (end of file). Otherwise, DOWN returns 0.

FINDNEXT, FINDPREV (DSEB only)



Searches for *needle* in the input data set, from the current input record forwards (FINDNEXT) or backwards (FINDPREV). You can limit the search to a range of columns, or to an exact matching case.

Returns

If the search is successful, the record in which *needle* was found becomes the current input record, and the FINDNEXT function returns the starting column of *needle* in the record. If the search is unsuccessful, the current input record remains the same, and FINDNEXT returns 0.

needle String(s) or numeric(s) to search for.

start The position, in bytes, of the start of the range in each input record to be searched.

Function reference: External REXX functions

end The position, in bytes, of the end of the range in each input record to be searched.

CASE Specifies that the comparison is case-sensitive.

If you want to save any changes you have made to the record that was current prior to calling FINDNEXT or FINDPREV, use the UPDATE function. Otherwise, if the search is successful, any changes made to that record are lost when FINDNEXT or FINDPREV moves to another record.

Here are some examples:

Example 1

```
FINDPREV('abc') /* Finds 'abc', 'ABC', 'ABc' and so on */
```

Example 2

```
FINDNEXT('abc',1,10,'CASE') /* Finds 'abc', but not 'ABC', 'ABc' and so on */
```

FLD

Syntax

```
►► FLD(start_column [ ,length ] [ ,type ] )
```

Can be used in FASTREXX condition expressions.

Fetches the value of a field from the current input record (INREC), starting at *start_column*, of *length* number of bytes, interpreted according to the specified *type*.

Returns

The value of the field from the current input record.

start_column

Position, in bytes, in the input record at which to start reading the field value. Can be specified as:

Absolute position

Must be a positive integer. Default value is 1. If *start* is greater than the current length of the input record, the function has no effect.

Relative to current INPOS

Can be specified as IPx or INx , or as Px or Nx . If this resolves to a value of less than or equal to zero, the function results in an error. If this resolves to a value that is greater than the current length of the input record, the function has no effect.

Relative to current OUTPOS

Must be specified as OPx or ONx . If this resolves to a value of less than or equal to zero, the function results in an error. If this resolves to a value that is greater than the current length of the input record, the function has no effect.

length The length of the field in bytes.

For binary fields, you must specify the length. It can be 2, 4, or 8.

For character fields, if you omit the length, FLD returns the remainder of the record.

For packed decimal fields, if you specify the length, it must be in the range 1–16. If you omit the length, FLD attempts to determine the packed field length from the record data and returns only that field.

For zoned decimal fields, if you specify the length, it must be in the range 1–31 or, if the field contains a separate sign character, in the range 1–32. If you omit the length, FLD returns the remainder of the record.

type The data type of the field. Valid values are:

- B** Binary. FLD interprets binary fields as being signed.
- C** Character. This is the default.
- P** Packed decimal.
- U** Interprets the field as Character, but converts it to uppercase before returning the string.
- Z** Zoned decimal. Interprets all of the COBOL external decimal variants as numeric data.

If you specify a value for *length* that would cause the record length to be exceeded:

- For character fields (*type* C, U), FLD returns the remainder of the record.
- For numeric fields (*types* B, P, Z), FLD returns a null string.

If you specify a numeric type (*types* B, P, Z), and the specified field contains invalid data for that type, then FLD returns a null string. Numeric data is always returned in integer form; that is, FLD does not perform scaling of numeric data.

The FLD function is similar to the built-in REXX SUBSTR function, except that FLD interprets the “substring” according to the specified data type, and returns the value formatted appropriately. (For a numeric field, FLD returns the value with a sign, and without leading zeros.)

Example 1

If the value of the packed decimal field that starts at column 8 is greater than 100, then do not process the current record.

```
If FLD(8,P) > 100 Then Return 'DROP'
```

Example 2

If the value of the 2-digit year field starting at column 42 is greater than 60, then insert the literal “19” before the year field; otherwise, insert “20”.

```
If FLD(42,2,Z) > 60 Then
  outrec = FLD(1,41)||'19'||FLD(42)
Else
  outrec = FLD(1,41)||'20'||FLD(42)
```

Example 3

If the 4-byte field that starts at column 11 does not contain valid packed decimal data, then do not process the current record.

```
If FLD(11,4,p) = '' Then Return 'DROP'
```

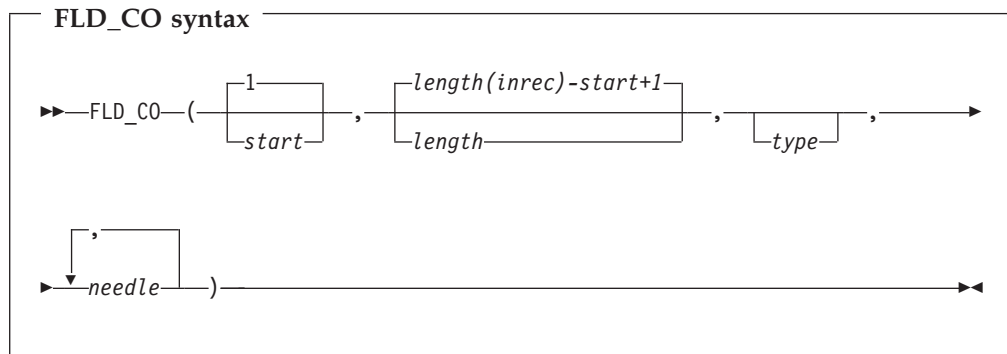
Function reference: External REXX functions

Example 4

If the value of the packed decimal field that starts at INPOS + 8 is greater than 100, then do not process the current record.

```
If FLD(P8,P) > 100 Then Return 'DROP'
```

FLD_CO



(Can be used in FASTREXX condition expressions.)

Searches the field within the input record specified by *start* and *length*, for one or more occurrences of *needle*. On successful execution when searching for a string, also updates the value of INPOS to the first byte of the located field in the input record.

Returns

If at least one occurrence of *needle* is found, returns 1. If no occurrences are found, returns 0.

start Position, in bytes, in the input record at which to start searching for occurrences of *needle*. Can be specified as:

Absolute position

Must be a positive integer. Default value is 1. If *start* is greater than the current length of the input record, the function has no effect.

Relative to current INPOS

Can be specified as IP*x* or IN*x*, or as P*x* or N*x*. If this resolves to a value of less than or equal to zero, the function results in an error. If this resolves to a value that is greater than the current length of the input record, the function has no effect.

Relative to current OUTPOS

Must be specified as OP*x* or ON*x*. If this resolves to a value of less than or equal to zero, the function results in an error. If this resolves to a value that is greater than the current length of the input record, the function has no effect.

length Length, in bytes, of the field to be searched.

- For character fields, the length defaults to the remaining record length from the start position to the end of the record (inclusive). A value of 0 also indicates that the field extends to the end of the record.
- For binary fields, you must specify the length. It can be 2, 4, or 8.

- For packed decimal fields, if you specify the length, it must be in the range 1-16. If you omit the length, FLD_CO attempts to determine the packed field length from the record data.
- For zoned decimal fields, if you specify the length, it must be in the range 1-31 or, if the field contains a separate sign character, in the range 1-32. If you omit the length, it defaults to the remainder of the record.

type The data type of the field. Valid values are:

- B** Binary. FLD_CO interprets binary fields as being signed.
- C** Character. This is the default. The comparison is case sensitive.
- P** Packed decimal.
- U** Interprets the field as Character, but converts it to uppercase before comparing it with *needle*.
- Z** Zoned decimal. Interprets all of the COBOL external decimal variants as numeric data.

needle String(s) or numeric(s) to search for. For the character types, FLD_CO searches the *haystack* for each of the *needles*. In this context, it behaves like a combination of the FLD and CONTAINS functions. For the numeric data types, the *haystack* is treated as a single numeric field, and an appropriate numeric comparison is performed against each of the *needles*. In this context it behaves like a combination of the FLD and NCONTAIN functions. You can search for up to 20 strings at a time.

To perform case-insensitive searches, specify *type* as 'U' and *needle* in uppercase.

Example 1

If the current input record contains "MIKE", "Mike" or "mike" in the first ten columns, then write the record.

```
If FLD_CO(1,10,'U','MIKE') Then WRITE('MDD')
```

Example 2

If the current input record contains "USA", "Australia" or "England", then drop the record from processing.

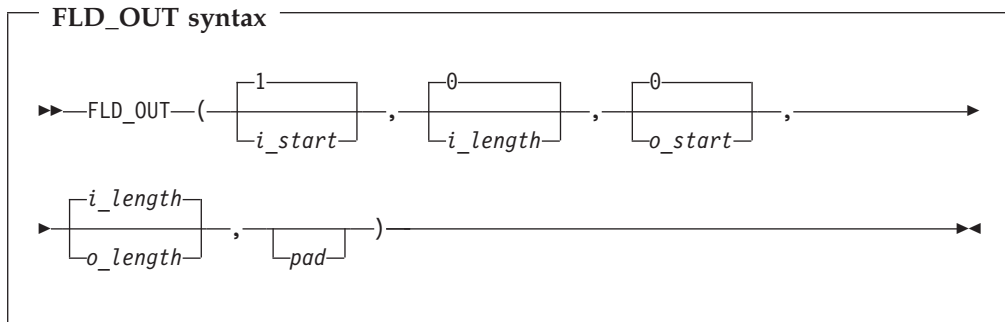
```
If FLD_CO(,,'USA','Australia','England') Then Return 'DROP'
```

Example 3

If the current input record contains "MIKE", "Mike" or "mike" in the ten columns immediately after the current INPOS, then write the record.

```
If FLD_CO(P0,10,'U','MIKE') Then WRITE('MDD')
```

FLD_OUT



Note: Commas following the last specified argument can be omitted.

Can be used in FASTREXX procedures.

Overlays the output record with a field from the input record. See OVLY_OUT for a function to overlay the output record with a literal. If the target field length exceeds the source field length, then the source field is padded to the specified length using the pad character. If the target field length is less than the source field, the source field is truncated from the right. On successful execution, also updates the value of OUTPOS to one byte past the end of the field overlaid in the output record.

Returns

A single blank

i_start Position, in bytes, in the input record at which to start reading the field to be copied. Can be specified as:

Absolute position

Must be a positive integer. Default value is 1.

Relative to current INPOS

Can be specified as IP*x* or IN*x*, or as P*x* or N*x*. Must resolve to a positive integer.

Relative to current OUTPOS

Must be specified as OP*x* or ON*x*. Must resolve to a positive integer.

i_length

Length, in bytes, of the source field. Must be a non-negative integer. Defaults to 0. If you omit *i_length* or specify zero, the remainder of the input record from the *i_start* position is used. This also applies if you specify a value that would cause the source field to be read from beyond the end of the current input record.

o_start

Position, in bytes, in the output record at which to start overlaying the copied field. If you omit *o_start* or specify zero, the field is appended to the end of the output record. If *o_start* is greater than the current length of the output record, the record is padded with the specified or defaulted pad character from the current record length to the specified start position. Can be specified as:

Absolute position

Must be a non-negative integer. Default value is 0.

Relative to current INPOS

Must be specified as IP*x* or IN*x*. Must resolve to a positive integer.

Relative to current OUTPOS

Can be specified as OP*x* or ON*x*, or as P*x* or N*x*. Must resolve to a positive integer.

o_length

Length, in bytes, of the target field. Defaults to the source field length (*i_length*). A value of 0 indicates that the target field length is the greater of *i_length* and the remaining output record length. If 0 is specified for both *o_start* and *o_length*, then *i_length* is used as the target length.

pad

Pad character. Defaults to the pad character set on the File Manager System Processing Options panel (when processing online) or the pad character specified in the SET function (when running in batch). If the current pad setting is OFF or unspecified, the default pad character is a blank.

Example 1

Copy the characters in columns one and two of the input record to columns three and four of the output record.

```
FLD_OUT(1,2,3,2)
```

Example 2

Append the characters in columns eleven and twelve of the input record to the end of the output record, padded with two blanks.

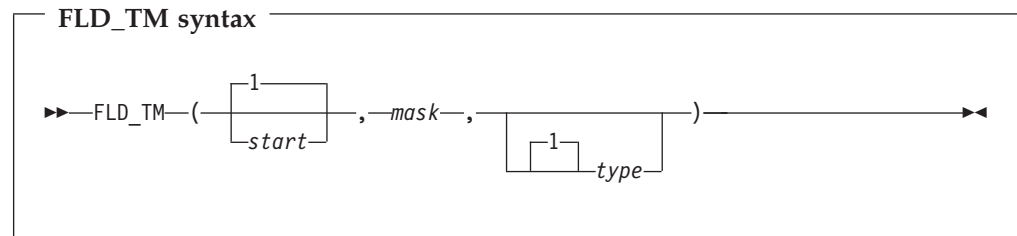
```
FLD_OUT(11,2,0,4, '  ')
```

Example 3

Search a field in the input record for the characters 'AA' and, if found, copy to the end of the output record (assumes that OUTPOS is still set to end of output record).

```
IF FLD_CO(10,2,C,'AA') Then
/* when successful, updates INPOS to 12 */
FLD_OUT(N2,2,P0,2)
/* copies from INPOS - 2, appends to end of output record */
```

FLD_TM



Note: Commas following the last specified argument can be omitted.

Can be used in FASTREXX condition expressions.

Tests selected bits of a field in the input record.

Function reference: External REXX functions

Returns

Returns 1 if the test evaluates as True, and 0 if the test evaluates as False.

start Position, in bytes, in the input record at which to start testing. The length of the field is defined by the *mask*. Can be specified as:

Absolute position

Must be a positive integer. Default value is 1. If *start* is greater than the current length of the input record, the function has no effect.

Relative to current INPOS

Can be specified as IP*x* or IN*x*, or as P*x* or N*x*. If this resolves to a value of less than or equal to zero, the function results in an error. If this resolves to a value that is greater than the current length of the input record, the function has no effect.

Relative to current OUTPOS

Must be specified as OP*x* or ON*x*. If this resolves to a value of less than or equal to zero, the function results in an error. If this resolves to a value that is greater than the current length of the input record, the function has no effect.

mask Bit-string determining which bits to test in the field. The length of the mask defines the length of the input field. This field defines a bit-string mapping used to test the specified bits in the input record. You can use the bit-string, hex-string or character-string formats to define this field. Therefore, '0100 0000'b, '40'x, and '' are all legitimate and equivalent ways of defining a mask to test the second bit of a one-byte field.

type Type of test.

- 1** FLD_TM returns True (1) if **all** the bits that are **on** in the mask are **on** in the input record field. This is the default value.
- 0** FLD_TM returns True (1) if **all** the bits that are **on** in the mask are **off** in the input record field.
- M** FLD_TM returns True (1) if **at least one** of the bits that are **on** in the mask is **on** in the input record, and **at least one** is **off**.
- N** FLD_TM returns True (1) if **at least one** of the bits that are **on** in the mask is **off** in the input record field.

Example 1

Test the third byte of the input record and, if the low order bit is set, overlay a hex FF into the second byte of the output record.

```
If FLD_TM(3,'01'x) Then Do
  OVLY_OUT('ff'x,2,1)
Return
End
Return 'DROP'
```

Example 2

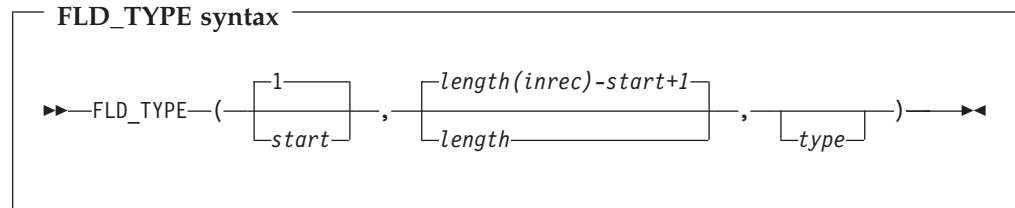
Test the third byte of the input record and if some of the three high order bits are set, and some are not, overlay the contents of the second byte of that record with a hex 04.

```
If FLD_TM(3,'11100000'b,M) Then Do
  OVLY_OUT('04'x,2,1)
```

Example 3

Test the current INPOS position of the input record and, if the low order bit is set, overlay a hex FF into the byte prior to this location in the output record.

```
If FLD_TM(P0, '01'x) Then Do
  OVLY_OUT('ff'x, IN1, 1)
Return
End
Return 'DROP'
```

FLD_TYPE

Note: Commas following the last specified argument can be omitted.

