IBM Education Assistance for z/OS V2R1

Item: PARMDD
Element/Component: BCP Scheduler

Agenda

- Trademarks
- Presentation Objectives
- Overview
- Usage & Invocation
- Interactions & Dependencies
- Migration & Coexistence Considerations
- Presentation Summary
- Appendix

Trademarks

Presentation Objectives

The objective of this presentation is to familiarize you with the new PARMDD= keyword on the JCL EXEC statement and to discuss why you might want to use it instead of the existing PARM= keyword.

Overview

- Problem Statement / Need Addressed
  - The parameter string that can be specified on the existing PARM= keyword is limited to no more than 100 characters; many programs have a need for parameter strings that exceed that limit.
  - Control block considerations prevent any significant compatible expansion beyond the 100 character limit.
- Solution
  - The new PARMDD= keyword allows the name of a DD to be specified.
  - The data set associated with the DDname can contain a parameter string of up to 32760 bytes.
- Benefit / Value
  - The PARMDD= capability satisfies a very long-standing user requirement and provides a very flexible means for providing long parameter strings to batch jobstep programs.

Usage & Invocation

- The new PARMDD= keyword is intended to replace usage of the PARM= keyword for batch jobstep programs requiring parameter strings in excess of 100 bytes.
- Use of the PARMDD= keyword is mutually-exclusive with use of the PARM= keyword.
- DD DSN= example:

```
//LONGPARM JOB MSGLEVEL=1

//STEP0001 EXEC PGM=MYPROGRM,PARMDD=PARMINDD

//PARMINDD DD DSN=MYPARMS.TEXT,DISP=SHR
```

- The parameter string that will be passed to the MYPROGRM program is contained in the MYPARMS.TEXT data set.

Usage & Invocation, continued

- The MYPARMS.TEXT data set in the previous example is a Physical Sequential (PS) data set and can have fixed, fixed-block, variable or variable-block record formats.
- Partitioned data set *members* with those record formats are also supported.
- DD DSN= example:

```
//LONGPARM JOB MSGLEVEL=1

//STEP0001 EXEC PGM=MYPROGRM,PARMDD=PARMINDD

//PARMINDD DD DSN=MYPARMS.LIBRARY(PARMS1),DISP=SHR
```

- The parameter string that will be passed to the MYPROGRM program is contained in the MYPARMS.LIBRARY data set PARMS1 member.

Usage & Invocation, continued

- The data set referenced by PARMDD= may also be a JES “instream” data set: DD * or DD DATA
- *The instream data set may contain symbols that can be resolved by JES2 (only JES2 supports symbol substitution).*
- DD * example:

```
//LONGPARM JOB MSGLEVEL=1
//STEP0001 EXEC PGM=MYPROGRM,PARMDD=PARMINDD
//PARMINDD DD *,SYMBOLS=EXECSYS
Input parameters for MYPROGRM running on &SYSNAME
/*
```

- The parameter string that will be passed to the MYPROGRM program is the string: `Input parameters for MYPROGRM running on AQFT`

Usage & Invocation, continued

- The data set referenced by PARMDD= may also be a UNIX System Service file
- DD PATH= example:

```
//LONGPARM JOB MSGLEVEL=1
//STEP0001 EXEC PGM=MYPROGRM,PARMDD=PARMINDD
//PARMINDD DD PATH='/tmp/myparms.txt'
```

- The parameter string that will be passed to the MYPROGRM program is contained in the USS `/SYSTEM/tmp/myparms.txt` file.

Usage & Invocation, continued

- The parameter string can be formed by concatenating data sets, partitioned data set members or USS files (subject to the rules for BSAM *like* data set concatenation)
- DD DSN= example:

```
//LONGPARM JOB MSGLEVEL=1
//STEP0001 EXEC PGM=MYPROGRM,PARMDD=PARMINDD
//PARMINDD DD DSN=MYPARMS.TEXT(PARMS1),DISP=SHR
//         DD DSN=MYPARMS.TEXT(PARMS2),DISP=SHR
```

- The parameter string that will be passed to the MYPROGRM program is formed from the records contained in the two data set members PARMS1 and PARMS2.

