# PDSUPDTE

PDSUPDTE is a batch card image tailoring procedure and is designed to change selected fields in JCL and control statements contained in libraries (Partitioned Datasets). Control statements can be entered either through SYSIN or through the operator's console. PDSUPDTE executes as a batch job with DD statements defining the libraries to be updated and control statements defining the modifications to be made to these libraries. The libraries are updated in place.

A line of data to be updated within designated libraries is defined as the input line. Changes to the input lines are defined by specifying a string of characters, called STRING1, with which to scan the input line and, if found within the input line, to be replaced by another string of characters, called STRING2.

1. The string to be replaced, STRING1, must be totally contained in columns 1 through 71, inclusive. Thus, the continuation column, column 72, and the sequence number fields, columns 73 through 80, are not affected by any updates.

2. When the equivalent of STRING1 is encountered in the input line, that set of equivalent characters must also be properly delimited on the left and on the right within the input line. If STRING1 is found in the first position of the input line, it is considered as delimited on the left. Likewise, if STRING1 is found at the end of the input line (ending in position 71), it is considered as delimited on the right. Elsewhere in the input line, one of the following characters must be to the left and to the right: ' , . / < > ? " : ; ^ + _ ( ) * % × = and blank. Also, '.V' is considered as a special two character delimiter.

3. When STRING2 is longer then STRING1, blanks are deleted from pools of two or more blanks following STRING1 within the line, up to and including column 71. Thus for a field containing just one blank, that blank would not be deleted. If there is no place to delete blanks, an error message is issued, the line is not updated, and processing continues with the next line.

4. When the length of STRING2 is less than that of STRING1, blanks are inserted after the next pool of two blanks following STRING1.

5. Updates to a line occur in the order that the control statements were entered. Each control statement processes the line in its entirety, before the next control statement processes the line.

# JCL FOR PDSUPDTE

Below is a list of the JCL requirements necessary to execute PDSUPDTE as a batch job.

1. Execute PGM=PDSUPDTE.

2. The PARM field of the EXEC statement can specify PARM=CHECK, PARM=UPDATE, PARM=INSTALL, or no parameter field, in which case PARM=CHECK is defaulted. PARM=UPDATE or PARM=INSTALL is required to cause the actual update to be done. PARM=CHECK, the default, does everything that PARM=UPDATE does except it does not write the updated record back to the data set. This can be used for verification before the updates are done without affecting the status of the data sets. PARM=INSTALL is used during system load down and requires special DD names in the job (see 6 below). An invalid parameter field produces an error message, and processing terminates.

3. A SYSPRINT DD statement is required. The SYSPRINT file is used to log all input control statements, output messages, and updates. Its DCB attributes are RECFM=FBA, LRECL=121. The default block size is 121. If the SYSPRINT DD statement is missing, an error message is issued to the console and processing terminates.

4. All libraries to be considered for update must be defined in the JCL by DD statements, each of whose DDNAME must be unique and the first character of which must be a commercial (@). For example:

```
//@JCLLIB DD DSN=IPO1.JCLLIB,DISP=SHR
```

   A library to be updated must be a partitioned data set (DSORG=PO), RECFM=F or FB, LRECL=80, and it must be allocated on a direct access volume. If a data set fails to meet these requirements a warning message is issued and processing continues with the next library. For partitioned data sets, if no member is specified for the data set on the DD statement, then all members of the library are processed by PDSUPDTE. Multiple libraries are allowed in one execution by specifying additional DD statements to define the additional libraries.

5. Control statements for PDSUPDTE can be entered through file name SYSIN, whose DCB attributes are RECFM=F or FB and LRECL=80. If the SYSIN DD statement is not present and PARM=INSTALL is not specified, then the operator is prompted for input through the operator's console via a WTOR sequence. The SYSIN DD statement is not used when PARM=INSTALL is specified (see 6 below).

6. When PARM=INSTALL is specified, special DD statements whose DD names begin with the letter "Z" are required. The DD name of the form "Zvvvvvv" is specified where "vvvvvv" is the volume serial of the device to be changed. For example:

```
//ZIPORES DD UNIT=3350,VOL=SER=MVSRES,DISP=OLD
```

will generate the following PDSUPDTE control statements:

```
IPORES<MVSRES<< CHANGE VOLUME SERIAL
3330-1<3350<IPORES< CHANGE DEVICE TYPE
```

using the information contained in the DD statement. These two control statements are generated for each special "Z" DD statement present in the step.