Can be used in FASTREXX condition expressions.

Tests the data type of a field in the input record.

Returns

Returns 1 if the test evaluates as True, and 0 if the test evaluates as False.

start Position, in bytes, in the input record at which to start testing. Can be specified as:

Absolute position

Must be a positive integer. Default value is 1. If *start* is greater than the current length of the input record, the function has no effect.

Relative to current INPOS

Can be specified as *IPx* or *INx*, or as *Px* or *Nx*. If this resolves to a value of less than or equal to zero, the function results in an error. If this resolves to a value that is greater than the current length of the input record, the function has no effect.

Relative to current OUTPOS

Must be specified as *OPx* or *ONx*. If this resolves to a value of less than or equal to zero, the function results in an error. If this resolves to a value that is greater than the current length of the input record, the function has no effect.

length Length of the field in the input record.

- For packed decimal fields, if you specify the length, it must be in the range 1-16. If you omit the length, FLD_CO attempts to determine the packed field length from the record data.
- For zoned decimal fields, if you specify the length, it must be in the range 1-31 or, if the field contains a separate sign character, in the range 1-32. If you omit the length, it defaults to the remainder of the record. A value of 0 also indicates that the field extends to the end of the record.

type Data type to test for.

Function reference: External REXX functions

- P** FLD_TYPE returns 1 if the field is a valid packed decimal field. Variant sign values (such as 'f'x for positive) are considered valid. Returns 0 otherwise.
- Z** FLD_TYPE returns 1 if the field is a valid zoned decimal field. FLD_TYPE recognizes all of the COBOL external decimal variants as numeric data. Returns 0 otherwise.

Example 1

If the first three bytes of the current input record contain a valid packed decimal number, tally the field. Otherwise, tally the first two bytes as a binary number.

```
If FLD_TYPE(1,3,P) Then
  TALLY(1,3,P,'Tally packed')
Else
  TALLY(1,2,B,'Tally binary')
```

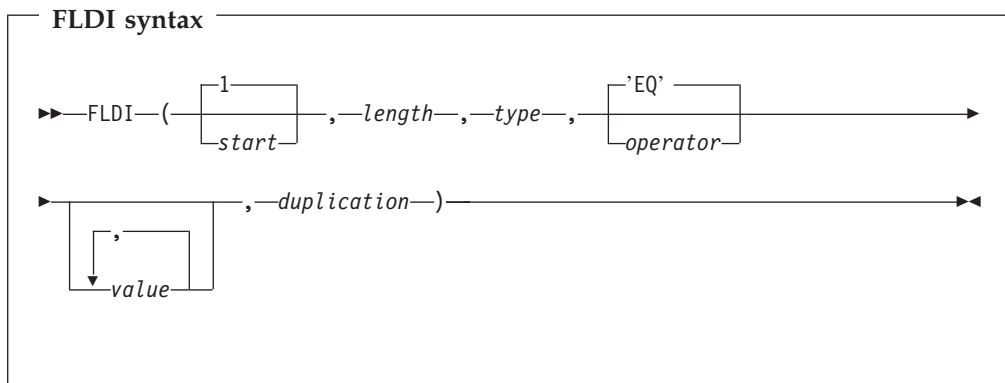
Example 2

If the three bytes starting at the current INPOS in the input record contain a valid packed decimal number, copy the three bytes to the end of the output record. Otherwise, copy the two bytes starting at INPOS to the end of the output record.

```
If FLD_TYPE(P0,3,P) Then
  FLD_OUT(P0,3,P0,3)
Else
  FLD_OUT(P0,2,P0,2)
```

Note: In this example, the abbreviated form of the relative position specification can be used in both arguments of the FLD_OUT function. This is because *i_start* naturally targets the input record and *o_start* naturally targets the output record.

FLDI



Can be used in FASTREXX condition expressions.

Performs a conditional test against input record field defined in by the start length and type parameters.

Notes:

1. The operator (operator), and non-numeric values should all be enclosed in quotes to avoid syntax errors.

2. If you specify a value for *length* that would cause the record length to be exceeded, a false result is returned.
3. If you specify a numeric type (types B, P, Z), and the specified field contains invalid data for that type, then the function returns a false result. Numeric data is always returned in integer form; that is, the function does not perform scaling of numeric data.

start Position in bytes, in the input record at which to start reading the field value. Can be specified as:

Absolute position

Must be a positive integer. If *start* is greater than the current length of the input record, the function has no effect.

Relative to current INPOS

Can be specified as IPx or INx, or as Px or Nx. If this resolves to a value of less than or equal to zero, the function results in an error. If this resolves to a value that is greater than the current length of the input record, the function produces a false result.

Relative to current OUTPOS

Must be specified as OPx or ONx. If this resolves to a value of less than or equal to zero, the function results in an error. If this resolves to a value that is greater than the current length of the input record, the function produces a false result.

length The length of the field in bytes.

- For binary fields, you must specify the length. It can be 2, 4, or 8.
- For character fields, if you omit the length, the length is defaulted:
 - For contains type operators, to the rest of the input record.
 - When all values are variable substitutions, to the rest of the input record.
 - Otherwise, the maximum literal value length is be used.
- For packed decimal fields, if you specify the length, it must be in the range 1–16. If you omit the length, the function attempts to determine the packed field length from the record data and returns only that field.
- For zoned decimal fields, if you specify the length, it must be in the range 1–31 or, if the field contains a separate sign character, in the range 1–32. If you omit the length, the function returns the remainder of the record. If this exceeds 32, then the function returns a false result.

type The data type of the field. Valid values are:

- B** Binary. The function interprets binary fields as being signed.
- C** Character. This is the default.
- P** Packed decimal.
- U** Interprets the field as character, but converts it to uppercase before returning the string.
- Z** Zoned decimal. Interprets all of the COBOL external decimal variants as numeric data.

operator

The default is EQ or =. This function supports all the operators described for dynamic template and criteria edit. For details about the operators supported and their description, see:

- “Dynamic Template panel” on page 510.

Function reference: External REXX functions

- “Record Identification Criteria panel” on page 602.
- “Record Selection Criteria panel” on page 607.

value The value or values entered must be valid in the context of the operator and the field which is being referenced. For example, only certain operators like CO (contains) allow multiple values. Numeric values should be entered when testing numeric fields, and so on.

- Specifying hexadecimal strings. A hexadecimal string must be in the form 'hhhhh'x. The value enclosed in quotes must be an even number of characters and contain valid hexadecimal characters (0–9, A–F).
- Specifying binary strings. A binary string must be in the form 'nnnnn'b. The value enclosed in quotes must be a combination of "0"s and "1"s.
- Specifying character strings. For non-numeric types, the value should be enclosed in quotes.
- Specify a variable by specifying &variable_name. A variable is substituted for the value if a matching character, numeric, or tally variable can be located. If a matching variable cannot be found, the string is treated as a literal value. If a numeric comparison is being performed, a character variable is converted to a number - if the conversion fails, the function returns a false result. If a numeric or tally variable is referenced in a character comparison, then the value is the number converted to its display form with leading zeros removed.

duplication

Specify an integer *n* to duplicate the literal value *n* times.

Note: This can only be used for operators that support a single value (for example, Not contains) and where the value is a literal constant and not a substitute variable.

Example 1

Check the input record and process only those records that contain values of 'Smith' or 'Jones'.

Note: In this case, use operator CU so the contains processing is not case-sensitive.

```
if FLDI(1,,C,'CU','Smith','Jones') then
  return
else
  return 'DROP'
```

Example 2

Process all records with a salary greater than 75000, where salary is a packed decimal value found at start position 28.

Note: In this case, allow File Manager to calculate the packed decimal field length

```
if FLDI(28,,P,'>',75000) then
  return
else
  return 'DROP'
```

Example 3

Process all input records with the value 'ABCABCABCABC' at start position 10.

Note: The length defaults to 15, the literal value length.

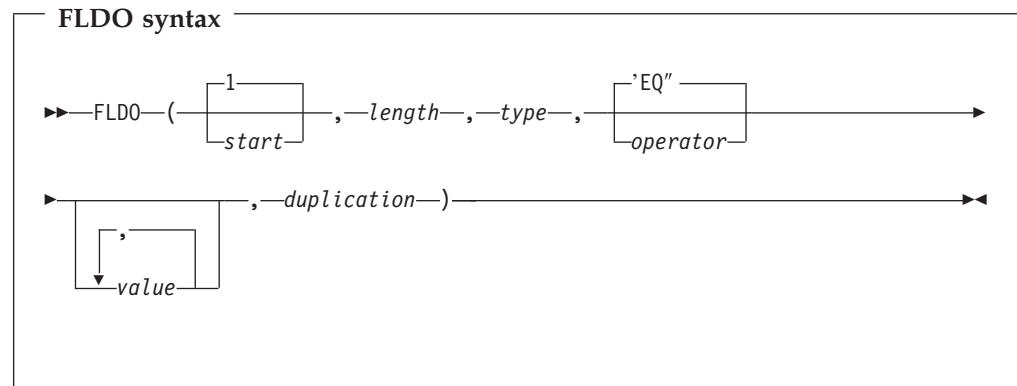
```
If FLDI(10,'=','ABC',5) then
    return
else
    return 'DROP'
```

Example 4

The same as Example 2, but using a numeric variable.

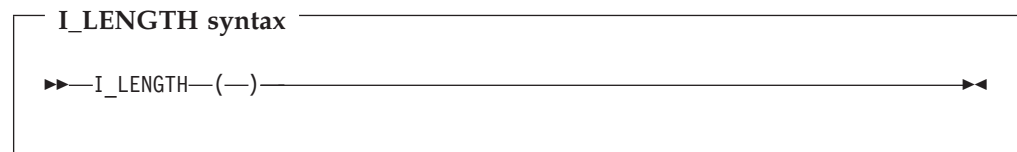
```
SETN(salary_high,75000)
if FLDI(28,,P,'>','&salary_high') then
    return
else
```

FLDO



FLDO is the same as FLDI except it tests the current output record.

I_LENGTH



Can be used in FASTREXX condition expressions.

Returns the length, in bytes, of the input record.

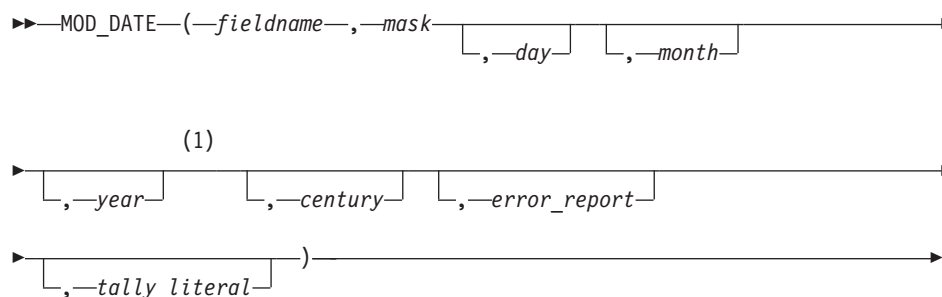
Example 1

Write only records with a length of 100 to the output file DD100.

```
If i_length() = 100 Then
    WRITE('DD100')
```

MOD_DATE

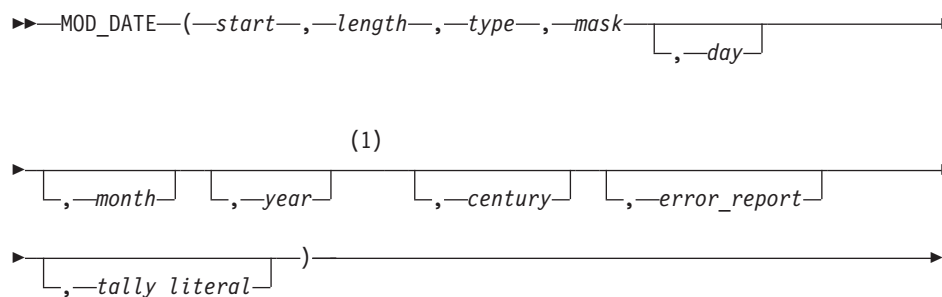
MOD_DATE syntax - with template



Notes:

- 1 At least one of these parameters (*day, month, year*) must be specified.

MOD_DATE syntax - without template



Notes:

- 1 At least one of these parameters (*day, month, year*) must be specified.

(Can be used in FASTREXX procedures.)

Notes:

1. Only modifies valid dates. Input and modified dates must range between and include 15 October 1582 to 31 December 9999.
2. Commas following the last specified argument can be omitted.

MOD_DATE can be run with or without a template and enables you to set, increment, or decrement a date field using year.month and day values. The *fieldname* or location parameters (*start, length, type*), the *mask* parameter and at least one of the *year* or *month* or *day* values must be specified for the function call to be valid. The input value is taken from the current output buffer and the modified value is stored in the output buffer.

fieldname

A template is required with this form and the name you specify must

match a field name in the template. For non-unique names, you can specify a name in the form *groupname.dataname*. Name matching is not case-sensitive. If the name is unqualified, then the first occurrence of the name is used. For dimensioned fields, you can refer to an individual array element by providing a suffix subscript in the form (*nm*), where *nm* is a valid subscript for the dimensioned field. If you do not provide a subscript, the function applies to all elements of the array.

Notes:

1. If you are running a copy process and you have specified an input and output template, then the field name must be defined to both input and output templates.
2. If you are copying multiple record layouts, the MOD_DATE function applies to records that have been identified as the record layout which contains this field.

start Position, in bytes, in the input record at which to start reading the field value. Can be specified as:

Absolute position

Must be a positive integer. Default value is 1. If *start* is greater than the current length of the input record, the function has no effect.

Relative to current INPOS

Can be specified as IPx or INx, or as Px or Nx. If this resolves to a value of less than or equal to zero or greater than the current length of the input record, the function has no effect.

Relative to current OUTPOS

Must be specified as OPx or ONx. If this resolves to a value of less than or equal to zero or greater than the current length of the input record, the function has no effect.

length The length of the field in bytes.

For binary fields, either omit the length or specify 4.

For packed decimal fields, if you specify the length, it must be in the range 1–16. If you omit the length, MOD_DATE attempts to determine the packed field length from the record data and returns only that field.

type The data type of the field. Valid values are:

- B** Binary.
- C** Character. This is the default.
- P** Packed decimal.

mask Picture string describing the date. File Manager supports all pictures described in the *z/OS Language Environment Programming Services*, Appendix B, "Date and time services tables". In addition to those masks, File Manager supports CYYDDD to handle old century value Julian dates. A "C" value of "0" is interpreted as "19", and a "C" value of "1" is interpreted as "20". By default, 2-digit years lie within the 100-year range starting 80 years prior to the system date covered by the LE masks. The default range is changed by specifying the *century* parameter.

day Day adjustment.

month Month adjustment.

year Year adjustment.

Function reference: External REXX functions

The *day*, *month*, and *year* adjustment values can set, increment, and or decrement the respective value in the date field. The presence of a plus or minus symbol indicates the number provided is to *increment* or *decrement* the current value. The absence of plus or minus symbols indicates the number is to *replace* the current value. If the first non-blank character is an asterisk, then the current day, month, or year value is used to set the current value, otherwise it is derived from the record. For the *month* value or *year* value, you can suffix the value with an "E" to indicate end of month adjustment. If the input date is the end of month, then the resultant date is the end of month, unless day arithmetic is also performed.

Here are examples showing how to code them:

'**+10' Sets the current day, month, or year and adds 10 to it.
'*-10' Sets the current day, month, or year and subtracts 10 from it.
'+10' Adds 10 to the current value found on the record.
'-10' Subtracts 10 from the current value found on the record.
'+3E' Adds 3 months and make adjustments for end of month if required.
'22' Sets the respective day, month, or year value to 22.

Note: If you specify the symbols "**", "+", or "-", you must code the value in quotes.

Month and Year Arithmetic:

If input date is the last day of the month and you specify "E" as the suffix on the month or year value, or if the resulting month has fewer days than the day component of the input date, the result is the last day of the resulting month. Otherwise, the result has the same day component as input date.

