Utilities for Partitioned Data Sets

EI&O Document ID: D0068
Last Updated: 11/22/2004

This document introduces and explains how to manage Partitioned Data Sets (PDS) used in OS/390 (MVS).

See other DOCWEB documents for information about utilities and procedures for other types of data sets.

Related search topics include JCL, jobs, batch, ISPF, TSO, and utilities.

UF Information Technology

UFIT

2046 NE Waldo Rd, Suite 2100
Gainesville Florida 32609-8942
(352) 392.2061
<editor@cns.ufl.edu>
# Table of Contents

Introduction to Partitioned Data Sets ................................................................. 1  
How to Create a Partitioned Data Set ................................................................. 3  
Parameters ......................................................................................................... 3  
Notes on Allocating Directory Blocks ............................................................ 3  
How to Compare PDS Directories ................................................................. 7  
Types of Comparisons .................................................................................... 7  
Examining PDS Directories and Members with **PDSLIS** ................................ 9  
Job Control Language .................................................................................... 9  
Command Verbs ............................................................................................. 9  
Example ......................................................................................................... 10  
Sample Program Syntax .............................................................................. 10  
NOTES: ......................................................................................................... 11  
How to Reallocate an Existing PDS ............................................................. 13  
How to Delete or Rename a PDS ................................................................. 15  
How to Delete a PDS or Members ............................................................... 15  
How to Rename a PDS or Member ............................................................ 15  
How to Regain Unused Space in a PDS ...................................................... 17  
JCL ............................................................................................................. 17  
Utilities for Partitioned Data Sets: Appendix A - Obsolete Utilities ................ 19  
Obsolete Utilities ......................................................................................... 19  
PDSDIRC ................................................................................................. 19  
PDSRESET .............................................................................................. 19  
PDSSCR ................................................................................................. 19  
Utilities for Partitioned Data Sets iii
List of Figures

1. JCL for the CREATLIB Procedure ................................................................. 4
2. Compile and Link a Fortran Program into a Load Module Library .................. 4
1. Using SUPERC to Compare PDS Directories .................................................. 7
1. JCL for the PDSLIST Status Report. ............................................................... 9
2. Example of Using PDSLIST ........................................................................... 10
3. Sample PDSLIST Program .......................................................................... 10
1. How to Copy a PDS. .................................................................................... 13
1. How to Use the DELETE Command. ............................................................. 15
2. How to Use the RENAME Command. ............................................................ 16
1. JCL for Using the SQUEEZE Procedure ....................................................... 17
A Partitioned Data Set (PDS) is an IBM OS/390 (MVS) file that contains one or more sequential files called members. Members can be used to store various types of information:

- General alphanumeric data, such as the contents of database records and fields.
- Text files, such as paragraphs or chapters in a novel.
- Source files, which are uncompiled job-specific instructions written in a programming language like Fortran, SAS, C, or COBOL.
- Object decks, which are compiled programs, not yet link-edited.
- Macros, a series of computer instructions invoked during the translation of a single coded word.
- Executable programs, stored as machine language, produced by compiling and link-editing a source file. An example would be a subroutine program. A PDS whose members are independently executable is identified in this document as a load module library. The synonyms program library and load library are also used here.
- Batch job instructions, using Job Control Language (JCL).
- TSO CLISTs and REXX execs.

Partitioned data sets are versatile. Information in one PDS member can be accessed quickly, directly, without disturbing adjacent members. Members are located via a directory, where each has a unique entry. At the same time, PDS format allows processing of sequential members in batch operations.

Created PDSs should be compact to save on storage costs and conserve disk space, yet flexible enough to permit growth in the data set. For example, if you don't allocate enough directory blocks when you create a PDS you may have to subsequently create another, larger PDS and transfer all your data to it. Conversely, allocating more directory blocks than will ever be used is wasteful and can result in unnecessary disk storage charges. For more information on directories and other PDS parameters, see Chapter 2 of this document, How to Create a Partitioned Data Set [http://docweb.cns.ufl.edu/docs/d0068b/d0068b.html].

