ThruPut Manager®

z/OS 2.n Supplement

MVS solutions inc.

This manual applies to ThruPut Manager® Version 7 Release 1.0 PTF level 7109 and to all further releases until otherwise indicated. This material contains proprietary information on ThruPut Manager. Use of this manual is subject to the terms of the user's software license agreement for ThruPut Manager.

The complete list of manuals relevant to this release is available from the Customer Center on www.mvssol.com, from where any of the manuals may be downloaded.

The Acrobat®PDF Format

This set of manuals has been formatted to make it suitable for soft copy as well as hard copy formats. If reading the manual online, note that:

- Entries in the Table of Contents are hot-linked and can be used for document navigation.
- Other cross-references appear in blue type and are also hot-linked for easy navigation.

For best results, turn on the Bookmarks feature in your Acrobat® PDF reader.

Summary of Changes

V7R1-7109 *(April 2017)*	• No changes.
V7R1-7108 *(February 2017)*	• No changes.
V7R1-7107 *(May 2016)*	• No changes.
V7R1-7106 *(November 2015)*	• Added comparison of z/OS 2.2 Job Execution Control and ThruPut Manager Dependent Job Control and deadline scheduling capabilities.
V7R1-7104 *(July 2015)*	• No changes
V7R1-7101 *(July 2014)*	• This is the base manual for ThruPut Manager Version 7 Release 1.0.

Table of Contents

Introduction: z/OS 2.2

With z/OS 2.2, IBM introduced JES2 Job Execution Control (JEC) and JES2 deadline scheduling.

ThruPut Manager has offered similar facilities since 1995. ThruPut Manager is controlling batch job dependencies using Dependent Job Control (DJC) and is providing deadline scheduling capabilities through Job Timing Services (JTS) and Job Chaining Services (JCS).

Functionally, the dependent job controls provided by IBM z/OS and MVS Solutions ThruPut Manager are quite similar. In both cases, this is a facility that can be used to group a number of jobs and define dependencies between jobs that belong to the group.

The next chapter is a comparison between JES2 Job Execution Control (JEC) and ThruPut Manager Dependent Job Control (DJC) and deadline scheduling capabilities.

It highlights the substantive differences between the two approaches, under the following topics:

- Dynamic Group Definition
- Simple Dependencies
- Conditional Dependencies
- Treatment of Unsatisfied Dependencies
- Group Display Commands
- Scheduling Parameters: **`WITH`**, **`HOLDUNTIL`**, and **`STARTBY`**
- Group Affinity
- Concurrent Jobs

JES2 Job Execution Control Comparison

Dynamic Group Definition

ThruPut Manager

With ThruPut Manager, DJC groups are defined dynamically by JECL statements inserted in the JCL of the jobs that are in the group. The group is defined by the first reference to it by a participating job.

IBM

In contrast, the IBM JEC group or network is predefined by a new construct called a JobGroup. The JobGroup is submitted as a block of JCL that defines the jobs that belong to the group (distinguished by a user chosen JobGroup name) and the dependencies amongst the jobs. The block of JCL containing the definition consists of JES2 Execution Control Statements.

The JCL for the jobs in the group contains a SCHEDULE statement which ties them to the JobGroup by name. The network definition is static and must be 'registered' before any of the jobs are submitted.

Value Added with ThruPut Manager

The DJC group is dynamically defined where as the IBM JEC Group is not.

Simple Dependencies

ThruPut Manager

ThruPut Manager provides **`RUNIF`** and **`ANDIF`** statements. A BEFORE condition is expressed by inserting a **`RUNIF`** in the target job. ThruPut Manager uses the **`ANDIF`** statement when 'and-ing' is desired, and multiple **`RUNIF`** statements when 'or-ing' is desired.

The GROUP parameter is specified on **`RUNIF`**, **`ANDIF`**, **`FLUSHIF`**, and **`CONDIF`**, to refer to jobs from other groups.

IBM

IBM provides BEFORE and AFTER statements to define dependencies where a job must run either before or after another job or jobs in the same group. These statements are effectively 'and-ed'.

Dependencies are restricted to jobs within the group.

Value Added with ThruPut Manager

- Dependencies can be 'and-ed' and 'or-ed'.
- Cross-group dependencies are supported. The dependent jobs can be in a different group, (using the GROUP parameter on **RUNIF**, **ANDIF**, **FLUSHIF**, **CONDIF** statements).

Conditional Dependencies

Both DJC and JEC support conditional dependencies based on Job return codes. The codes can be system abend codes, user abend codes, or numerical return codes.

Value Added with ThruPut Manager

ThruPut Manager goes a bit further allowing the following capabilities:

- Conditional dependencies can refer to jobs in another DJC group.
- An EVENT can be created and SIGNALed via JECL or operator commands. The SIGNAL can also 'post' a return code to the EVENT. Conditional DJC statements can refer to the EVENT the same as they do to Jobs. Events can be referred to in other groups, so, as an example, a step in one DJC group could signal an EVENT causing a job (perhaps in another DJC group) to be started.
- An event may also be SIGNALed whenever a message with specified text is detected in the address space of the job containing the associated **DJC MESSAGE** statement.

Treatment of Unsatisfied Dependencies

ThruPut Manager

A job with dependencies is HELD in the execution queue until the dependencies are satisfied. Since ThruPut Manager analyzes jobs on submission, any errors are found early.