Here are some examples:

- Assume today is January 31, 2007. Adding one month results in the end of February, 2007-02-28.
- "+3E" is specified for *month*, and the input date is February 28th 2007. The result is May 31st 2007.
- "+3" is specified for *month*, and the input date is February 28th 2007. The result is May 28th 2007.

century

Specify a value between 0 and 100 to be used to define the 100-year range to interpret a 2-digit year. The default value is 80 years prior to the system date. Use this parameter to adjust that value.

error_report

Specify "Y" if you want to produce error messages when the function cannot modify a date because either the input or resultant date is invalid. The default is to ignore such dates.

tally_literal

Specify a literal to appear on a tally report that counts every successful operation of the function.

Examples

This COBOL copybook describes an input record with different date formats and is used here to illustrate MOD_DATE usage with a template or copybook.

```
01 DATE-REC.  
   03 DATE-YYYYMMDD          pic x(8).  
   03 DATE-MM-DD-YYYY        pic x(10).
```

```

|           03 DATE-DD-MM-YYYY           pic x(10).
|           03 DATE-YYYYDDD             pic x(7).
|           03 DATE-YYYYDDDP            pic 9(7) packed-decimal.
|           03 DATE-YYYYMMDDb          pic 9(8) binary.
|           03 DATE-CYYDDD             pic 9(6) packed-decimal.
|           03 DATE-DDMMYYYY           pic 9(8).
|           03 DATE-YYMMDD OCCURS 6 times pic 9(6) packed-decimal.
|           03 filler                   pic x.

```

Example 1

Add 60 days to all date fields and report any errors. Note all array fields are modified.

```

| $$FILEM DSC INPUT=DDIN,
| $$FILEM IGNLEN=YES,
| $$FILEM TCIN=h1q.COBOL(SAMPLE),
| $$FILEM OUTPUT=DDOUT,PROC=*
| MOD_DATE('DATE-YYYYMMDD','YYYYMMDD','+60',,,,Y)
| MOD_DATE('DATE-YYYYMMDDb','YYYYMMDD','+60',,,,Y)
| MOD_DATE('DATE-MM-DD-YYYY','MM/DD/YYYY','+60',,,,Y)
| MOD_DATE('DATE-DD-MM-YYYY','DD/MM/YYYY','+60',,,,Y)
| MOD_DATE('DATE-YYYYDDD','YYYYDDD','+60',,,,Y)
| MOD_DATE('DATE-YYYYDDDP','YYYYDDDP','+60',,,,Y)
| MOD_DATE('DATE-CYYDDD','CYYDDD','+60',,,,Y)
| MOD_DATE('DATE-DDMMYYYY','DDMMYYYY','+60',,,,Y)
| MOD_DATE('DATE-YYMMDD','YYMMDD','+60',,,,Y)
| /*

```

Example 2

Same as example 1 without using a copybook.

```

| $$FILEM DSC INPUT=DDIN,
| $$FILEM OUTPUT=DDOUT,PROC=*
| MOD_DATE(1,8,C,'YYYYMMDD','+60',,,,Y)
| MOD_DATE(40,4,B,'YYYYMMDD','+60',,,,Y)
| MOD_DATE(9,10,C,'MM/DD/YYYY','+60',,,,Y)
| MOD_DATE(19,10,C,'DD/MM/YYYY','+60',,,,Y)
| MOD_DATE(29,7,C,'YYYYDDD','+60',,,,Y)
| MOD_DATE(36,,P,'YYYYDDDP','+60',,,,Y)
| MOD_DATE(44,,P,'CYYDDD','+60',,,,Y)
| MOD_DATE(48,8,C,'DDMMYYYY','+60',,,,Y)
| MOD_DATE(56,,P,'YYMMDD','+60',,,,Y)
| MOD_DATE(60,,P,'YYMMDD','+60',,,,Y)
| MOD_DATE(64,,P,'YYMMDD','+60',,,,Y)
| MOD_DATE(68,,P,'YYMMDD','+60',,,,Y)
| MOD_DATE(72,,P,'YYMMDD','+60',,,,Y)
| MOD_DATE(76,,P,'YYMMDD','+60',,,,Y)
| /*

```

Example 3

Add 2 months and 20 days to all date fields and ignore errors.

```

| $$FILEM DSC INPUT=DDIN,
| $$FILEM IGNLEN=YES,
| $$FILEM TCIN=h1q.COBOL(SAMPLE),
| $$FILEM OUTPUT=DDOUT,PROC=*
| MOD_DATE('DATE-YYYYMMDD','YYYYMMDD','+20','+2')
| MOD_DATE('DATE-YYYYMMDDb','YYYYMMDD','+20','+2')
| MOD_DATE('DATE-MM-DD-YYYY','MM/DD/YYYY','+20','+2')
| MOD_DATE('DATE-DD-MM-YYYY','DD/MM/YYYY','+20','+2')
| MOD_DATE('DATE-YYYYDDD','YYYYDDD','+20','+2')
| MOD_DATE('DATE-YYYYDDDP','YYYYDDDP','+20','+2')

```

Function reference: External REXX functions

```
| MOD_DATE('DATE-CYYDDD','CYYDDD','+20','+2')  
| MOD_DATE('DATE-DDMMYYYY','DDMMYYYY','+20','+2')  
| MOD_DATE('DATE-YYMMDD','YYMMDD','+20','+2')  
| /*
```

Example 4

Set all date field values to the current date plus 1 for year, month, and date.
Change the century window used for DATE-YYMMDD field to 60.

```
| $$FILEM DSC INPUT=DDIN,  
| $$FILEM IGNLEN=YES,  
| $$FILEM TCIN=h1q.COBOL(SAMPLE),  
| $$FILEM OUTPUT=DDOUT,PROC=*  
| MOD_DATE('DATE-YYYYMMDD','YYYYMMDD','*+1','*+1','*+1')  
| MOD_DATE('DATE-YYYYMMDDB','YYYYMMDD','*+1','*+1','*+1')  
| MOD_DATE('DATE-MM-DD-YYYY','MM/DD/YYYY','*+1','*+1','*+1')  
| MOD_DATE('DATE-DD-MM-YYYY','DD/MM/YYYY','*+1','*+1','*+1')  
| MOD_DATE('DATE-YYYYDDD','YYYYDDD','*+1','*+1','*+1')  
| MOD_DATE('DATE-YYYYDDDP','YYYYDDD','*+1','*+1','*+1')  
| MOD_DATE('DATE-CYYDDD','CYYDDD','*+1','*+1','*+1')  
| MOD_DATE('DATE-DDMMYYYY','DDMMYYYY','*+1','*+1','*+1')  
| MOD_DATE('DATE-YYMMDD','YYMMDD','*+1','*+1','*+1','60')  
| /*
```

NCONTAIN

Syntax

```
→ NContain(number, match) →
```

Compares the value represented by *number* against the value or values defined by *match*.

Returns

If the numeric value of any of the *match* arguments is equal to the numeric value of *number*, then NCONTAIN returns 1. Otherwise, NCONTAIN returns 0.

number

The value, represented by a number, a function returning a value or a variable to which a value has been assigned, that you are comparing with *match*.

match The value or list of values that you are comparing with *number*. You can search for up to 20 values at a time.

For a similar function for matching string values, see “CONTAINS” on page 1078.
For a FASTREXX-eligible equivalent, see “FLD_CO” on page 1082.

Example 1

If the current record contains a packed decimal value of 10, 20, or 30 starting at column 8, then print the record.

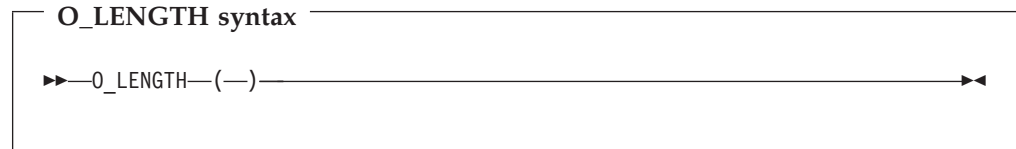
```
If NCO(FLD(8,P),10,20,30) Then PRINT(inrec,'CHAR')
```

Example 2

If the current record contains a zoned decimal value of 11, 12, or 13 starting at column 10, then drop the record from processing.

```
If NCO(FLD(10,5,Z),11,12,13) Then Return 'DROP'
```

O_LENGTH



Can be used in FASTREXX condition expressions.

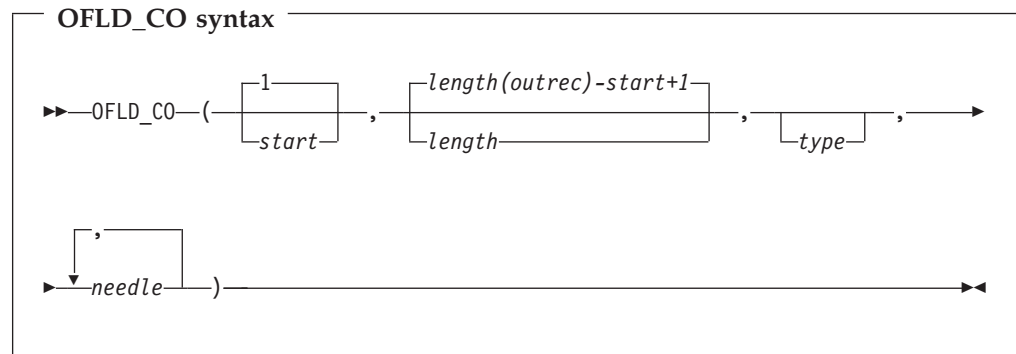
Returns the length, in bytes, of the output record.

Example 1

If the current length of the output record is 100, overlay the last 20 columns with asterisks.

```
If o_length() = 100 Then
  OVLV_OUT('* ',81,20,'*')
```

OFLD_CO



(Can be used in FASTREXX condition expressions.)

Searches the field within the output record specified by *start* and *length*, for one or more occurrences of *needle*. On successful execution when searching for a string, also updates the value of OUTPOS to the first byte of the located field in the output record.

Returns

If at least one occurrence of *needle* is found, returns 1. If no occurrences are found, returns 0.

start Position, in bytes, in the output record at which to start searching for occurrences of *needle*. Can be specified as:

Function reference: External REXX functions

Absolute position

Must be a positive integer. Default value is 1. If *start* is greater than the current length of the output record, the function has no effect.

Relative to current INPOS

Must be specified as IP*x* or IN*x*. If this resolves to a value of less than or equal to zero, the function results in an error. If this resolves to a value that is greater than the current length of the input record, the function has no effect.

Relative to current OUTPOS

Can be specified as OP*x* or ON*x*, or as P*x* or N*x*. If this resolves to a value of less than or equal to zero, the function results in an error. If this resolves to a value that is greater than the current length of the output record, the function has no effect.

- length* Length, in bytes, of the field to be searched.
- For character fields, the length defaults to the remaining record length from the start position to the end of the record (inclusive). A value of 0 also indicates that the field extends to the end of the record.
 - For binary fields, you must specify the length. It can be 2, 4, or 8.
 - For packed decimal fields, if you specify the length, it must be in the range 1-16. If you omit the length, FLD_CO attempts to determine the packed field length from the record data.
 - For zoned decimal fields, if you specify the length, it must be in the range 1-31 or, if the field contains a separate sign character, in the range 1-32. If you omit the length, it defaults to the remainder of the record.
- type* The data type of the field. Valid values are:
- B** Binary. FLD_CO interprets binary fields as being signed.
 - C** Character. This is the default. The comparison is case sensitive.
 - P** Packed decimal.
 - U** Interprets the field as Character, but converts it to uppercase before comparing it with *needle*.
 - Z** Zoned decimal. Interprets all of the COBOL external decimal variants as numeric data.
- needle* String(s) or numeric(s) to search for. For the character types, FLD_CO searches the *haystack* for each of the *needles*. In this context, it behaves like a combination of the FLD and CONTAINS functions. For the numeric data types, the *haystack* is treated as a single numeric field, and an appropriate numeric comparison is performed against each of the *needles*. In this context it behaves like a combination of the FLD and NCONTAIN functions. You can search for up to 20 strings or numerics at a time.

To perform case-insensitive searches, specify *type* as 'U' and *needle* in uppercase.

Example 1

If the current output record contains "MIKE", "Mike" or "mike" in the first ten columns, then write the record.

```
If FLD_CO(1,10,'U','MIKE') Then WRITE('MDD')
```

Example 2

If the current output record contains "USA", "Australia" or "England", then drop the record from processing.

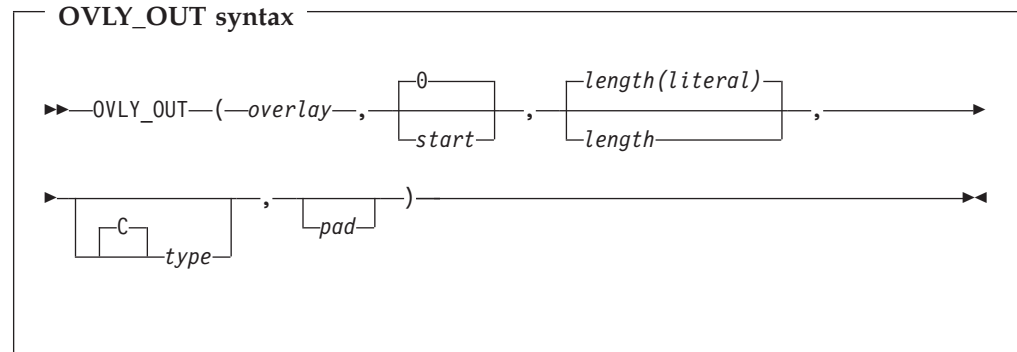
```
If FLD_CO(,, 'USA', 'Australia', 'England') Then Return 'DROP'
```

Example 3

If the current output record contains "MIKE", "Mike" or "mike" in the ten columns immediately after the current INPOS, then write the record.

```
If FLD_CO(P0,10, 'U', 'MIKE') Then WRITE('MDD')
```

OVLY_OUT



Note: Commas following the last specified argument can be omitted.

Can be used in FASTREXX procedures.

Overlays the output record with a string. If the length of the target field exceeds the length of the literal, the target field is padded to the specified length using the pad character. If the length of the target field is less than the length of the literal, the following occurs:

- Character overlays are truncated on the right without error. For example, `OVLY_OUT('ABCD',1,2)` overlays 'AB'.
- Numeric overlay truncations are considered to be an error. For example, `OVLY_OUT(500000,1,2,'B')` fails because you can't fit the specified value into a 2-byte binary field.

On successful execution, also updates the value of `OUTPOS` to one byte past the end of the field overlaid in the output record.

Returns

A single blank.

overlay An expression that resolves to a string, which is overlaid on that part of the output record specified by *start* and *length*. To be eligible for FASTREXX processing, this must be a literal string, a symbol or a blank-delimited sequence of symbols and/or literal strings.

If the first character of the literal is an ampersand and the literal that follows matches an existing character or numeric variable or tally literal (matching not case-sensitive), then the variable value is substituted according to the type. For example, if the type is character and a numeric or tally value are referenced, then the literal is the numeric value in

Function reference: External REXX functions

display format with no leading zeros. If the type is binary, packed or zoned, then the variable value is converted to binary, packed or zoned number.

Notes:

1. Conversion errors may occur when converting a character variable to a numeric.
2. If a variable name is not found, then the string is interpreted as a literal.

start Position, in bytes, in the output record at which to start overlaying the string. If you omit *start*, specify zero, or specify a value one greater than the length of the current output record, the field is appended to the end of the output record. If *start* is greater than the current length of the output record, the record is padded with the specified or defaulted pad character from the current record length to the specified start position. Can be specified as:

Absolute position

Must be a non-negative integer that is less than or equal to the maximum length of the output data set. Default value is 0.

Relative to current INPOS

Must be specified as IP*x* or IN*x*. Must resolve to a non-negative integer.

Relative to current OUTPOS

Can be specified as OP*x* or ON*x*, or as P*x* or N*x*. Must resolve to a non-negative integer.

length Length, in bytes, of target field in the output record. Defaults as shown here:

Character fields

Defaults to the length of the literal. A value of 0 indicates that the target field length is the greater of the source (literal) length and the remaining record length. In particular, if 0 is specified for both start and length, then the length of the literal is used as the target length.

Packed decimal

Defaults to the last packed length value determined from the input record by a preceding function. For example,

```
if FLD(1,P) = 2 then  
  ONLY_OUT('5',1,,P)
```

Uses the length determined by the FLD function to default the packed decimal length. If no previous packed decimal length was calculated, a length error occurs and the procedure is terminated.

type The data type of the literal to be written to the output record.