Usage & Invocation, continued

- Input records with a fixed record format and records from JES instream data sets are examined for the presence of *sequence numbers*
  - 8 contiguous numeric characters in the last 8 bytes of a record
  - Commonly produced by the TSO ISPF editor
- If sequence numbers are found, the record length is adjusted so that the sequence numbers are not seen in subsequent processing.
- Each input record (of any supported type) is examined for trailing blanks.
- If trailing blanks are found, the record length is adjusted so that the trailing blanks are not seen by subsequent processing
- The parameter string passed to the program is formed by the simple left to right concatenation of the input records after sequence numbers (if any) and trailing blanks (if any) have been eliminated.
- Double ampersands (&&) in the parameter string are converted to single ampersands.

Usage & Invocation, continued

- Example input:

```
Record #1 input,
Record #2 input is &&T=1,
Record #3 is last.
```

- Parameter string passed to the program:

```
Record #1 input,Record #2 input is &T=1,Record #3 is last.
```

Usage & Invocation, continued

- The PARMDD= keyword may be specified within a JCL procedure and overridden when the procedure is expanded

- Sample procedure:

```
//LONGPRMB  PROC
//STEP0001 EXEC PGM=MYPGM,PARMDD=PARMDD01
//OUTPUT   DD SYSOUT=A
//PARMDD01 DD *
PARAMETERS SUPPLIED THROUGH PARMDD01 (DEFAULT)
//PARMDD02 DD *
PARAMETERS SUPPLIED THROUGH PARMDD02 (OVERRIDDEN)
/*
```

- Invocation of the procedure, overriding the PARMDD= specification:

```
//LONGPRMR JOB  MSGLEVEL=1
//STEP034A EXEC PROC=LONGPRMB,PARMDD.STEP0001=PARMDD02
```

Usage & Invocation, continued

- Programs that are expected to receive parameter strings through the PARMDD mechanism must be prepared to receive parameter strings up to 32760 bytes in length.
  - Many older programs *assume* that the parameter string can never exceed 100 bytes because it could never do so in the past!
- <u>*No*</u> change to the previously documented interface!
  - On entry to the program, register 1 points to a one element, fullword parameter list consisting of a 31-bit pointer.
  - The 31-bit pointer points to a halfword length field that is located immediately in front of the parameter string area.
  - The halfword length field contains the length of the following parameter string. The parameter string length can now be up to 32760 bytes.

Interactions & Dependencies

Software Dependencies

- Authorized programs – programs that have specified the AC(1) Binder attribute -- that have been determined to be able to properly receive parameter strings in excess of 100 characters must be re-linked by the Binder with the LONGPARM attribute. Failure to do so will cause a program fetched from an APF-authorized library to be terminated if the parameter string exceeds 100 characters. Programs that gain authorization from a Program Properties Table (PPT) entry are similarly affected.

- Hardware Dependencies
  - None

- Exploiters
  - No explicit exploitation is required for programs that are properly coded to the existing, documented interface
  - Programs will be unaware that parameters are being provided through the PARMDD facilities.
  - System utilities such as IEBCOPY and IEHLIST will work properly with parameters provided through PARMDD input
  - The DD specified by PARMDD= may be used by BPXBATCH instead of the STDPARM DD

Migration & Coexistence Considerations

- The PARM= keyword continues to behave as it did.
- There are no toleration/coexistence APARs/PTFs.
- No migration actions are required. As longer parameter strings come into use, it will be necessary to use the Binder to re-link authorized programs that can handle long parameter strings with the LONGPARM attribute.
- Jobs that use PARMDD facilities must be run on z/OS V2R1 or later level systems. When use of the PARMDD keyword is detected, the job is marked as having to run on a z/OS V2R1 or later level system and will be automatically scheduled to such a system by the JES.

Presentation Summary

- The new PARMDD= keyword on the JCL EXEC statement is an alternative way of providing parameter input to batch jobstep programs
- PARMDD provides a mechanism for inputting very long program parameter strings from a variety of data sets

Appendix

- JCL Reference (SA22-7597)
- JCL User's Guide (SA22-7598)
- z/OS: MVS Programming: Assembler Services Guide (SA22-7605)
- z/OS: MVS System Messages Volume 8 (IEF-IGD) (SA22-7638)