A PDS may be stored on tape, in an unloaded format, or on a direct access (disk drive) volume. However, data within a PDS stored on tape can only be accessed or processed after the PDS has been reloaded onto a disk drive volume.

A PDS should be created with a data set name conforming to EI&O naming conventions, under which it will be cataloged. The names of its individual members must be one to eight characters in length, and unique within that data set. However, after a PDS is created, its members can be assigned one or more aliases, which are alternate names for the same member. Aliases permit access to information under familiar names. For further details on EI&O naming conventions, see EI&O Document D0045 [http://docweb.cns.ufl.edu/docs/d0045/d0045.html].
The **CREATLIB** JCL procedure produces empty, ready-to-use partitioned data sets (PDS). If you accept **CREATLIB**'s default values, it will create a PDS formatted for exclusive use as a load module library. By adding parameters to the default EXEC statement, you can create partitioned data sets customized for other types of data.

**Parameters**

There are nine different parameters associated with **CREATLIB**. The record format, logical record length, and block size parameters specify how data in a PDS will be formatted, or "packaged." The cylinders/tracks, primary/secondary allocations, directory blocks, and volume parameters define space on direct access storage media. The unit parameter specifies a type of input/output device.

Values for these parameters (for uses other than load libraries) depend on the nature of your data, and how it can most efficiently be stored and accessed. Parameter values can be requirements of the program you are using. The list below shows the parameters whose values can be changed on the **CREATLIB EXEC** statement. In the left column are keywords for each parameter. The right hand column defines the keywords and lists the default values for each parameter.

- **RECFM** = Record format. Default is undefined-length (U).
- **LRECL** = Logical record length. Default is 0 (zero).
- **BLKSIZE** = Block size. Default is 15,476.
- **CYLTRK** = Unit of space to be allocated, cylinder (CYL) or track, (TRK). Default is TRK.
- **PRIM** = Primary allocation of cylinder or tracks. Default unit is 5.
- **SEC** = Secondary (overflow) allocations of cylinder or tracks. Default is 1.
- **DIR** = Number of directory blocks. Default is 1.
- **VOL** = An online DASD volume. Default is null.
- **UNIT** = Type of input/output device. Default is SYSDA.

**Notes on Allocating Directory Blocks**

Directory entries for members of an object library require the least space, since the member's name and its relative location in the data set are all that must be stored. Directory entries for load modules created by the linkage editor require more space: The system must include additional information for use at execution, such as the size of the module and the entry point address.

The following guidelines can help you determine the number of directory blocks...
needed for a PDS. The figures are typical of data sets at EI&O.

<table>
<thead>
<tr>
<th>TYPE OF LIBRARY</th>
<th>NUMBER OF ENTRIES THAT WILL FIT IN A DIRECTORY BLOCK</th>
</tr>
</thead>
<tbody>
<tr>
<td>SOURCE</td>
<td>6-16*</td>
</tr>
<tr>
<td>OBJECT</td>
<td>16</td>
</tr>
<tr>
<td>LOAD MODULE</td>
<td>6</td>
</tr>
</tbody>
</table>

* Directory entries for source and macro libraries will vary in size, depending on the method used to build them. Using ISPF statistics will increase the size of entries to maintain update information. Since you cannot get additional directory space once a PDS has been created, we suggest you allocate 10%-15% more directory blocks than you think you need.

If you need further help defining parameters, see EI&O Document D0045 [http://docweb.cns.ufl.edu/docs/d0045/d0045.html] or System 370/390 Job Control Language. See chapter 5 in this document [http://docweb.cns.ufl.edu/docs/d0068e/d0068e.html] for procedures to follow when a PDS directory proves too small.