- B** Binary. The literal string must represent a positive or negative integer, and is stored in the output field as a signed two's-complement format binary number, right-justified in the target field. The length must be 2, 4, or 8, and cannot be omitted.
- C** Character. This is the default.
- P** Packed decimal. The literal string must represent a positive or negative integer, and is stored right-justified in the target field as a

signed packed decimal number using the preferred positive ('c'x) and negative ('d'x) sign indicators. The length must be between 1 and 16.

Z Zoned decimal (COBOL external decimal with non-character trailing sign). The literal string must represent a positive or negative integer, and is stored in the output field as a signed zoned decimal number. The length must be between 1 and 31.

pad Pad character. Defaults to the pad character set on the File Manager System Processing Options panel. If the current pad setting is OFF, the default pad character is a blank. For numeric types such as B, P or Z, the pad character is not used when pre-fill characters are required to right-justify a numeric field. The pre-fill characters are always leading zeros, as required by the field type."

See FLD_OUT for a function to overlay the output record with a field from the input record.

Example 1

Set columns one and two of the output record to asterisks.

```
OVLY_OUT('**',1,2)
```

Example 2

Append the two-byte packed decimal value 2 to the end of the output record.

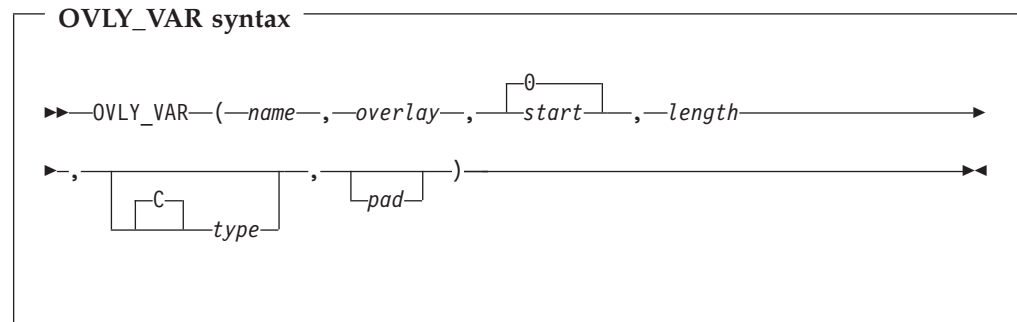
```
OVLY_OUT(2,0,2,P)
```

Example 3

Search the input record for a literal and then overlay the last two bytes of that literal with a new literal in the output record.

```
If FLD_CO(1,,C,'AABB') Then
  OVLY_OUT('CC',IP2,2)
```

OVLY_VAR



(Can be used in FASTREXX procedures.)

Note: Commas following the last specified argument can be omitted.

Function reference: External REXX functions

Overlays the named character variable with a string. If the length of the target field exceeds the length of the literal, the target field is padded to the specified length using the pad character. If the length of the target field is less than the length of the literal:

- Character overlays are truncated on the right without error. For example, `OVLV_VAR(MYVAR, 'ABCD', 1, 2)` overlays 'AB'.
- Numeric overlay truncations are considered to be an error. For example, `OVLV_VAR(MYVAR, 500000, 1, 2, 'B')` fails because you cannot fit the specified value into a 2-byte binary field.

On successful execution, also updates the value of current variable position to one byte past the end of the field overlaid in the variable.

Returns

A single blank.

name 1–256 character variable identifier. Variable name matching is not case sensitive. If the name is not found, a severe error occurs and the procedure is terminated. Cannot be a system character variable or a system numeric variable. See “Using FASTREXX variables” on page 1068.

overlay An expression that resolves to a string, which is overlaid on that part of the variable specified by *start* and *length*. To be eligible for FASTREXX processing, this must be a literal string, a symbol, or a blank-delimited sequence of symbols or literal strings. If the first character of the literal is an ampersand and the literal that follows matches an existing character, numeric variable, or tally literal (matching not case-sensitive), then the variable value is substituted according to the type. For example, if the type is character and a numeric or tally value are referenced, then the literal is the numeric value in display format with no leading zeros. If the type is binary then the variable value is converted to a binary number.

Note: Conversion errors may occur.

start Position, in bytes, in the variable at which to start overlaying the string. If you omit *start*, specify zero, or specify a value one greater than the length of the current variable, the field is appended to the end of the variable. If *start* is greater than the current length of the variable, the record is padded with the specified or defaulted pad character from the current variable length to the specified start position. Can be specified as:

Absolute position

Must be a non-negative integer that is less than or equal to the maximum length of the output data set. Default value is 0.

Relative to current variable position

Must be specified as IPx or Inx, OPx or ONx, or as Px or Nx. Must resolve to a non-negative integer.

length Length, in bytes, of target field in the variable. Defaults:

Character fields

To the length of the literal. A value of 0 indicates that the target field length is the greater of the source (literal) length and the remaining variable length. In particular, if 0 is specified for both *start* and *length*, then the length of the literal is used as the target length.

Packed decimal

To the last packed length value determined from the input record by a preceding function. For example:

```
if FLD(1,P) = 2 then
  OVLY_VAR(MYVAR,'5',1,,P)
```

Uses the length determined by the FLD function to default the packed decimal length. If no previous packed decimal length was calculated a length error occurs and the procedure is terminated.

type The data type of the field. Valid values are:

- B** Binary. FLD_CO interprets binary fields as being signed.
- C** Character. This is the default. The comparison is case sensitive.
- P** Packed decimal.
- U** Interprets the field as Character, but converts it to uppercase before comparing it with *needle*.
- Z** Zoned decimal. Interprets all of the COBOL external decimal variants as numeric data.

pad Pad character. Defaults to the pad character set on the File Manager System Processing Options panel. If the current pad setting is OFF, the default pad character is a blank. For numeric types such as B, P, or Z, the pad character is not used when pre-fill characters are required to right-justify a numeric field. The pre-fill characters are always leading zeros, as required by the field type.

Example 1

Set columns one and two of the variable to asterisks.

```
OVLY_VAR(MYVAR,'**',1,2)
```

Example 2

Append the two-byte packed decimal value 2 to the end of the variable.

```
OVLY_VAR(MYVAR,2,0,2,P)
```

Example 3

Search the variable for a literal and then overlay the last two bytes of that literal with a new literal in the variable.

```
If TESTC(MYVAR,'CU','AABB') Then
  OVLY_VAR(MYVAR,'CC',IP2,2)
```

PRINT

Syntax

```
▶▶—PRINT(record,format)—————▶▶
```

Prints the *record* string in the specified *format*. The print output destination is determined as follows:

Function reference: External REXX functions

Procedure invocation	Print output destination
From an online panel	Determined by the value of the PRINTOUT field on the Set Print Processing Options panel.
From a batch job	SYSPRINT
From REXX program	Determined by the PRINTOUT parameter of the SET function.

Returns

- 0 Function was successful.
- 4 Function was unsuccessful and the record is not printed because either:
 - A *format* of SNGL or TABL was specified and File Manager cannot determine the record type.
 - A template has been specified and the record did not pass the selection criteria.
- 12 Function was unsuccessful and the record is not printed because either:
 - You did not provide parameters.
 - A *format* of SNGL or TABL was specified, but no template or copybook was provided

record Any string or variable representing a string can be used, however, INREC or OUTREC are most commonly used in this function.

format Can be CHAR, HEX, SNGL or TABL. If you specify TABL or SNGL format:

- On the function or panel that you are enhancing, you must specify a copybook or template that describes the record type to be printed.
- File Manager determines the type of the record to be printed by comparing its length with the record types in the template, and also by using any record identification criteria in the template.
- When using PRINT with DSC or the Copy Utility (option 3.3) and you have specified both an input and an output copybook or template, then the copybook or template used to format the printed record is determined as follows: if the value of the record to be printed matches the input record (INREC variable), then the input copybook or template is used; otherwise, the output copybook or template is used.
Please ensure the record value matches the template that File Manager uses to print the data set.
- Only those fields that have been selected in the template are printed.
- If the record has been reformatted by template processing, the variable INREC contains the input record value and OUTREC contains the reformatted output record.

Note: Avoid using PRINT in a REXX procedure for a:

- DSP function
- FCH function
- Print Utility (option 3.2)

because output from PRINT is interspersed with their normal output, which can be confusing. Using PRINT in a REXX procedure that runs from the

Find/Change Utility panel (option 3.6), does not result in this problem, because the report produced by the panel is sent to a data set, separate from print output.

Example 1

Print the first hundred records.

```
If PRTCOUNT() < 100 Then PRINT(inrec,'CHAR')
```

Example 2

Print the current input record in TABL format.

```
rc = PRINT(inrec,'TABL')
```

PRTCOUNT

<p>Syntax</p> <p>►► PRTCOUNT() ◀◀</p>
--

Can be used in FASTREXX condition expressions.

Returns the current count of records printed. The count is incremented for each record printed by the DSP function or Print Utility (option 3.2), and for each invocation of the PRINT function.

Example

Print the first 10 input records.

```
If PRTCOUNT() < 10 Then PRINT(inrec,'CHAR')
```

RECCUR (DSEB only)

<p>Syntax</p> <p>►► RECCUR() ◀◀</p>
--

Returns the current record number.

Example

If the current record is the hundredth record in the file, then print it.

```
If RECCUR() = 100 Then PRINT(inrec,'CHAR')
```

RECSIN

Syntax

```

▶▶ RECSIN() ◀◀
    
```

Can be used in FASTREXX condition expressions.

Returns the count of records read so far from the input data set. When the input data set is a PDS, the RECSIN count restarts for each member being processed.

When used with DSEB, RECSIN returns the record number of the furthest record read so far in the data set. (For example, if you have moved down as far as record number 500 in the data set, then you move up to a previous record, RECSIN still returns 500 after moving up.)

Example

Print every hundredth record.

```
If RECSIN()//100 = 0 Then PRINT(inrec,'CHAR')
```

RECSOUT

Syntax

```

▶▶ RECSOUT((1)ddname) ◀◀
    
```

Notes:

- 1 When used with a DSC or DSP function, *ddname* is optional.

Can be used in FASTREXX condition expressions.

Returns the count of records so far written to the specified output data set.

The argument you can specify is:

ddname

Specifies that the count of records so far written to the data set identified by the specified *ddname* be returned. If *ddname* is omitted when used with a DSC or DSP function, the default is the *ddname* of the primary output data set. The primary output data set depends on the File Manager function or panel being used:

Function or panel option	Primary output data set is...
Print Utility (option 3.2)	Determined by the value of the PRINTOUT field on the Set Print Processing Options panel. For details, see "Printing from File Manager" on page 274.

Function or panel option	Primary output data set is...
DSC function Copy Utility (option 3.3)	The data set that is the target of the copy function.
DSP function	When used in a batch job, the primary output data set is SYSPRINT. When used in a REXX procedure, the primary output data set is determined by the PRINTOUT parameter of the SET function. For details, see "SET (Set Processing Options)" on page 1019.

Specifying the ddname of the primary output data set is the same as omitting the argument.

If you specify a ddname that is not the ddname of the primary output data set and has not previously been specified as the argument to a WRITE function, the value returned is zero.

The count of records written to an output data set is incremented each time a WRITE function is issued against the specified data set. In the case of the primary output data set, the count is also incremented each time a record is written to the data set by the File Manager function. Unless a record is discarded using the RETURN DROP (or STOP IMMEDIATE) instruction, each record selected for processing is written to the primary output data set. For information about how to discard records, see "RETURN return values" on page 1125.

Note: The RECSOUT function treats each member of the primary output data set as separate, that is, the count starts at zero for each output member. However, the count is maintained across members of the input data set so that, if copying from a PDS to a sequential data set, the RECSOUT function reflects the total number of records written, regardless of how many input members are involved.

If you are using the DSC function or Data Copy Utility and you have specified REXX member selection, the RECSOUT function is disabled for the primary output data set. When RECSOUT targets a *ddname* other than the primary output data set, it still functions as normal. However, you must keep in mind that after a decision has been made to DROP or PROCESS the member, no further records are passed to your REXX procedure, so subsequent records are not counted.

Example 1

If more than one hundred records have been written to the EXT100 file, then terminate File Manager processing.

```
rc = WRITE(EXT100)
If RECSOUT(EXT100) > 100 Then Return 'STOP'
```

RSTR_OUT

Syntax
<pre>►► RSTR_OUT () ◀◀</pre>

Function reference: External REXX functions

(Can be used in FASTREXX condition expressions.)

Restores the most recently saved copy of the output buffer.


There are no synchronization restrictions. The SAVE_OUT() invocation that a RSTR_OUT() invocation is “reversing” could have occurred for the current record, or any previously processed record.

There is no stacking of saved output buffers. If RSTR_OUT() is invoked twice in succession, then the second invocation restores the same data as the first.

If RSTR_OUT() is invoked without a prior invocation of SAVE_OUT(), then the output buffer is “cleared”. The effect is the same as executing SET_OLEN(0).

SAVE_OUT

Syntax



```
▶▶—SAVE_OUT()—◀◀
```

(Can be used in FASTREXX condition expressions.)

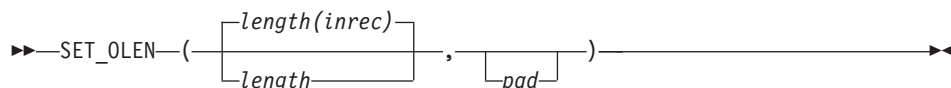
Saves a copy of the current output buffer.

There are no synchronization restrictions. The SAVE_OUT() invocation that a RSTR_OUT() invocation is “reversing” could have occurred for the current record, or any previously processed record.

There is no stacking of saved output buffers. If SAVE_OUT() is invoked twice in succession, then the data saved by the first invocation is lost.

SET_OLEN

SET_OLEN syntax



```
▶▶—SET_OLEN—(— $\left. \begin{array}{l} \text{length}(inrec) \\ \text{length} \end{array} \right\}$ —, — $\left[ \text{pad} \right]$ —) —◀◀
```

Note: Commas following the last specified argument can be omitted.

Can be used in FASTREXX procedures.

Sets the length of the output record. If the specified length is greater than the current length of the output record, the pad character is used to fill out the record to the specified length. If the specified length is less than the current OUTPOS, OUTPOS is reset to the new length + 1.

Returns

A single blank.

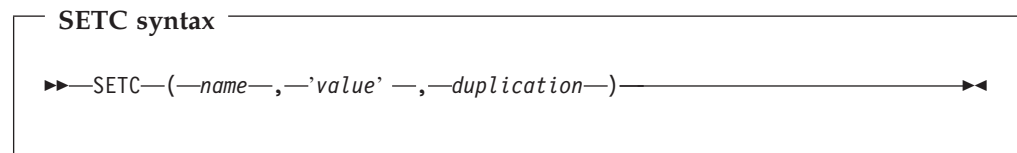
length Length, in bytes, to set. Must be a non-negative integer. The default value is the length of the input record (which is also the original length of the output record).

pad Pad character. Defaults to the pad character set on the File Manager System Processing Options panel. If the current pad setting is OFF, the default pad character is a blank.

Example

Set the length of the current output record to 80.
 SET_OLEN(80)

SETC



Can be used in FASTREXX condition expressions.

Defines or changes a character variable.

name 1–256 character variable identifier. Variable name matching is not case sensitive. Cannot be a system character variable or a system numeric variable. See “Using FASTREXX variables” on page 1068.

value A single value can be entered:

- Specifying hexadecimal strings. A hexadecimal string must be in the form 'hhhhh'h'x. The value enclosed in quotes must be an even number of characters and contain valid hexadecimal characters (0–9, A–F).
- Specifying binary strings. A binary string must be in the form 'nnnnnn'b. The value enclosed in quotes must be a combination of "0"s and "1"s.
- Specifying character strings. For non-numeric types, the value should be enclosed in quotes.
- Specify a variable by specifying &variable_name. A variable is substituted for the value if a matching character, numeric, or tally variable can be located. If a matching variable cannot be found, the string is treated as a literal value. If a numeric or tally variable is referenced, then the value is the number converted to its display form with leading zeros removed.

duplication

Specify an integer *n* to duplicate the literal value *n* times.

Note: This can only be used where the value is a literal constant and not a substitute variable.