# PDSUPDTE CONTROL STATEMENTS

There are four types of control statements associated with PDSUPDTE: the basic control statement, the extended control statement, the END statement, and the comment statement. The syntax of the basic control statement is as follows:

```
STRING1<STRING2< optional comments
```

where STRING1 and STRING2 are as previously defined. The 'less than' symbol (<) is used to delineate the strings. The basic form causes all delineated occurrences of STRING1 on all lines of the libraries to be replaced by STRING2. An example follows:

```
UNIT=3330-1,VOL=SER=IPOLIB /*LIB*/ input line1 before update
UNIT=3330-1,VOL=SER=IPORES /*RES*/ input line2 before update


3330-1<3350< first control statement
IPOLIB<IPORES< second control statement

UNIT=3350,VOL=SER=IPORES /*LIB*/ resultant line1 after update
UNIT=3350,VOL=SER=IPORES /*RES*/ resultant line2 after update
```

The syntax of the extended form is as follows:

```
STRING1<STRING2<STRING3< optional comments
```

where STRING1 and STRING2 are as previously defined and STRING3 is used to limit the set of lines to be updated. STRING3 must appear on the original line, as it was before any updates, before STRING1 can be replaced by STRING2. All 80 columns are searched for STRING3. When STRING3 is used in the members of the install process jobs, it is offset with the characters /* - */. An example follows:

```
UNIT=3330-1,VOL=SER=IPOLIB /*LIB*/ input line1 before update
UNIT=3330-1,VOL=SER=IPORES /*RES*/ input line2 before update

3330-1<3350</*RES*/< first control statement
IPORES<SYSRES</*RES*/< second control statement

UNIT=3330-1,VOL=SER=IPOLIB /*LIB*/ resultant line1 after update
UNIT=3350,VOL=SER=SYSRES /*RES*/ resultant line2 after update
```

The END control statement causes the termination of further input of any control statements. The syntax of the END statement is END coded in columns 1 through 3 and a blank in column 4. The remainder of the statement is not used and can contain comments. This functions the same as a normal end-of-file on SYSIN.

A comment statement consists of a </* in columns 1 through 3. Comment statements cannot be continued. A new comment statement should be used.

Only the first 72 columns of the control statement are used, columns 73 through 80 are ignored. STRING1 begins in column 1. The maximum length for STRING1, STRING2, or STRING3 is 70 characters. The minimum length for STRING1 is 1. The minimum length for STRING2 and STRING3 is zero. If STRING2 is null, a length of zero, STRING1 is deleted from the line. If STRING3 is null, the extended form reverts to the basic form.

A control statement can only be continued after a "less than" symbol (<) by placing a dash (-) or a plus (+) immediately following the "less than" symbol. The remainder of the statement is not scanned and can contain comments. When using a dash (-) to continue a statement, the control statement scan resumes in column 1 of the next statement. When using a plus (+) to continue a statement, the control statement scan resumes at the first non-blank character of the next statement. The continuation character, dash or plus, is not considered as part of the control statement.

Comments can be placed after the first blank following the third 'less than' symbol (<) of the control statement or following the continuation character of a continued statement.

For input from the console, a WTOR sequence is used. Up to 72 characters may be entered in a single reply, or continuations may be used by placing a dash (-) as the last character of the response. For console input only, if an invalid control statement is encountered, a message is written to the operator, the statement is ignored, and a prompt for a new control statement is issued. When in console input mode, certain messages are logged to the operator's console as well as to SYSPRINT.

Some examples of the syntax of PDSUPDTE control statements:

```
3330-1<3350<UNIT=< A COMMENT AFTER THE BLANK
```

The above control statement would change all delineated occurrences of the string '3330-1' to '3350' on every line that contained the string 'UNIT=' of every library specified.