**Job Control Language**

Figure 1 below shows the JCL used to create a default load module library. Note that other than standard `JOB` statement information, all you need to supply is a data set name in the `EXEC` statement.

**Figure 1. JCL for the CREATLIB Procedure**

```
//jobname JOB (,,time,lines),'your name',CLASS=class
/*ROUTE PRINT node.location
// EXEC CREATLIB,LIBRARY='data.set.name'
```

**Examples.** The JCL example in Figure 2 on the following page shows how to create a load module library using the CREATLIB defaults, and then store a program on it. Our explanation starts with the first `EXEC` statement: The PDS is given a data set name, UF.TEACHER.LIB. The next two lines of JCL instruct the computer to compile and link a Fortran program called BGCLASS then store the resulting executable program in UF.TEACHER.LIB. The last line is where the Fortran language instructions for BGCLASS would begin.

**Figure 2. Compile and Link a Fortran Program into a Load Module Library**

```
//LIB JOB (,,time,lines),'your name',CLASS=class
/*ROUTE PRINT node.location
// EXEC CREATLIB,LIBRARY='UF.TEACHER.LIB'
// EXEC VSFCL,LIBRARY='UF.TEACHER.LIB',PROGRAM=BGCLASS
//FORT.SYSIN DD *
...FORTRAN program ....
```
The **EXEC** statement below creates a PDS formatted for 80-byte records. It can be used to store card-image data, object decks (compiled programs), or source images (in a programming language). When object decks are to be stored in a PDS, **BLKSIZE** must be either 400, 800 or 3200. These are multiples of blocking factors 5, 10, 40 and the 80-byte logical record length.

Note that the parameters shown below are separated by commas but no additional spaces, and are continued on a second line.

```plaintext
// EXEC CREATLIB, LIBRARY='UF.TEACHER.MYLIB', RECFM=FB,
// LRECL=80, BLKSIZE=3200, VOL=volume, UNIT=SYSDA
```

A final **CREATLIB** example is the following **EXEC** statement, which creates a PDS with a primary storage space allocation of 10 cylinders, two secondary cylinders, and 2 directory blocks:

```plaintext
// EXEC CREATLIB, LIBRARY='UF.TEACHER.KOLIB', CYLTRK=CYL,
// PRIM=10, SEC=2, DIR=2
```
You can use SUPERC, which is part of ISPF/PDF, to compare the directories of two partitioned data sets. SUPERC is available interactively under the Utilities menu within ISPF and in OS/390 batch.

SUPERC permits comparisons even when data sets being compared have redundant data such as blank lines, duplicate words, and duplicate characters in binary data.

Types of Comparisons

SUPERC offers four compare levels. They are:

- **File Compare**
  - Summarizes differences between two files being compared.

- **Line Compare**
  - Record oriented. Shows matching, inserted, deleted and reformatted lines. Most useful for comparing lines of program source code. Can be used to help detect regressions and to validate appropriateness of code modifications.

- **Word Compare**
  - Reveals differences in data strings delimited by spaces and punctuation marks, such as commas. Finds matching words, even when they are on different lines. Most useful for comparing two text data sets and tracking revisions in text documents.

- **Byte Compare**
  - Finds differences in bytes. Best for comparing machine-readable and unformatted data.

Output Choices

Two output choices are available: A listing of the results of a search or comparison, and/or a structured data set containing update information.

Process Options

More than 30 process options are available, to specify where and how searches and compares are made, and how output is displayed or used. Different process options are available under each of the comparison levels described above. Note: More than one level of compare may be performed on two data sets. However, results may not be identical because of the methods used for different compare levels.

See the IBM manual ISPF/PDF Guide and Reference, SuperC Reference, for further details on compare levels and more information about SUPERC, including how to search for specific terms within a PDS. Figure 1 shows the JCL needed to run SUPERC in batch. You should replace "data.set.one" and "data.set.two" with the names of the two partitioned data sets you want to compare.