Example 1

Set up a variable called A10 with A value repeated 10 times.

```
If RECSIN() <= 0
SETC ('CHECK_CHAR', 'A', 10)
else
```


Function reference: External REXX functions

```
| SETC('CHECK_CHAR','B',10)
|
| If FLDI(20,10,, 'EQ', '&CHECK_CHAR') then
|   Return
| Else
|   Return "DROP"
```

Example 2

Copy the tenth input record into variable REC10.

```
| If RECSIN() = 10 then
|   SETC('REC10', '&ZINREC')
```

SETN

SETN syntax

```
▶▶ SETN (—name—, —value—) ◀◀
```

Can be used in FASTREXX condition expressions.

Defines or changes a numeric variable.

name 1–256 character variable identifier. Variable name matching is not case sensitive. Cannot be a system character variable or a system numeric variable. See “Using FASTREXX variables” on page 1068.

value An integer or substitute numeric or tally variable preceded with an optional plus or minus sign. The number can set, increment, or decrement the respective value in the variable. The variable is initialized with a zero value if it does not exist. The presence of a plus or minus symbol indicates the number provided is to increment or decrement the current value respectively. The absence of plus or minus symbols indicates the number is to replace the current value. Ensure that the value is enclosed in quotes if a plus or minus sign is specified. A substitute variable name should begin with an ampersand (&*variable_name*). This must refer to an existing tally or numeric variable. If the variable does not exist, the procedure fails with a severe error. The maximum number supported is 31 digits plus sign.

Example 1

Count the number of records with surname smith that have an age over 50.

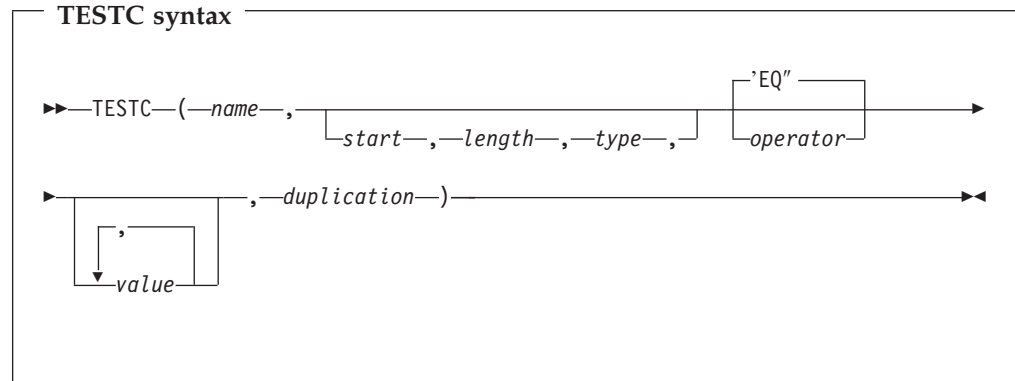
```
| If FLDI(1,20,C, 'CU', 'SMITH') and FLD(25,,P) > 50 then
|   SETN ('Smith_over_50', '+1')
```

Example 2

Save tally value for first 50 record.

```
| TALLY(1,P, 'Total Car crashes')
| IF RECSIN() = 50 then
|   SETN('Total_first_50', '&total car crashes')
```

TESTC



Can be used in FASTREXX condition expressions.

Notes:

1. The operator (operator) and non-numeric values should all be enclosed in quotes to avoid syntax errors.
2. The start length and type parameters are optional to provide the capability to examine part of the variable. The second operand can be the operator. See examples provided.
3. If you specify a value for *length* that would cause the record length to be exceeded, a false result is returned.
4. If you specify a numeric type (types B, P, Z), and the specified field contains invalid data for that type, then the function returns a false result. Numeric data is always returned in integer form; that is, the function does not perform scaling of numeric data.

name 1–256 character variable identifier. Variable name matching is not case sensitive. If the name is not found, a variable is created with a length of 1 and value X'00'.

start Position in bytes in the variable at which to start reading the field value. Can be specified as:

Absolute position

Must be a positive integer. If *start* is greater than the current length of the input record, the function has no effect.

Relative start position

Can be specified as IPx or INx, or as Px or Nx, OPx, or ONx.

When testing a variable, the current variable position is the basis of calculation. The current variable position is changed by the TESTC function with a contains type operator. If this resolves to a value of less than or equal to zero, the function produces a false result. If this resolves to a value that is greater than the current length of the variable, the function produces a false result.

length The length of the field in bytes.

- For binary fields, you must specify the length. It can be 2, 4, or 8.
- For character fields, if you omit the length, the length is defaulted:
 - For contains type operators, to the rest of the variable.
 - When all values are variable substitutions, to the rest of the variable.
 - Otherwise, the maximum literal value length is used.

Function reference: External REXX functions

- For packed decimal fields, if you specify the length, it must be in the range 1–16. If you omit the length, the function attempts to determine the packed field length from the variable value and returns only that field.
- For zoned decimal fields, if you specify the length, it must be in the range 1–31 or, if the field contains a separate sign character, in the range 1–32. If you omit the length, the function returns the remainder of the variable. If this exceeds 32, then the function returns a false result.

type The data type of the field. Valid values are:

- B** Binary. The function interprets binary fields as being signed.
- C** Character. This is the default.
- P** Packed decimal.
- U** Interprets the field as character, but converts it to uppercase before returning the string.
- Z** Zoned decimal. Interprets all of the COBOL external decimal variants as numeric data.

operator

The default is EQ or =. This function supports all the operators described for dynamic template and criteria edit. For details about the operators supported and their description, see:

- “Dynamic Template panel” on page 510.
- “Record Identification Criteria panel” on page 602.
- “Record Selection Criteria panel” on page 607.

value The value or values entered must be valid in the context of the operator and the field which is being referenced. For example, only certain operators like CO (contains) allow multiple values. Numeric values should be entered when testing numeric fields, and so on.

- Specifying hexadecimal strings. A hexadecimal string must be in the form 'hhhhh'h'x. The value enclosed in quotes must be an even number of characters and contain valid hexadecimal characters (0–9, A–F).
- Specifying binary strings. A binary string must be in the form 'nnnnnn'b. The value enclosed in quotes must be a combination of "0"s and "1"s.
- Specifying character strings. For non-numeric types, the value should be enclosed in quotes.
- Specify a variable by specifying &variable_name. A variable is substituted for the value if a matching character, numeric, or tally variable can be located. If a matching variable cannot be found, the string is treated as a literal value. If a numeric comparison is being performed, a character variable is converted to a number - if the conversion fails, the function returns a false result. If a numeric or tally variable is referenced in a character comparison, then the value is the number converted to its display form with leading zeros removed.

duplication

Specify an integer *n* to duplicate the literal value *n* times.

Note: This can only be used for operators that support a single value (for example, Not contains) and where the value is a literal constant and not a substitute variable.

Example 1

SET variable named TESTREC to current input record. Check the variable and process only those records that contain values of 'Smith' or 'Jones'.

Note: In this case, use the operator CU so the contains processing is not case-sensitive. The start, length, and type parameters have been omitted.

```
SETC('TESTREC', '&ZINREC')
if TESTC(TESTREC, 'CU', 'Smith', 'Jones') then
  return
else
  return 'DROP'
```

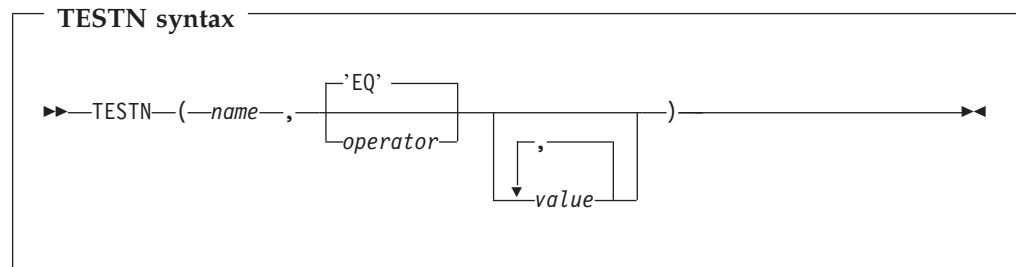
Example 2

Process all records with a salary greater than 75000, where salary is a packed-decimal value found at start position 28.

Note: In this case, allow File Manager to calculate the packed-decimal field length.

```
SETC('TESTREC', '&ZINREC')
if TESTC(TESTRC, 28, P, '>', 75000) then
  return
else
  return 'DROP'
```

TESTN



Can be used in FASTREXX condition expressions.

name This is a 1-256 name that matches either a tally literal or a numeric variable. Variable name matching is not case sensitive. If the name is not found, a numeric variable is created with value of 0.

operator

The default is EQ or =. This function supports all the operators described for dynamic template and criteria edit. For details about the operators supported and their description, see:

- "Dynamic Template panel" on page 510.
- "Record Identification Criteria panel" on page 602.
- "Record Selection Criteria panel" on page 607.

value

The value or values entered must be valid in the context of the operator and the field which is being referenced. For example, only certain operators like CO (contains) allow multiple values. Numeric values should be entered when testing numeric fields, and so on.

Function reference: External REXX functions

- Specifying hexadecimal strings. A hexadecimal string must be in the form 'hhhhh'h'x. The value enclosed in quotes must be an even number of characters and contain valid hexadecimal characters (0–9, A–F).
- Specifying binary strings. A binary string must be in the form 'nnnnnn'b. The value enclosed in quotes must be a combination of "0"s and "1"s.
- Specifying character strings. For non-numeric types, the value should be enclosed in quotes.
- Specify a variable by specifying &variable_name. A variable is substituted for the value if a matching character, numeric, or tally variable can be located. If a matching variable cannot be found, the string is treated as a literal value. If a numeric comparison is being performed, a character variable is converted to a number - if the conversion fails, the function returns a false result. If a numeric or tally variable is referenced in a character comparison, then the value is the number converted to its display form with leading zeros removed.

Example 1

Count the number of records with A in the first byte and stop processing after 20.

```
IF FLD(1,1) = 'A' then
  SETN(COUNTA, '+1')

if TESTN(COUNTA, '>', 20) then
  return "STOP IMMEDIATE"
else
  return
```

Example 2

When the total of packed decimal field start at column 28 is greater than 100, stop processing

Note: In this case, allow File Manager to calculate the packed-decimal field length.

```
TALLY(28,P,'Total Sales')
if TESTN('Total Sales', '>', 100) then
  return "STOP IMMEDIATE"
else
  return
```

TALLY

Syntax

```
▶▶ TALLY(start, length, Z  
type, string) ▶▶
```

Can be used in FASTREXX procedures.

Accumulates the value of the specified input record field in a TALLY register and, at the end of the File Manager function, prints on SYSPRINT the TALLY register prefixed by *string*. The TALLY is maintained across members of a PDS.

Note: If REXX member selection has been used with the DSC function or the Data Copy Utility, input records are only passed to the REXX procedure until a decision has been made on whether to DROP or PROCESS the member. TALLY accumulates the value of the specified input record field for all records processed by the REXX procedure. This includes records that were processed for members that were subsequently DROPped, and excludes records that were not passed to the REXX procedure because a decision had already been made to PROCESS or DROP the member.

Returns

A single blank.

start Start, in bytes, position of the field to be tallied in the input record.

length Length, in bytes, of the field to be tallied in the input record.

type The data type of the field to be accumulated. The values that can be specified are:

B Signed binary. If you specify B for *type*, *length* must be 2, 4, or 8.

P Packed decimal. If you specify P for *type*, *length* must be between 1 and 16.

UB Unsigned binary. If you specify UB for *type*, *length* must be 2, 4, or 8.

Z Zoned decimal. This is the default. If you specify Z for *type*, *length* must be between 1 and 32 or, if the field contains a separate sign character or leading blanks, between 1 and 33. TALLY ignores leading blanks for zoned decimal fields, allowing simple character numeric fields to be tallied. Records containing only blanks in the target field are ignored. If a separate sign is present, there must be no blanks between the sign and the zoned data.

string A literal string that is prefixed to the accumulated TALLY total.

The field whose value is to be accumulated starts at position *start* in the input record, and is *length* bytes long. If the sum of *start* and *length* is more than one greater than LENGTH(INREC), the TALLY function returns a blank without changing the TALLY register.

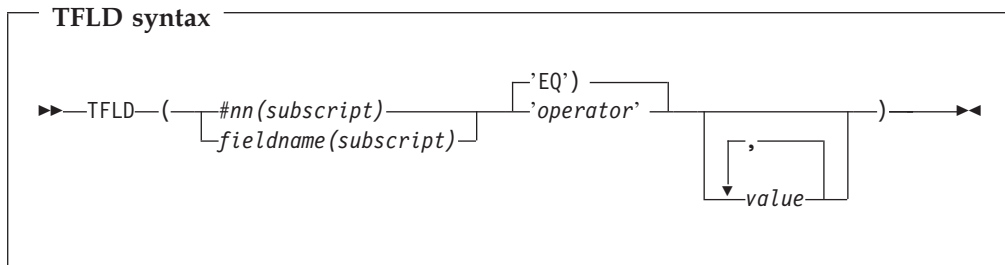
You can code more than one TALLY function in your procedure. File Manager creates a separate TALLY register for each TALLY function with a unique combination of arguments. This means that you can accumulate a given field in more than one TALLY register by specifying a different value for *string* in each TALLY function.

Example

Accumulate hours recorded in personnel records depending on record type.

```
Select
  When (FLD(1,1) == 'E') Then
    TALLY(15,4,B,'Sum of employee hours')
  When (FLD(1,1) == 'S') Then
    TALLY(15,4,B,'Sum of supervisor hours')
  Otherwise
    TALLY(28,4,B,'Sum of manager hours')
End
```

TFLD



(Can be used in FASTREXX condition expressions.)

Performs a conditional test against any field defined in a template. For dimensioned fields, you can apply the condition to any or all of the elements of the array. You can only use this function if the associated function is running with a copybook or template.

Note: The Field name (*fieldname*), field reference (*#nn*), operator (*operator*), and non-numeric values should all be enclosed in quotes to avoid syntax errors.

#nn or *fieldname*

#nn Use this form when providing free-format criteria during template edit. *nn* is the field reference number displayed during template edit. It is not valid to use *fieldname* for a field reference when providing criteria during template edit.

fieldname

Use this form of identifying fields when coding user procedures. For non-unique names, you can specify a name in the form *groupname.dataname*. Name matching is not case-sensitive. If the name is unqualified, then the first occurrence of the name is used. Do not code *#nn* values in user procedures, as the displayed field reference values do not identify the correct field when running from a user procedure.

subscript

This applies *only* to dimensioned fields. You can specify one of these forms:

- (ANY) This is the default if you do not specify a subscript for a dimensioned field and it indicates that at least one element of the associated array must satisfy the condition for a true result.
- (ALL) This indicates that all elements of the associated array must satisfy the condition for a true result.
- (*nn*) This refers to a single array element and you should provide a valid subscript for the dimensioned field.

operator

The default is EQ or =. This function supports all the operators described for dynamic template and criteria edit. For details about the operators supported and their description, see

- "Dynamic Template panel" on page 510
- "Record Identification Criteria panel" on page 602

- “Record Selection Criteria panel” on page 607

value The value or values entered must be valid in the context of the operator and the field which is being referenced. For example, only certain operators like CO (contains) allow multiple values. Numeric values should be entered when testing numeric fields, and so on.

Specifying hexadecimal strings

A hexadecimal string must be in the form 'hhhhh'x. The value enclosed in quotes must be an even number of characters and contain valid hexadecimal characters (0–9, A–F).

Specifying binary strings

A binary string must be in the form 'nnnnn'b. The value enclosed in quotes must be a combination of "0"s and "1"s.

Specifying character strings

For non-numeric types, the value should be enclosed in quotes.

Example 1

Check every element of the dimensioned field CONTACTS and process only those records with contact values of 'Smith' or 'Jones'.

Note: In this case, we use operator CU so the contains processing is not case-sensitive.

```
if TFLD('CONTACTS(ANY)', 'CU', 'Smith', 'Jones') then
  return
else
  return 'DROP'
```

Example 2

Check the monthly pay for a contract record types and process those with every month occurrence higher than 8000.

Note: MPAY is not unique, so we qualify which MPAY we want to check.

```
Copybook
01 REC-CONTRACT.
   05 MPAY      PIC S9(8) Binary OCCURS 12 Times.
01 REC-Employee.
   05 MPAY      PIC S9(8) Binary OCCURS 12 Times.

if TFLD('REC-CONTRACT.MPAY(ALL)', '>', 8000) then
  return
else
  return 'DROP'
```

Performance notes

TFLD is faster than FLD_CO, but requires a template to reference a field value.

```
TFLD('#3',, 'CO', 'A') & TFLD('#3',, 'CO', 'B')
```

would be faster if coded as:

```
TFLD('#3',, 'ACO', 'A', 'B')
```


TM

Syntax

```
▶▶—TM(string,mask)—————▶▶
```

Tests selected bits of a string and sets the condition code accordingly. See “FLD_TM” on page 1085 for a similar function that is FASTREXX eligible.