Using the same example expanded to 3 lines, demonstrates the use of the plus signs used for continuations and the fact that comments can be placed after the continuation character.

```
3330-1<+ some more comments on continuation statement
3350<+ note the plus continuation causes blanks on next
UNIT=< statement to be ignored
```

# NO UPDATE/RESUME UPDATE FACILITY

There is a NO UPDATE and a RESUME UPDATE facility. For certain members of partitioned data sets, it may be inappropriate for them to be processed by PDSUPDTE in a massive update. By coding a .NU. (No Update) on the first line of the member not to be considered for update, updating is suspended until end-of-file is reached for the member or until a .RU. (Resume Update) is encountered, in which case updating resumes at the next line of the member. This facility can be overridden by specifying the member name in the data set in the DD statement. For example, assume the following lines existed in a member of a partitioned data set being processed by PDSUPDTE:

```
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
 RENAME DSNAME=IPO1.LINKLIB, .NU..RU.
 VOL=3330-1=IPORES,NEWNAME=IPO1.LINKLIB
 ... etc.
```

And, had one of the control statements for PDSUPDTE been as follows:

```
IPO1<MSPIPO<
```

then, the IPO1 on the line containing the .NU. and .RU. would not be changed, but the IPO1 on the following line would be changed. The resulting lines would be as follows:

```
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
 RENAME DSNAME=IPO1.LINKLIB, .NU..RU.
 VOL=3330-1=IPORES,NEWNAME=MSPIPO.LINKLIB
 ... etc.
```

Had the above JCL existed in a data set whose DD statement specifically referenced only the one member, then the resulting lines would be as follows:

```
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
 RENAME DSNAME=MSPIPO.LINKLIB, .NU..RU.
 VOL=3330-1=IPORES,NEWNAME=MSPIPO.LINKLIB
 ... etc.
```

# SAMPLE JCL

Suppose the following JCL existed in member SAMPLE of library IPO1.JCLLIB:

```
//SAMPLE JOB (ACCT#),'PGMRNAME',
// MSGCLASS=A,MSGLEVEL=(1,1),CLASS=A
//* COMMENT IPO1 COMMENT IPO1 TWO BLANKS
//CATLG EXEC PGM=IEHPROGM
//SYSPRINT DD SYSOUT=A
//MOUNT DD UNIT=3330-1,VOL=SER=IPORES,DISP=OLD
//SYSIN DD *
 CATLG DSNAME=IPO1.PROCLIB,VOL=3330-1=IPORES
 CATLG DSNAME=IPO1.LINKLIB,VOL=3330-1=IPORES
/*
```

Also, suppose that the following JCL was used to update IPO1.JCLLIB:

```
//PDSUPDTE JOB (ACCT#),'PGMRNAME',
// MSGCLASS=A,MSGLEVEL=(1,1),CLASS=A
//STEP EXEC PGM=PDSUPDTE,PARM=UPDATE
//SYSPRINT DD SYSOUT=A
//@TEST DD DSN=IPO1.JCLLIB,DISP=SHR
//SYSIN DD *
CLASS=A<CLASS=X<
A<*<SYSOUT=<
IPO1<MSPIPO<
/*
```

Then, member SAMPLE of IPO1.JCLLIB would be changed to the following:

```
//SAMPLE JOB (ACCT#),'PGMRNAME',
// MSGCLASS=A,MSGLEVEL=(1,1),CLASS=X
//* COMMENT MSPIPO COMMENT MSPIPO TWO BLANKS
//CATLG EXEC PGM=IEHPROGM
//SYSPRINT DD SYSOUT=*
//MOUNT DD UNIT=3330-1,VOL=SER=IPORES,DISP=OLD
//SYSIN DD *
 CATLG DSNAME=MSPIPO.PROCLIB,VOL=3330-1=IPORES
 CATLG DSNAME=MSPIPO.LINKLIB,VOL=3330-1=IPORES
/*
```

Some things to notice in this example are:

1. On the JOB statement, MSGCLASS=A did not get changed to MSGCLASS=X because the string 'CLASS=A' is not properly delimited on the left. Also, since string 'SYSOUT=' is not present, the 'A's on this statement did not get changed to '*'s.

2. The comment statement was used to demonstrate how blanks are deleted from a line in order to expand a field. That is, IPO1 was changed to MSPIPO requiring two blanks to be deleted for each substitution.