Figure 1. Using SUPERC to Compare PDS Directories
Types of Comparisons

// jobname JOB (,,time,lines),'your name',CLASS=class,REGION=2M
/*ROUTE PRINT node.location
// SUPERC EXEC PGM=ISRSUPC,
// PARM=(DELTAL,LINECMP,'','')
// OLDDD DD DSN=data.set.one,DISP=SHR
// NEWDD DD DSN=data.set.two,DISP=SHR
// OUTDD DD SYSOUT=A
/*
The utility **PDSLIST** prints the contents of a Partitioned Data Set (PDS) and/or generates a PDS directory status report. Print operations and status reports can be customized with optional keywords. The JCL in Figure 2 invokes a directory status report which lists the number of directory blocks allocated, number of blocks actually used, total main members, total number of member aliases, and the average number of entries per directory block.

### Job Control Language

To generate the directory status report described above, invoke the **PDSLIST** cataloged procedure as in the example below. `data.set.name` is the name of the cataloged partitioned data set.

**Figure 1. JCL for the PDSLIST Status Report.**

```
//jobname JOB (,,time,lines),'your name',CLASS=class
/*ROUTE PRINT node.location
// EXEC PDSLIST,DSN='data.set.name'
```

### Command Verbs

Two commands verbs, **LIST** and **PRINT**, are used with **PDSLIST**. Minimum abbreviations are shown below in parentheses.

- **(L) LIST**  
  **LIST** operations produce directory information.

- **(P) PRINT**  
  **PRINT** operations print the contents of individual members or groups of members.

### Optional Keywords

The following are optional keywords. When applicable, minimum abbreviations are shown in parentheses. An equal sign indicates that a dd name, data set name, option or other value must be supplied.

- **CUTOFF=n**  
  Specifies that only the first `n` lines of each member selected by the PRINT and PREFIX options are to be printed. Useful for browsing through a program library. This keyword does not apply to the LIST option.

- **(DDN) DDNAME=**  
  Identifies the ddname which defines the PDS to be processed. SYSLIB is the default ddname.

- **(DSN) DSNAME=**  
  Identifies the name of the data set. Not necessary if the dsname is already coded in the JCL.

- **(O) OPTIONS=**  
  Identifies the options below chosen for current run
of PDSLIST.

**STATUS** indicates that the status of the data set directory is to be printed. The names of members will not be printed unless you use the PRINT command.

**NAMES** indicates that member names, alias names, and TTR (member addresses) are to be listed.

**ALL** indicates that SSI (System Status Indicator) and stow information are to be printed with the member and alias names.

**VHEX** indicates that member contents are to be printed in both a character and vertical hexadecimal format, rather than in just character format.

**(IND) INDEX=** specifies that the 1-43 character index of all data sets on the volume be processed.

**(PRE) PREFIX=** specifies that the 1-8 character prefix of all members in a PDS be processed.

### Example

In Figure 2, SYSLIB identifies the data set to be processed. "SYSLIB" is the default ddname. The actual ddname can be supplied by SYSIN. SYSIN identifies the control data set (optional).

### Figure 2. Example of Using PDSLIST

```
//jobname JOB (,,time,lines), 'your name', CLASS=class
/*ROUTE PRINT node.location
// EXEC PGM=PDSLIST
// SYSPRINT DD SYSOUT=A
// SYSLIB DD DSN=dsname, DISP=SHR
// SYSIN DD *
LIST DDNAME=ddn, DSNAME=dsn, OPTIONS=(op,op), PREFIX=pre
PRINT
```

### Sample Program Syntax

### Figure 3. Sample PDSLIST Program

```
//jobname JOB (,,time,lines), 'your name', CLASS=class
/*ROUTE PRINT node.location
// EXEC PGM=PDSLIST
// SYSPRINT DD SYSOUT=A
// SYSLIB1 DD DSN=SYS1.USERLIB, DISP=SHR
// SYSLIB2 DD DSN=SYS1.MACLIB, DISP=SHR
// SYSLIB3 DD DSN=SYS1.PROCLIB, DISP=SHR
```
The sample program in Figure 3 will perform the following functions:

- All member names that begin with **IEE** will be listed along with their addresses and alias names for the data set **SYS1.USERLIB**.