Returns

If the tested bits are all ones, then TM returns 1. Otherwise, TM returns 0.

string A literal string, or variable representing a string.

mask A bit-string determining which bits to test in *string*.

The length of the test is based on the length of the shorter of the two arguments, *string* and *mask*. A *mask* bit of one indicates that the equivalent bit in *string* is to be tested. When a *mask* bit is zero, the equivalent *string* bit is ignored.

Example 1

Test the third byte of the input record and, if the low order bit is set, overlay a hex FF into the second byte of that record.

```
If TM(FLD(3,1),'01'x) Then Do
  outrec = OVERLAY('FF'x,outrec,2)
  Return
End
Return 'DROP'
```

Example 2

Test the third byte of the input record and, if the high order bit is set, logically OR a hex 04 over the contents of the second byte of that record.

```
If TM(FLC(3,1),'10000000'b) Then Do
  outrec = OVERLAY(BITOR(fld(2,1),'04'x),outrec,2)
  Return
End
Return 'DROP'
```

TOP (DSEB only)

Syntax

```
▶▶—TOP()—————▶▶
```

Moves to the first input record.

UP (DSEB only)

Syntax

```
►►UP(n)◄◄
```

Moves up *n* number of input records, or to the first input record, if there are less than *n* records above the current input record.

If, after moving, the current input record is the first input record, then the UP function returns the string value "TOF" (top of file).

UPDATE (DSEB only)

Syntax

```
►►UPDATE()◄◄
```

Replaces the current input record with the value in OUTREC. If you leave DSEB or move to another record before calling the UPDATE function, then any changes you made to the current OUTREC are lost.

VAR_OUT

VAR_OUT syntax

```
►►VAR_OUT(—name—, —1i_start—, —0i_length—, —0o_start—, —  

i_length  

o_length—, —pad—)◄◄
```

(Can be used in FASTREXX procedures.)

Note: Commas following the last specified argument can be omitted.

Overlays the output record with a field from the variable. See "OVLY_OUT" on page 1099 for a function to overlay the output record with a literal. If the target field length exceeds the source field length, then the source field is padded to the specified length using the *pad* character. If the target field length is less than the source field, the source field is truncated from the right. On successful execution, also updates the value of OUTPOS to one byte past the end of the field overlaid in the output record.

Returns

A single blank.

Function reference: External REXX functions

name A 1–256 character variable identifier. Variable name matching is not case-sensitive. If the name is not found, a variable is created and populated from the current input record.

i_start Position, in bytes, in the variable at which to start reading the field to be copied. Can be specified as:

Absolute position

Must be a positive integer. Default value is 1.

Relative to current INPOS

Can be specified as IP*x* or IN*x*, or as P*x* or N*x*, or as OP*x* or ON*x*. Must resolve to a positive integer.

i_length

Length, in bytes, of the source field. Must be a non- negative integer. Defaults to 0. If you omit *i_length* or specify zero, the remainder of the variable from the *i_start* position is used. This also applies if you specify a value that would cause the source field to be read from beyond the end of the current variable.

o_start Position, in bytes, in the output record at which to start overlaying the copied field. If you omit *o_start* or specify zero, the field is appended to the end of the output record. If *o_start* is greater than the current length of the output record, the record is padded with the specified or defaulted *pad* character from the current record length to the specified *start* position. Can be specified as:

Absolute position

Must be a positive integer. Default value is 1.

Relative to current INPOS

Can be specified as IP*x* or IN*x*, or as P*x* or N*x*, or as OP*x* or ON*x*. Must resolve to a positive integer.

Relative to current OUTPOS

Can be specified as OP*x* or ON*x*, or as P*x* or N*x*. Must resolve to a positive integer.

o_length

Length, in bytes, of the target field. Defaults to the source field length (*i_length*). A value of 0 indicates that the target field length is the greater of *i_length* and the remaining output record length. If 0 is specified for both *o_start* and *o_length*, then *i_length* is used as the target length.

pad

Pad character. Defaults to the pad character set on the File Manager System Processing Options panel (when processing online) or the pad character specified in the SET function (when running in batch). If the current pad setting is OFF or unspecified, the default pad character is a blank.

Example 1

Copy the characters in columns 1 and 2 of the variable to columns 3 and 4 of the output record.

```
VAR_OUT(MYVAR,1,2,3,2)
```

Example 2

Append the characters in columns 11 and 12 of the variable to the end of the output record, padded with two blanks.

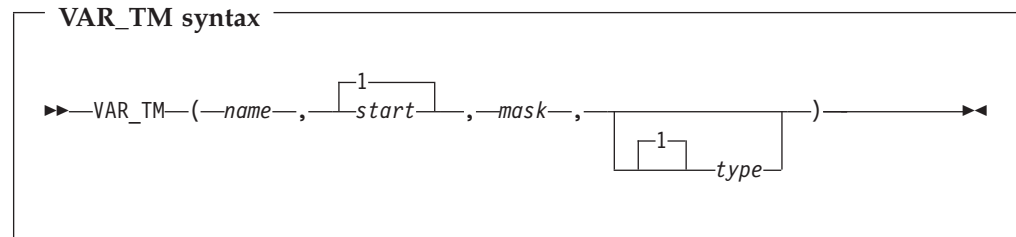
```
VAR_OUT(MYVAR,11,2,0,4,'')
```

Example 3

Search a field in the variable for the characters 'AA' and, if found, copy to the end of the output record (assumes that OUTPOS is still set to end of output record).

```
IF TESTC(MYVAR,'CU','AA') Then
/* when successful, updates variable position to 12 */
  VAR_OUT(MYVAR,N2,2,P0,2)
/* copies from variable position - 2, appends to end of output record */
```

VAR_TM



(Can be used in FASTREXX procedures.)

Note: Commas following the last specified argument can be omitted.

name 1–256 character variable identifier. Variable name matching is not case sensitive. If the name is not found, the function returns a false result.

start Position, in bytes, in the variable at which to start testing. The length of the field is defined by the mask (*mask*). Can be specified as:

Absolute position

Must be a positive integer. Default value is 1. If *start* is greater than the current length of the variable, the function has no effect.

Relative to variable position

Can be specified as *IPx* or *INx*, or as *Px* or *Nx*, or as *OPx* or *ONx*. If this resolves to a value of less than or equal to zero, the function results in an error. If this resolves to a value that is greater than the current length of the variable, the function has no effect.

mask A bit string determining which bits to test in the field. The length of *mask* defines the length of the input field. This field defines a bit string mapping used to test the specified bits in the variable. You can use the bit string, hex string or character string formats to define this field. Therefore, '01000000'b, '40'x, and '' are all legitimate and equivalent ways of defining a mask to test the second bit of a one-byte field.

type The type of test:

- 1** VAR_TM returns True (1) if all the bits that are on in the mask are on in the variable field. This is the default value.
- 0** VAR_TM returns True (1) if all the bits that are on in the mask are off in the variable field.
- M** VAR_TM returns True (1) if at least one of the bits that are on in the mask is on in the variable, and at least one is off.

Function reference: External REXX functions

| N VAR_TM returns True (1) if at least one of the bits that are on in
| the mask is off in the variable field.

Example 1

Test the third byte of the variable and, if the low order bit is set, overlay a hex FF into the second byte of the output record.

```
If VAR_TM(3,'01'x) Then Do  
  OVLY_OUT('ff'x,2,1)  
  Return  
End  
Return 'DROP'
```

Example 2

Test the third byte of the variable and, if some of the three high order bits are set, and some are not, overlay the contents of the second byte of that record with a hex 04.

```
If VAR_TM(3,'1110000'b,M) Then Do  
  OVLY_OUT('04'x,2,1)
```

Example 3

Test the current position of the variable and, if the low order bit is set, overlay a hex FF into the byte prior to this location in the output record.

```
If VAR_TM(P0,'01'x) Then Do  
  OVLY_OUT('ff'x,IN1,1)  
  Return  
End  
Return 'DROP'
```

WRITE

Syntax



Notes:

- 1 When used with a DSC or DSP function, *ddname* is optional.

Can be used in FASTREXX procedures.

Writes a record to the specified data set or sets.

Returns

A single blank.

ddname

Specifies a record is to be written to the data set identified by the specified *ddname*. If *ddname* is omitted when used with a DSC or DSP function, the default is the *ddname* of the primary output data set. The primary output

Function reference: External REXX functions

data set depends on the File Manager function or panel being used:

Function or panel option	Primary output data set is...
Print Utility (option 3.2)	Determined by the value of the PRINTOUT field on the Set Print Processing Options panel. For details, see "Set Print Processing Options panel (option 0.1)" on page 634.
Copy Utility (option 3.3) or DSC function	The data set that is the target of the copy function. For the DSC function, if a ddname is associated with the primary output data set, then specifying that ddname is the same as omitting the argument. Do not target the primary output data set with another ddname.
DSP function	When used in a batch job, the primary output data set is SYSPRINT. When used in a REXX procedure, the primary output data set is determined by the PRINTOUT parameter of the SET function. For details, see "SET (Set Processing Options)" on page 1019. If a ddname is associated with the primary output data set, then specifying that ddname is the same as omitting the argument. Do not target the primary output data set with another ddname.

Except for the primary output data set, the data set attributes of the output data set are derived from:

- The pre-allocated data set DCB attributes. Existing record formats, lengths and block sizes are preserved.
- If it has been allocated without DCB attributes then these attributes are inherited from the input data set.

Sequential data sets specified on a WRITE statement to which no records have been written during the execution of a program are still opened and closed by File Manager. This means that:

- Newly allocated data sets with DISP=(NEW,...) are initialized to an "empty" state and only contain an EOF record.
- Existing data sets with data are handled as follows:
 - DISP=(OLD,...) data sets are reset to an "empty" state and contain only an EOF record; all previously existing data is lost.
 - DISP=(MOD,...) data sets have their data preserved unaffected.

The record is written from the contents of one of the following REXX variables:

- If it has been assigned a value by the procedure, the variable OUTREC.ddname, where ddname is the ddname specified in the WRITE function.
- If the variable OUTREC.ddname has not been assigned a value, or has been unassigned using the DROP instruction, the File Manager-defined variable, OUTREC.

The length of the record written depends upon the data set attributes of the output data set. If the output data set contains variable-length records, the length of the record is determined from the length of the data in the REXX variable. If the length of the data is greater than the maximum record length specified in the data set

Function reference: External REXX functions

attributes, the record is truncated. If the output data set contains fixed-length records, the length of the record written is the length specified in the data set attributes, truncated or padded as necessary.

If the record format of the output data set specifies that the records contain a carriage control character, depending on the output device, the first character of the record data in the REXX variable is interpreted as a carriage control character. For more information about records containing carriage control characters, see the *z/OS DFSORT Installation and Customization*.

If you are using the DSC function or Data Copy Utility and the input data set is a PDS(E) and the *ddname* of the target refers to a PDS(E), then members are created in the target data set with names matching the name of the input data set member as the result of the WRITE execution.

If you are using the DSC function or Data Copy Utility and you have specified REXX member selection, the WRITE function is disabled for the primary output data set. When WRITE targets a *ddname* other than the primary output data set, it still functions as normal. However, you must keep in mind that after a decision has been made to DROP or PROCESS the member, no further records are passed to your REXX procedure, so subsequent WRITES are not processed.

Example 1

If the current record is type 01, then write it to the DD01 file.

```
If FLD(1,2) = 01 Then WRITE('DD01')
```

Example 2

If the current record is type 02, then write it to the DD02 and DD02COPY files.

```
If FLD(1,2) = 02 Then WRITE('DD02', 'DD02COPY')
```

UK44322

Release Date: 3 March 2009

This set of PTFs contains the following APAR fixes:

APAR #	APAR Abstract	Doc Impact
PK79535	LOCATE command selects an exact location rather than selecting an appropriate start position.	User's Guide and Reference (SC19-2495-00)

PK79535

Initial problem description

The LOCATE command requires an exact match and accepts the parameters NEXT, PREV, FIRST, LAST.

Outline of solution

The behavior of the LOCATE command when viewing a Load Module has been changed as follows:

Performs a symbol name or symbol address locate and, if a matching symbol-reference is found, moves the display to the specified reference. If the symbol-reference is not found, the display is moved to the entry logically preceding the symbol-reference.

Documentation impact

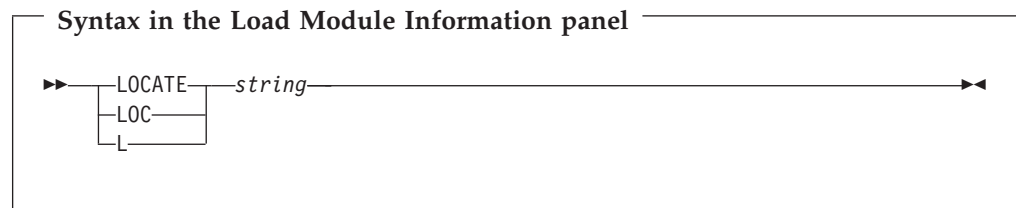
This APAR requires changes to be made to:

- User's Guide and Reference (SC19-2495-00)

Changes to the User's Guide and Reference

Chapter 15, "Primary commands", subsection "LOCATE primary command"

In the syntax diagram entitled "Syntax in the Load Module Information panel", remove the parameters NEXT, PREV, FIRST, and LAST so that the new diagram looks like this:



UK42251

Release Date: 18 December 2008

This set of PTFs contains the following APAR fixes:

APAR #	APAR Abstract	Doc Impact
PK75870	File Manager enhancements to AUDIT logging and reporting.	User's Guide and Reference (SC19-2495-00) User's Guide and Reference for IMS (SC19-2497-00) Customization Guide (SC19-2494-00)

PK75870

Initial problem description

File Manager does not have the ability to:

1. Produce an immediate report when SMF logging has been requested.
2. Display non displayable characters in hexadecimal on a formatted audit report.
3. Show key fields with changed fields.
4. Highlight changes on audit report.

Outline of solution

File Manager will be enhanced to support:

1. A demand installation option that forces an audit trail and automatically submits a job on completion of the edit session.
2. A Show Key option which will flag key fields and include key fields in the report when only changed fields have been requested.
3. Two hexadecimal display options - one to display hex for all fields on a formatted display. The other to display hex for characters that cannot be currently displayed.
4. A highlight changes option to add an asterisk to the report to indicate which fields have been changed.

Documentation impact

This APAR requires changes to be made to:

- User's Guide and Reference (SC19-2495-00)
- User's Guide and Reference for IMS (SC19-2497-00)
- Customization Guide (SC19-2494-00)

Changes to the User's Guide and Reference

Chapter 14, "Panels and fields", subsection "Print Audit Trail panel"

Replace the figure "The Print Audit Trail panel" with this:

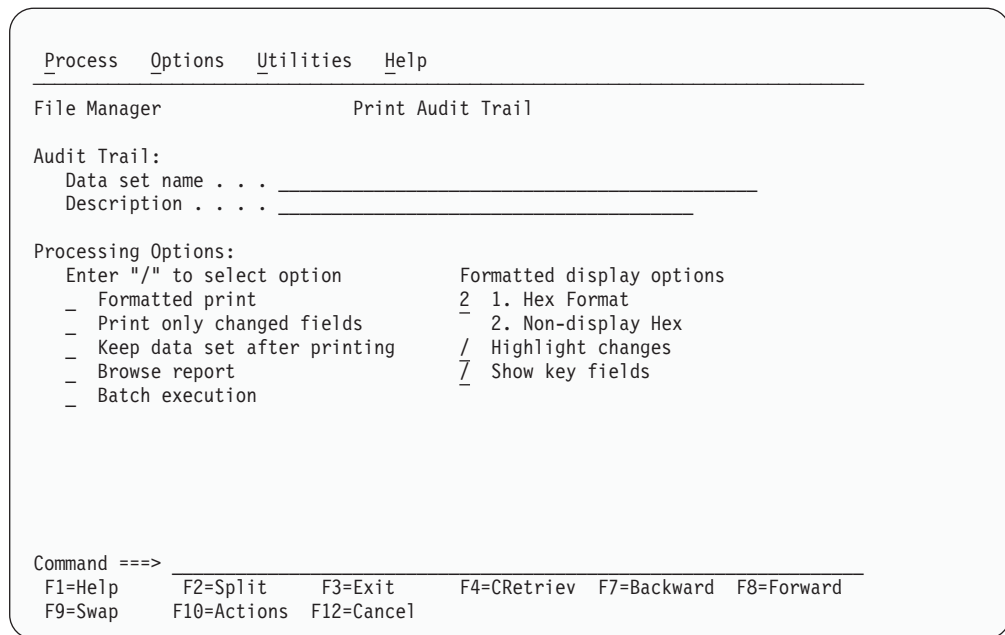


Figure 3. The Print Audit Trail panel

Add these parameter descriptions after the description for **Batch execution**:

Hex Format

To produce an UPDOWN hexadecimal display below the standard field display.