3. The extended form of the control statement was used to change the SYSOUT class.

# PDSUPDTE MESSAGES

PDS001 * ERROR * SYSPRINT DD MISSING

This message is issued via WTO when no SYSPRINT DD statement is found in the JOB step.
Action: The program terminates with a return code of 16. No processing has taken place.
User Response: Add a SYSPRINT DD statement to the JOB step.

PDS002 * ERROR * INVALID PARAMETER FIELD

This message is issued when an invalid parameter field of the EXEC statement is encountered.
Action: The program terminates with a return code of 16. No processing has taken place.
User Response: Make sure the parameter field is null, or the CHECK or UPDATE keyword is specified.

PDS003 ENTER PDSUPDTE CONTROL STATEMENT OR 'END'

This message is issued during the WTOR control statement input sequence. It is a request for a new control statement or the continuation of a previous statement.
Action: Wait for operator's reply.
User Response: Enter the control statement or 'END'. If 'END' is entered, further input processing is ended.

PDS004 INVALID CONTROL STATEMENT SYNTAX

This message is issued when a control statement's syntax is invalid.
Action: If input is from SYSIN, the remaining control statements are read and validated, after which the program terminates with a return code of 12. If input is via the operator's console, the statement in error is ignored and processing resumes normally.
User Response: Verify the control statement syntax. Refer to section on control statement syntax rules.

PDS005 END OF INPUT CONTROL STATEMENTS

This message is issued when end-of-file is encountered on SYSIN or when an 'END' control statement is encountered.
Action: If all control statements are valid, processing of the libraries begins.
User Response: None.

PDS006 FILE XXXXXXXX INVALID FOR PDSUPDTE

This message is issued when a DD statement is found defining a library to be processed by PDSUPDTE, but failed to meet the requirements of libraries that can be processed by PDSUPDTE. XXXXXXXX is the DD name of the invalid library.
Action: Processing continues with the next library.
User Response: Validate the DSORG, RECFM, LRECL, and device type of the specified library.

PDS007 I/O ERROR . . . SYNAD ERROR MESSAGE . . .

This message is issued when an I/O error occurs on the library being updated. The text of the message is that which is available to a SYNAD error routine.
Action: Processing of this library terminates and processing resumes with the next library.
User Response: Check the data set and device for possible errors.

PDS008 PROCESSING TERMINATED FOR FILE XXXXXXXX

This message is issued when processing has ended prematurely for a given file, where XXXXXXXX is the DD name of the file.
Action: Processing continues with the next library.
User Response: Check previous messages to determine the cause.

PDS009 * ERROR * TOO MANY CONTROL STATEMENTS

This message is issued when too many control statements were entered.
Action: Control statement input ends and processing terminates with no updates done.
User Response: Each control statement requires 8+L1+L2+L3 bytes of storage, where L1, L2, and L3 are the lengths of STRING1, STRING2, and STRING3, respectively. The total of all control statements must be less than 4092 bytes of storage. Rerun the job twice with different control statements.

PDS010 FILE XXXXXXXX HAD NO UPDATES

This message is issued when no updates were made to a library because no text was found that matched any control statements.
Action: A possible return code of 4 with processing continuing with the next library.
User Response: Make sure all control statements specified the correct updates.

PDS011 PDSUPDTE ENDED. CODE=XX

This message is issued during the termination of PDSUPDTE, where XX is the return code.
Action: none
User Response: Check the return code.

PDS012 data set NOT APPLICABLE TO PDSUPDTE

This message is issued when a DD statement was found defining a library to be updated, but the library was not supported by PDSUPDTE.
Action: Processing continues with the next library.
User Response: Make sure the library specified is one supported by PDSUPDTE.

# PDSUPDTE RETURN CODES

0000 All processing completed with no errors or warnings.
User Response: none

0004 Either one or more of the libraries had no updates or one or more of the libraries was invalid to be processed by PDSUPDTE, or both.
User Response: Check messages logged to SYSPRINT file.

0008 At least one line could not be updated because there were no blanks to expand the line.
User Response: Check messages logged to SYSPRINT file to determine the lines that could not be updated.

0012 At least one library had an I/O error.
User Response: Check messages logged to SYSPRINT file to determine where the errors occurred.

0016 Either the SYSPRINT DD statement was missing, an invalid control statement was encountered, or an invalid parameter field was found.
User Response: Check messages logged to SYSPRINT file to determine where the errors occurred.