- The contents of all members of **SYS1.MACLIB** will be printed.

- The contents of all members whose names begin with **VSF** will be printed along with any directory user data fields present for those members in data set **SYS1.PROCLIB**.

**NOTES:**

- Only **RECFM F** or **FB** card image records are supported by the **PRINT** command.

- An entire disk pack can be searched for a particular member by coding both **INDEX** and **PREFIX** keywords.

- Only columns 1-72 of the input control statements are scanned. Columns 73-80 are ignored but can be used for sequence numbering or other identification. The entire statement, however, is reproduced in the output listing.
Sometimes a PDS has to be copied and reallocated. You would need to do so, for example, if your data set has run out of directory space. You cannot enlarge the directory of an existing PDS. You can, however, reallocate it with a bigger directory. You can also reallocate a PDS to empty it and reuse it.

There are many ways to accomplish this, but you can do it by following these simple steps:

1. First, use `CREATLIB` to create a new PDS. Make sure you give this PDS a unique data set name. Use the `DIR=` keyword in `CREATLIB` to raise or lower the number of directory blocks.

2. Second,
   a. If your goal is to create an empty PDS with the same name as an existing one, delete your original PDS (this assumes you don't wish to keep the data it contained) and then rename your new one to the old name.
   b. To retain the information in your old PDS, copy it to the new PDS, delete the old PDS, then rename the new one to the old name.

See Chapter 2 in this document on how to create a new PDS. The sample JCL in Figure 1 shows how to copy the contents of one PDS into another.

**Figure 1. How to Copy a PDS.**

```
//jobname  JOB (,,time,lines),'your name',CLASS=class,REGION=2M
/*ROUTE PRINT node.location
 // EXEC CREATLIB,...
 // EXEC PGM=IEBCOPY
 //SYSPRINT DD SYSOUT=A
 //SYSUT1 DD DSN=old.dataset.name,DISP=SHR
 //SYSUT2 DD DSN=new.dataset.name,DISP=OLD
/*
```

`IEBCOPY` is an IBM utility for copying partitioned data sets. It is documented in the MVS/DFP Utilities manual.

If you want to delete the old PDS, use the `DELETE` command described in Chapter 6, *How to Delete or Rename a PDS* [???]. We recommend that you do NOT use this command in the same job as the copy process in case the copy process is not successful for some reason. If the copy process fails, you could inadvertently delete your data set before you copy it.

If you want the new data set to have the same name as the old data set, use the `RENAME` command, also described in Chapter 6.
http://docweb.cns.ufl.edu/docs/d0068f/d0068f.html]
This chapter explains how to delete an entire PDS or individual members. It also explains how to rename an entire PDS or individual member, and how to assign an alias to a member.

The easiest way to accomplish either of these tasks is to submit an OS/390 batch job to execute TSO via batch and use the TSO commands **DELETE** or **RENAME**. You can also sign on to TSO and use these commands interactively from the TSO READY prompt.

**How to Delete a PDS or Members**

You use the **DELETE** command to get rid of an individual member within a PDS (retaining the PDS and other members) or to delete the entire data set.

Use the following form of the **DELETE** command to delete an individual member:

```
DELETE 'data.set.name(member)'  
```

For example, to delete a member named PHOTON in a PDS named UF.ENTERPRS.WEAPONS, you would use:

```
DELETE 'UF.ENTERPRS.WEAPONS(PHOTON)'  
```

To delete the entire data set, just specify the data set name on the DELETE command, as in the following example.

```
DELETE 'data.set.name'  
```

The sample batch job in Figure 1 shows how to use the TSO **DELETE** command to 1) delete a specific member (ROSES) in the PDS named UF.GARDEN.FLOWERS; and 2) to delete the entire PDS named UF.GARDEN.WEEDS.