Non-display Hex

To produce an UPDOWN hexadecimal display below the standard field display only for fields that contain non-displayable \ characters.

Highlight changes

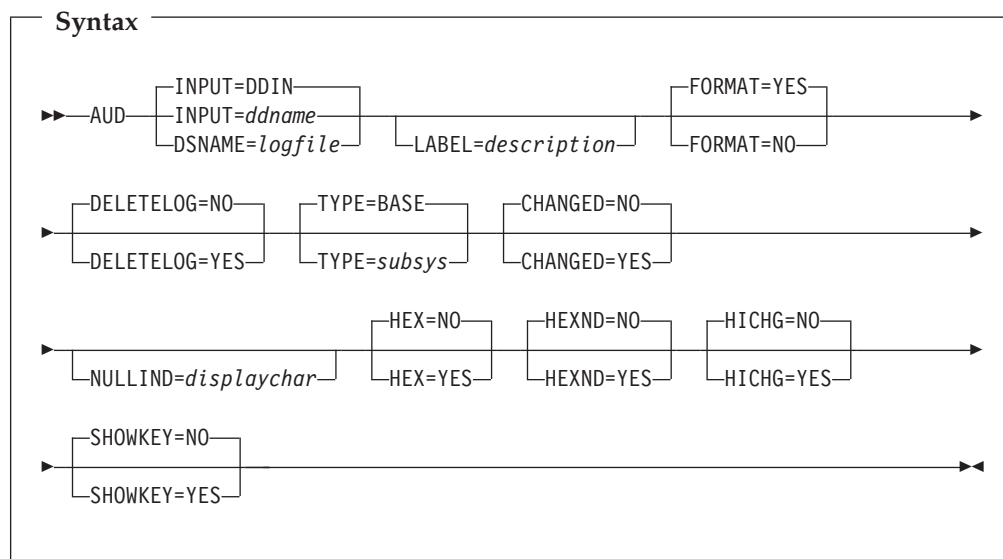
Highlight the changed fields. An asterisk is placed to left of the before data to indicate the field has been changed.

Show key fields

To display key fields even when **Print only changed fields** is selected. A "K" is printed to the left of key field names. For a KSDS data set, a key field is any elementary field that intersects or is contained in the key area.

Chapter 16, "Functions", section "File Manager functions", subsection "AUD (Print Audit Trail Report)"

Replace the syntax diagram with this:



Add these parameter descriptions after the description for NULLIND:

HEX=YES

Produces an UPDOWN hexadecimal display below the standard field display.

HEXND=YES

Produces an UPDOWN hexadecimal display below the standard field display only for fields that contain non displayable characters.

HICHG=YES

Highlights the changed fields. An asterisk is placed to the left of the "before" data to indicate the field has been changed.

SHOWKEY=YES

Displays key fields even when CHANGED=YES has been selected. A "K" is printed to the left of key field names. For a KSDS data set, a key field is any elementary field that intersects, or is contained in, the key area. Note: does not apply to DB2 audit reports.

Changes to the User's Guide and Reference for IMS

Chapter 9, "Panels and fields", subsection "Print Audit Trail panel"

Replace the figure "Print Audit Trail panel" with this:

```

Process  Options  Help
-----
FM/IMS          Print Audit Trail

Audit Trail:
  Data set name . . . 'FMNUSER.IMSAUDIT.D020927.T130548'
  Description . . . . 'EDIT THE SUBURB DATABASE'

Processing Options:
  Enter "/" to select option          Formatted display options
  / Formatted print                    2 1. Hex Format
  _ Print only changed fields          _ 2. Non-display Hex
  Delete data set after printing      _ Highlight changes
  / Browse report                      _ Show key fields
  _ Batch execution

Command ==>
  F1=Help      F2=Split      F3=Exit      F4=CRetriev  F7=Backward  F8=Forward
  F9=Swap      F10=Actions  F12=Cancel

```

Figure 4. Print Audit Trail panel

Add these parameter descriptions after the description for **Batch execution**:

Hex Format

To produce an UPDOWN hexadecimal display below the standard field display.

Non-display Hex

To produce an UPDOWN hexadecimal display below the standard field display only for fields that contain non-displayable \ characters.

Highlight changes

Highlight the changed fields. An asterisk is placed to left of the before data to indicate the field has been changed.

Show key fields

To display key fields even when **Print only changed fields** is selected. A "K" is printed to the left of key field names. For a KSDS data set, a key field is any elementary field that intersects or is contained in the key area.

Chapter 11, "Batch reference", subsection "Print Audit Report (AUD)"

Replace the whole section up to, but excluding, the example with this:

Purpose

Print a formatted or unformatted audit trail report.

Usage Notes

You must specify the name of the audit trail data set from which you want to produce a report.

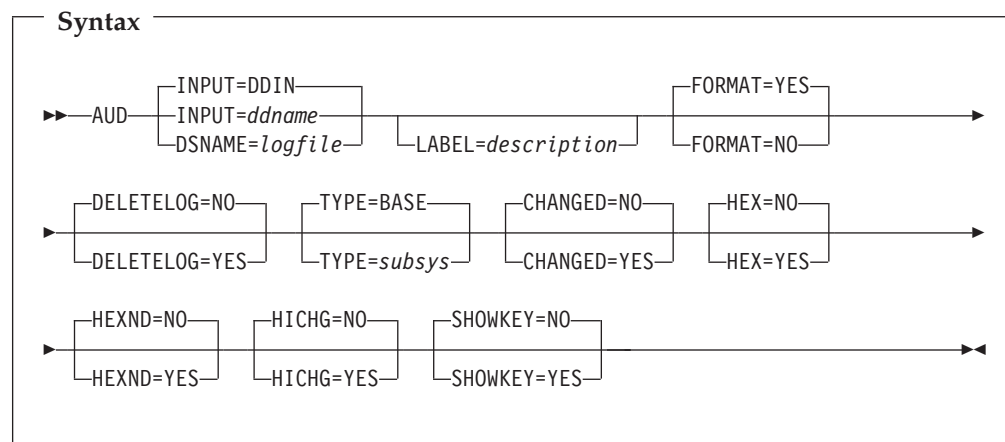
Options

You can choose between printing an unformatted report or, if the audited function used a template, a report formatted according to the template you

used. You can also specify whether or not you want the audit trail data set to be deleted after printing. You can provide a description to help to identify the audit trail report.

Related functions

None.



INPUT=*ddname*

Defines a reference to a DD or TSO ALLOC statement for the input data set or HFS file. The default is DDIN.

DSNAME=*logfile*

Specifies the name of the audit trail log data set.

LABEL=*Description*

Optional identification for the audit trail report. Must be within quotes if it contains blanks.

FORMAT

Determines the formatting of the audit trail report.

YES Default. Report is formatted according to the template used in the audited Edit session.

NO Report is not formatted.

DELETEDLOG

Determines whether or not the audit trail data set is deleted after printing. DELETEDLOG is not allowed for log data sets allocated using a DD statement.

NO Default. The audit trail data set is not deleted.

YES The audit trail data set is deleted after the report is printed.

TYPE Specifies the subsystem used for the audited Edit session. Can be one of:

BASE Default.

IMS

DB2

CHANGED

NO Default. All fields are reported.

YES Only fields that are changed are reported.

HEX=YES

Produces an UPDOWN hexadecimal display below the standard field display.

HEXND=YES

Produces an UPDOWN hexadecimal display below the standard field display only for fields that contain non displayable characters.

HICHG=YES

Highlights the changed fields. An asterisk is placed to the left of the "before" data to indicate the field has been changed.

SHOWKEY=YES

Displays key fields even when CHANGED=YES has been selected. A "K" is printed to the left of key field names. For a KSDS data set, a key field is any elementary field that intersects, or is contained in, the key area. Note: does not apply to DB2 audit reports.

Changes to the Customization Guide

Chapter 5, "Customizing the File Manager audit facility", subsection "Setting the appropriate options to produce an audit trail"

Insert the following bullet point after the first bullet point:

- If you want to produce an audit trail and report on the changes made at conclusion of an edit function then specify AUDITLOG=DEMAND. The job submitted will be determined by skeleton member FMN0FTAD found in FMN.SFMNSLIB. You should customize the job card and JCL in this skeleton to specify the reporting options you require. (For FM/CICS, customize skeleton FMN3FTAD.) The reporting options in the skeleton are described in the File Manager User's Guide and Reference, SC19-2495.

Chapter 14, "Customizing the FM/DB2 audit facility", subsection "Determining if an audit trail is to be produced"

Insert the following bullet point as the last bullet point:

- Demand; an audit trail will be produced for each user, regardless of whether or not the user wants to record audit information and an audit report job will be submitted at conclusion of the edit function or when you change SSIDs. Specify AUDIT=(DEMAND,...) The job submitted will be determined by skeleton member FMN2FTAD found in FMN.SFMNSLIB. You should customize the job card and JCL to specify the reporting options you require. The reporting options in the skeleton are described in the File Manager User's Guide and Reference, SC19-2495.

Chapter 19, "Customizing FM/IMS"

In figure 12, describing the sample source for the FMN1POPT module, add the following option with default values and description, to the FMN1POPD macro section:

```
IMSAUDLG=N,                Audit logging enforced? Y/N/D X
```

Also, add the following option with default values and description, to the FMN1POPI macro section in this figure:

```
IMSAUDLG=,                Audit logging enforced? Y/N/D X
```

Chapter 20, "Customizing the FM/IMS security environment", subsection, "Security Exit Parameters", Table 30. Parameters - Exit Type A

Add the following item to the description of the CREATE field:

D See IMSAUDLG=D description.

Chapter 21, "Customizing the FM/IMS audit facility"

Add the following bullet point to the list in the third paragraph:

- Force audit logging during Edit and at conclusion of the Edit session submit an audit report job to report on the changes. The job submitted will be determined by skeleton member FMN1FTAD found in FMN.SFMNSLIB. You should customize the job card and JCL to specify the reporting options you require. The reporting options in the skeleton are described in the File Manager User's Guide and Reference, SC19-2495. Also refer to IMSAUDLG option in Appendix C.

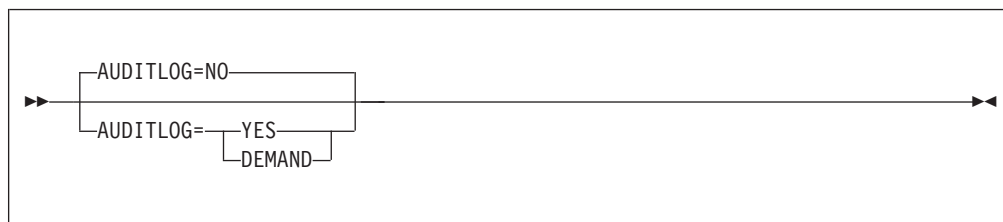
Chapter 29, "Customizing the FM/CICS audit facility"

Add the following as the fourth paragraph:

You can force the production of an audit trail, and report on the changes made, by specifying AUDITLOG=DEMAND in FMN3POPT. The job submitted will be determined by skeleton member FMN3FTAD found in FMN.SFMNSLIB. You should customize the job card and JCL in this skeleton to specify the reporting options you require. The reporting options in the skeleton are described in the File Manager User's Guide and Reference, SC19-2495.

Appendix A, "File Manager options"

Replace the AUDITLOG syntax diagram with the following diagram:



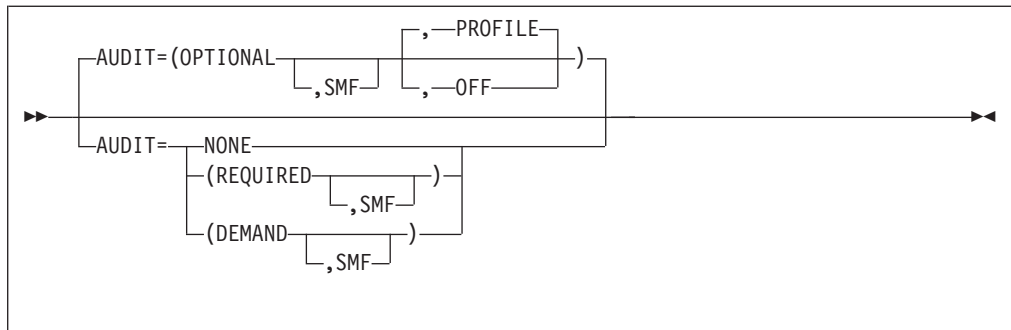
Add the following to the explanation of the AUDITLOG parameters:

DEMAND

Audit trail logging is mandatory and a job described by skeleton FMN0FTAD (FMN3FTAD for FM/CICS) will be submitted to report on the changes made by a user when exiting the edit function. The reporting options in the skeleton are described in the File Manager User's Guide and Reference, SC19-2495.

Appendix B, "FM/DB2 options"

Replace the AUDIT syntax diagram with the following diagram:



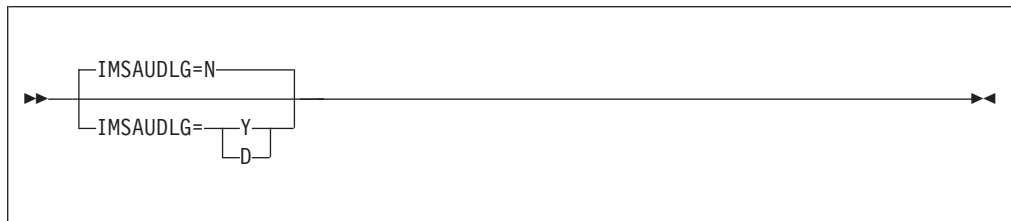
Add the following to the explanation of the AUDIT parameters:

DEMAND

an audit trail is produced, regardless of the setting of Create an audit trail in the editor options. An audit report job will be submitted at the conclusion of an edit function. If **AUDIT=(DEMAND,SMF)** is specified, then the audit trail is also written to SMF.

Appendix C, "FM/IMS options"

Replace the IMSAUDLG syntax diagram with the following diagram:



Add the following to the explanation of the IMSAUDLG parameters:

- D** audit logging is enforced during Edit and at conclusion of the Edit session an audit report job will be submitted to report on the changes. This job can be customized by changing member FMN1FTAD from FMN.SFMNSLIB to specify job card and the reporting options you require. The reporting options in the skeleton are described in the File Manager User's Guide and Reference, SC19-2495.

Part 2. General documentation changes

Customization Guide (SC19-2494-00)	65	Change #8: June 2010	67
December 2008	65	Chapter 16, "Functions"	67
Part 1. Customizing File Manager	65	Change #7: May 2010	67
Chapter 1, "Preparing to customize File Manager"	65	Change #6: May 2010	67
Chapter 3, "Customizing File Manager"	65	Chapter 16, "Functions"	67
Chapter 7, "Customizing File Manager to use library management system libraries".	65	Change #5: April 2010	67
Part 4. Customizing File Manager CICS Component	65	Change #4: August 2009	67
Chapter 24, "Preparing to customize FM/CICS".	65	Change #3: May 2009	68
Appendix A. File Manager Options	65	Change #2: December 2008	68
LMS option	65	Change #1: November 2008	68
LMSUBSYS option	66		
June 2009	66	User's Guide and Reference for DB2 Data (SC19-2496-00)	71
Part 1. Customizing File Manager	66	Change #1 November 2010	71
Chapter 2, "Customizing the operating environment for File Manager".	66		
User's Guide and Reference (SC19-2495-00)	67	User's Guide and Reference for IMS Data (SC19-2497-00)	73
		User's Guide and Reference for CICS (SC19-2498-00)	75

This section describes enhancements and updates in the documentation for File Manager for z/OS Version 9 Release 1. These changes are not associated with individual APAR or PTF numbers, as they do not require the application of any code updates.