**Figure 1. How to Use the DELETE Command.**

```
// jobname JOB (,,time,lines),'your name',CLASS=class  
/*ROUTE PRINT node.location  
// EXEC TSO  
DELETE 'UF.GARDEN.FLOWERS(ROSES)'  
DELETE 'UF.GARDEN.WEEDS'  
/*
```

The first **DELETE** command will retain the data set and all members, except ROSES, which will be deleted. The second example will delete the entire PDS.

**How to Rename a PDS or Member**

You can use the **RENAME** command both to rename an entire data set and to rename an individual member. You can also use it to create an ALIAS for a member.
To rename an entire PDS, use the following form:

```
RENAME 'old.name' 'new.name'
```

To rename an individual member, use:

```
RENAME 'pds.name(oldname)' 'pds.name(newname)'
```

This form of the RENAME command retains the original PDS name and just renames the specified member.

To create an alias for a member, use:

```
RENAME 'pds.name(oldname)' 'pds.name(newname)' ALIAS
```

The ALIAS option specifies that "newname" is to be an alias, rather than a replacement.

The sample job in Figure 2 shows how to use the RENAME command to rename an entire PDS, rename an individual member from ROSES to TULIPS, and create an alias.

**Figure 2. How to Use the RENAME Command.**

```
//jobname
JOB (,,time,lines),'your name',CLASS=class
/*ROUTE PRINT node.location
// EXEC TSO
RENAME 'UF.GARDEN.SEEDS' 'UF.GARDEN.BULBS'
RENAME 'UF.GARDEN.FLOWERS(ROSES)' 'UF.GARDEN.FLOWERS(TULIPS)'
RENAME 'UF.GARDEN.TREES(PINES)' 'UF.GARDEN.TREES(NEEDLES)' ALIAS
/*
```

The first example renames the entire data set. The second example changes the name of member ROSES to TULIPS. The third example creates an alias called NEEDLES for member PINES.
The SQUEEZE procedure uses IBM's IEBCOPY utility, which compresses members to new contiguous locations so that space previously held by deleted members is combined into a contiguous space. This space can then be used to add new members to the PDS. Optionally, it can be released, in whole or in part, to other users of the disk volume. If you will not be adding new members or updating members, you should release unused space so that you do not get billed for space you do not need.

**JCL**

Use the following JCL to squeeze a partitioned data set and release the unused space:

**Figure 1. JCL for Using the SQUEEZE Procedure**

```jcl
//jobname JOB (,,time,lines),'your name',CLASS=class /*ROUTE PRINT // EXEC SQUEEZE,DSN='data.set.name',RLSE=RLSE,ALLOC=CYL
```

In the example in Figure 1, `data.set.name` is the name of the cataloged data set to be compressed. `RLSE=RLSE` and `ALLOC=CYL` are optional parameters.

When `RLSE=RLSE` is specified on the EXEC statement, OS/390 allocation/de-allocation releases unused space after IEBCOPY has finished. Space thus released becomes available for other users.

Coding `RLSE=RLSE` and `ALLOC=CYL` requests that all unused cylinders be released except for unused space within the last cylinder. This unused space will be retained as part of the data set, which leaves some space for future growth.
Obsolete Utilities

PDSDIRC

The PDSDIRC utility is no longer supported. Please see Chapter 3 in this document for information on how to compare the directories of two partitioned data sets.

PDSRESET

The PDSRESET command no longer works. Please see Chapter 5 in this document for information on how to copy and reallocate a PDS.

PDSSCR

The PDSSCR utility is no longer supported. Please see Chapter 6 in this document for information on how to rename and delete a PDS.