The changes are grouped by manual and listed within each section in reverse date order. That is, the most recent documentation change appears at the beginning of each manual section.

Customization Guide (SC19-2494-00)

December 2008

Part 1. Customizing File Manager

Chapter 1, "Preparing to customize File Manager"

In Table 1, add a new step (after step 8), "Customize to use HLASM copybooks." pointing to page 13.

Chapter 3, "Customizing File Manager"

In the section, "Changing the JCL skeleton for batch mode", on page 22, second complete paragraph, change the beginning of the first sentence to read:

If you plan to use File Manager to access COBOL copybooks, PL/I include books or HLASM copybooks in library management system (LMS) libraries

Chapter 7, "Customizing File Manager to use library management system libraries"

In the fifth paragraph on page 53, change the beginning of the first sentence to read:

If you plan to use File Manager to access COBOL copybooks, PL/I include books or HLASM copybooks stored in LMS libraries,

In the section, "Accessing source code in CA-Panvalet libraries", on page 53, replace the first sentence in the first paragraph with:

File Manager provides an interface to CA-Panvalet, enabling you to use File Manager to work with COBOL copybooks, PL/I include books and HLASM copybooks stored in CA-Panvalet libraries.

In the section, "Capabilities provided by File Manager via FMNCRAEX", replace the first sentence of point #1 with the following:

1. The ability to extract LMS files containing COBOL copybooks, PL/I include books and High Level Assembler copybooks.

Part 4. Customizing File Manager CICS Component

Chapter 24, "Preparing to customize FM/CICS"

In Table 35, add a new step (after step 10), "Customize to process HLASM copybooks." pointing to page 195.

Appendix A. File Manager Options

LMS option

Change the description to read:

specifies whether or not File Manager will use COBOL copybooks, PL/I include books or HLASM copybooks stored in CA-Panvalet libraries or other Library Management System libraries.

Change the first sentence in each of the descriptions of "PANVALET", "USERLMS", and "(PANVALET,USERLMS) or (USERLMS,PANVALET)" to read:

File Manager will use copybooks or include books stored in

LMSUBSYS option

Change the first paragraph of the description to read:

specifies that File Manager will, when accessing COBOL copybooks, PL/I include books, or HLASM copybooks, attempt the access via an I/O subsystem using the DFSMSdfp SUBSYS=xxxx allocation parameter.

Change the third paragraph of the description to read:

This facility is used to access copybooks or include books in OEM library management system data sets.

June 2009

Part 1. Customizing File Manager

Chapter 2, "Customizing the operating environment for File Manager"

In section "Enabling File Manager to work with certain products", subsection "Enabling WebSphere MQ support", in the paragraph:

"For File Manager base function, the WebSphere MQ load libraries SCSQANLE, SCSQAUTH and SCSQLOAD must be made available to the TSO user, either in the linklist, or as part of the STEPLIB in the TSO procedure, or as part of the ISPLLIB concatenation."

remove the phrase:

", or as part of the ISPLLIB concatenation"

User's Guide and Reference (SC19-2495-00)

Change #8: June 2010

Chapter 16, "Functions"

Subsection: "File Manager functions"

- For the SCS function:
 - In the description for SORTBY, replace the list of possible values with:

ALLOC	Sorts by allocated space
DATE	Sorts by creation date
DSORG	Sorts by organisation
FREESP	Sorts by free space
LRECL	Sorts by logical record length
NAME	Sorts by data set name (the default)
RECFM	Sorts by record format
 - In the description for DATEFORM, replace the first value ("YYDD") with "YYDDD".

Change #7: May 2010

Chapter 11, "Using UNIX System Services and the Hierarchical File System", section "Specifying an HFS file":

- Remove the entire subsection, "Using wildcard characters to specify file names".

Change #6: May 2010

Chapter 16, "Functions"

Subsection: "File Manager functions"

- For the FCH function, remove the //FMNOUT DD SYSOUT=* statement from the sample JCL.

Change #5: April 2010

Chapter 16, "Functions", section "File Manager functions", subsection "DSM (Data Set Compare)"

- In the last bullet point in the **Purpose** section, remove the sentence in brackets so that the point reads:
 - Create output data sets containing records identified as inserted, deleted, old and new changed records, and old and new matched records.

Change #4: August 2009

Chapter 6, "Managing data sets", section "Finding and changing data in multiple PDS members":

- **Subsection "Using the CHANGE command in the Find/Change Utility"**
After Figure 68, "Find/Change Utility: example of results from CHANGE command", add:

Note: On the Find/Change Utility report, the record number can have one of these prefixes:

K Indicates the change involved a key.

X Indicates that although it was eligible to be changed, the change could not be performed due to record length restrictions.

KX Indicates both of the above.

- **Subsection "Changing data with a REXX procedure"**

Change the sentence:

"The Find/Change Utility report prefixes each record number selected by the REXX procedure with an "S" and each record changed by the REXX procedure with a "C"."

to:

"The Find/Change Utility report prefixes each record number *selected* by the REXX procedure with an "S", each record *changed* by the REXX procedure with a "C", and each record *added* by the REXX procedure with a "+"."

Change #3: May 2009

Chapter 6, "Managing data sets", section "Comparing data sets", subsection "Record synchronization"

- Change the sentence "Any records that do not match are regarded as paired insertions and deletions."

to:

"Non-matching records are regarded as changed records."

Change #2: December 2008

Chapter 16, "Functions", section "External REXX functions"

- In the list of functions at the beginning of the section:
 - In the description for RSTR_OUT, add the sentence:
"(Can be used in FASTREXX condition expressions.)"
 - In the description for SAVE_OUT, add the sentence:
"(Can be used in FASTREXX condition expressions.)"
- In the subsection, "RSTR_OUT", after the syntax diagram add the sentence:
"(Can be used in FASTREXX condition expressions.)"
- In the subsection, "SAVE_OUT", after the syntax diagram add the sentence:
"(Can be used in FASTREXX condition expressions.)"

Change #1: November 2008

Chapter 7, "Using File Manager utilities"

In the subsection, "Working with WebSphere MQ", add the following at the end of step 3.b:

File Manager supplies a sample copybook and template in the sample library (SFMNSAM1). The member FMNPMQMD is a PL/I copybook extracted from the Websphere MQ supplied copybook which describes common message header descriptions. This sample may be extended to include your application data layout definition by adding the appropriate PL/I statements, or a %INCLUDE statement.

Likewise, the member FMNMQMD is a COBOL copybook which describes common message header descriptions.

The member FMNTPMQD is the template version of the PL/I copybook and the member FMNTCMQD is the template version of the COBOL copybook.

Both templates include the identification criteria used to identify each message header type as shown in Table 7. These are easily determined by inspecting the copybook.

Table 7. Criteria for identifying message header types

Layout name	PL/I template: add this ID criteria	COBOL template: add this ID criteria
MQCIH	#2=='CIH '	#3=='CIH '
MQDH	#2=='DH '	#3=='DH '
MQDLH	#2=='DLH '	#3=='DLH '
MQIIH	#2=='IIH '	#3=='IIH '
MQMDE	#2=='MDE '	#3=='MDE '
MQMD1	#2=='MD ' & #3=1	#3=='MD ' & #4=1
MQMD2	#2=='MD ' & #3=2	#3=='MD ' & #4=2
MQRFH1	#2=='RFH ' & #3=1	#3=='RFH ' & #4=1
MQRFH2	#2=='RFH ' & #3=2	#3=='RFH ' & #4=2
MQRMH	#2=='RMH '	#3=='RMH '
MQTM	#2=='TM '	#3=='TM '
MQTMC2	#2=='TMC '	#3=='TMC '
MQWIH	#2=='WIH '	#3=='WIH '
MQXQH	#2=='XQH '	#3=='XQH '

User's Guide and Reference for DB2 Data (SC19-2496-00)

Change #1 November 2010

In Chapter 15, "FM/DB2 panels and fields", subsection "Basic SELECT Prototyping panel", replace the second panel with:

<u>P</u> rocess	<u>O</u> ptions	<u>U</u> tilities	<u>H</u> elp		
FM/DB2 (DFA2)		Basic SELECT Prototyping	Row 1 of 19		
SELECT ? FROM ? WHERE ? ORDER BY ?					
Row count	<u>ALL</u>	Number of rows to display			
Select columns (S/A/D) or enter predicates to build the SELECT statement:					
S	LOp (Tab Column Name	Data Type(length)	Op Value)		
-	- #1 EMPNO	CHAR(6)	-		
-	- #1 FIRSTNME	VARCHAR(12)	-		
-	- #1 MIDINIT	CHAR(1)	-		
-	- #1 LASTNAME	VARCHAR(15)	-		
-	- #1 WORKDEPT	CHAR(3)	-		
-	- #1 PHONENO	CHAR(4)	-		
-	- #1 HIREDATE	DATE(4)	-		
-	- #1 JOB	CHAR(8)	-		
-	- #1 EDLEVEL	SMALLINT(2)	-		
-	- #1 SEX	CHAR(1)	-		
-	- #1 BIRTHDATE	DATE(4)	-		
-	- #1 SALARY	DECIMAL(9,2)	-		
-	- #1 BONUS	DECIMAL(9,2)	-		
-	- #1 COMM	DECIMAL(9,2)	-		
-	- #2 DEPTNO	CHAR(3)	-		
-	- #2 DEPTNAME	VARCHAR(36)	-		
-	- #2 MGRNO	CHAR(6)	-		
-	- #2 ADMRDEPT	CHAR(3)	-		
-	- #2 LOCATION	CHAR(16)	-		
	**** End of data ****				
Command ==>			Scroll PAGE		
F1=Help	F2=Split	F3=Exit	F4=Expand	F6=Execute	F7=Backward
F8=Forward	F9=Swap	F10=Left	F11=Right	F12=Cancel	

User's Guide and Reference for IMS Data (SC19-2497-00)

There are no general documentation changes.

User's Guide and Reference for CICS (SC19-2498-00)

There are no general documentation changes.

Part 3. Appendixes

Index

E

editor session
UNICODE data 42

P

PK75870
UK42251 55
PK77613
UK44837 45
PK79535
UK44322 53
PK83865
UK49461, UK49462, UK49463,
UK49464, UK49465, UK49466,
UK49467 37
PK84077
UK48204, UK48205, UK48206,
UK48207, UK48239, UK48243,
UK48254 39
PK84110
UK47675, UK47676, UK47684,
UK47687, UK47708, UK47710,
UK47712 41
PK85062
UK49461, UK49462, UK49463,
UK49464, UK49465, UK49466,
UK49467 38
PK90394
UK47675, UK47676, UK47684,
UK47687, UK47708, UK47710,
UK47712 43
PK93460
UK52267, UK52268, UK52269,
UK52270, UK52271, UK52272,
UK52273, UK52274, UK52275,
UK52276, UK52277, UK52278 20
PK94449
UK52267, UK52268, UK52269,
UK52270, UK52271, UK52272,
UK52273, UK52274, UK52275,
UK52276, UK52277, UK52278 30
PK95812
UK52267, UK52268, UK52269,
UK52270, UK52271, UK52272,
UK52273, UK52274, UK52275,
UK52276, UK52277, UK52278 31
PK95961
UKnnnn6, UK24879, UK24880,
UK24893, UK24898, UK24899,
UK24906 16
PM02105
UK54336, UK54337, UK54340,
UK54342, UK54343, UK54344 15
PM08924
UK58321, UK58323, UK58349,
UK58351, UK58361, UK58369 13

PM20322
UK62005, UK62016, UK62018,
UK62049, UK62129, UK62130,
UK62136, UK62138, UK62136 9

PM20660
UK63245, UK63246, UK63247,
UK63251, UK63252 5

PM22660
UK63245, UK63246, UK63247,
UK63251, UK63252 6

PM28967
UK64397, UK64398, UK64399,
UK64400 3

U

UK21782
PK95961 16
UK21784
PK95961 16
UK21785
PK95961 16
UK21789
PK95961 16
UK42251
PK75870 55
UK42251, UK42253, UK42277, UK42278,
UK42279, UK42284, UK42317, UK42329,
UK42367, UK42381, UK42382, UK42383
PK75870 55
UK42253
PK75870 55
UK42277
PK75870 55
UK42278
PK75870 55
UK42279
PK75870 55
UK42284
PK75870 55
UK42317
PK75870 55
UK42329
PK75870 55
UK42367
PK75870 55
UK42381
PK75870 55
UK42382
PK75870 55
UK42383
PK75870 55
UK44322
PK79535 53
UK44837
PK77613 45
UK47675
PK84110 41
PK90394 43

UK47675, UK47676, UK47684, UK47687,
UK47708, UK47710, UK47712
PK84110 41
PK90394 41
UK47676
PK84110 41
PK90394 43
UK47684
PK84110 41
PK90394 43
UK47687
PK84110 41
PK90394 43
UK47708
PK84110 41
PK90394 43
UK47710
PK84110 41
PK90394 43
UK47712
PK84110 41
PK90394 43
UK48204
PK84077 39
UK48204, UK48205, UK48206, UK48207,
UK48239, UK48243, UK48254
PK84077 39
UK48205
PK84077 39
UK48206
PK84077 39
UK48207
PK84077 39
UK48239
PK84077 39
UK48243
PK84077 39
UK48254
PK84077 39
UK49461
PK83865 37
PK85062 38
UK49461, UK49462, UK49463, UK49464,
UK49465, UK49466, UK49467
PK83865 37
UK49462
PK83865 37
PK85062 38
UK49463
PK83865 37
PK85062 38
UK49464
PK83865 37
PK85062 38
UK49465
PK83865 37
PK85062 38
UK49466
PK83865 37
PK85062 38

UK49467		UK54343	
PK83865	37	PM02105	15
PK85062	38	UK54344	
UK52267		PM02105	15
PK93460	20	UK58321	
PK94449	30	PM08924	13
PK95812	31	UK58321, UK08097, UK08098, UK08099	
UK52267, UK52268, UK52269, UK52270,		PM08924	13
UK52271, UK52272, UK52273, UK52274,		UK58323	
UK52275, UK52276, UK52277, UK52278		PM08924	13
PK95812	19	UK58349	
UK52268		PM08924	13
PK93460	20	UK58351	
PK94449	30	PM08924	13
PK95812	31	UK58361	
UK52269		PM08924	13
PK93460	20	UK58369	
PK94449	30	PM08924	13
PK95812	31	UK62005	
UK52270		PM20322	9
PK93460	20	UK62005, UK62016, UK62018, UK62049,	
PK94449	30	UK62129, UK62130, UK62136, UK62138,	
PK95812	31	UK62144	
UK52271		PM20322	9
PK93460	20	UK62016	
PK94449	30	PM20322	9
PK95812	31	UK62018	
UK52272		PM20322	9
PK93460	20	UK62049	
PK94449	30	PM20322	9
PK95812	31	UK62129	
UK52273		PM20322	9
PK93460	20	UK62130	
PK94449	30	PM20322	9
PK95812	31	UK62136	
UK52274		PM20322	9
PK93460	20	UK63245	
PK94449	30	PM20660	5
PK95812	31	PM22660	6
UK52275		UK63245, UK63246, UK63247, UK63251,	
PK93460	20	UK63252	
PK94449	30	PM20660	5
PK95812	31	PM22660	5
UK52276		UK63246	
PK93460	20	PM20660	5
PK94449	30	PM22660	6
PK95812	31	UK63247	
UK52277		PM20660	5
PK93460	20	PM22660	6
PK94449	30	UK63251	
PK95812	31	PM20660	5
UK52278		PM22660	6
PK93460	20	UK63252	
PK94449	30	PM20660	5
PK95812	31	PM22660	6
UK54336		UK64397	
PM02105	15	PM28967	3
UK54336, UK54337, UK54340, UK54342,		UK64397, UK64398, UK64399, UK64400	
UK54343, UK54344		PM28967	3
PM02105	15	UK64398	
UK54337		PM28967	3
PM02105	15	UK64399	
UK54340		PM28967	3
PM02105	15	UK64400	
UK54342		PM28967	3
PM02105	15	UNICODE data	42



Printed in USA

Spine information:



File Manager for z/OS V9R1

Addenda to V9R1 User's Guides and Customization
Guide