IBM

A job with dependencies is HELD in the SETUP queue until the dependencies are satisfied and then it is submitted. If it is also managed by ThruPut Manager, then analysis is delayed until the hold is resolved.

Value Added with ThruPut Manager

Problems with jobs with dependencies are exposed earlier and this allows them to be dealt with earlier.

Group Display Commands

ThruPut Manager

DJC DISPLAY GROUP command is used to display the group. The **JOBS** and **DETAILS** parameters display the conditions that were specified and whether they were True or False (assuming the condition has been set). For example:

```
/DJC D GROUP TST002 ALLJOBS DETAILS
DTM6420I DJC DISPLAY
Group TST002 Created Apr 13, 0 Jobs HD(15) HC(5)
 JOB05975 TTW0001 Ended RC(1) Apr 13 15:51:59
- has a RUNIF from TTW0004
- has a FLUSHIF from TTW0005
- has a RUNIF from TTW0006
- has a FLUSHIF from TTW0007
- has a FLUSHIF from TTW0009
- has a FLUSHIF from TTW0009
 JOB05976 TTW0004 Ended RC(4) Apr 13 15:52:03
- RUNIF TTW0001 NORMAL  (T)
 JOB05977 TTW0005 Ended RC(5) Apr 13 15:51:56
- FLUSHIF TTW0001 NORMAL  (T)
 JOB05978 TTW0006 Ended RC(6) Apr 13 15:52:03
- RUNIF TTW0001 CODE EQ 1 (T)
 JOB05979 TTW0007 Ended RC(7) Apr 13 15:51:58
- FLUSHIF TTW0001 CODE EQ 1 (T)
 JOB05980 TTW0008 Flushed (PREDECESSOR)
 JOB05981 TTW0009 Ended RC(9) Apr 13 15:51:58
- FLUSHIF TTW0001 CODE EQ 2 (F)
             - FLUSHIF TTW0001 CODE NE 2 (T)
```

IBM

The **$DG** command displays the group. The parameters **JOB** and **JOBS** are added to show the basic relationships between the jobs in the group. For example:

```
$DG5966,LONG,JOB,JOBS
$HASP890 JOB(TST002)
$HASP890 JOB(TST002) JOB GROUP DEPENDENCY LIST VIA JOBS
$HASP890             PARENT    DEP JOB   DEP STAT  COMP ACT
$HASP890             --------  --------  --------  ---------
$HASP890             TTW0001   TTW0009   COMPLETE  FLUSH
$HASP890             TTW0001   TTW0008   COMPLETE  FLUSH
$HASP890             TTW0001   TTW0007   COMPLETE  FLUSH
$HASP890             TTW0001   TTW0006   COMPLETE  SATISFY
$HASP890             TTW0001   TTW0005   COMPLETE  FLUSH
$HASP890             TTW0001   TTW0004   COMPLETE  SATISFY
$HASP890             JOB GROUP JOB LIST
$HASP890             JOB NAME  JOBID     JOB STAT  COMP STAT
$HASP890             --------  --------  --------  ---------
$HASP890             TTW0009   JOB05973  Q=HARDCPY FLUSHED
$HASP890             TTW0008   JOB05972  Q=HARDCPY FLUSHED
$HASP890             TTW0007   JOB05971  Q=HARDCPY FLUSHED
$HASP890             TTW0006   JOB05970  Q=HARDCPY COMPLETE
$HASP890             TTW0005   JOB05969  Q=HARDCPY FLUSHED
$HASP890             TTW0004   JOB05968  Q=HARDCPY COMPLETE
$HASP890             TTW0001   JOB05967  Q=HARDCPY COMPLETE
```

Value Added with ThruPut Manager

- Layout allows dependencies to be determined more easily.
- More details are included: true or false results for the conditions, return codes.

Scheduling Parameters – WITH, HOLDUNTL, STARTBY

ThruPut Manager

ThruPut Manager provides (JCS) **WITH** and (JTS) **HOLD_UNTIL**.

Though it does not support the **STARTBY** feature, in ThruPut Manager Automation Edition, Service Level Manager (SLM) provides methods to ensure that the job flow is as desired through other means.

IBM

The IBM **SCHEDULE** statement has the **WITH**, **HOLDUNTL** and **STARTBY** keywords.

Group Affinity

ThruPut Manager

With ThruPut Manager you set the affinity of each job in the group in the job's JCL or using JAL statements.

IBM

z/OS has a parameter to set the affinity at the JEC Group Level.

Concurrent Jobs

Concurrent jobs are supported by z/OS JES2 2.2. ThruPut Manager does not have an analogous feature.

ThruPut Manager treats jobs using **`CONCURRENT`** as ThruPut Manager *exempt* jobs. In case an installation is running ThruPut Manager AE, these jobs are not analyzed and are not managed by Service Level Manager (SLM).

Introduction: z/OS 2.1

This new version of z/OS delivers a number of new functions that need careful consideration by the installation if they are to be properly exploited. Several of these new functions require updates to ThruPut Manager to both support and exploit them.

The following table lists many of the z/OS 2.1 new functions that are batch related. For each entry the impact on or value added by ThruPut Manager is indicated.

Function	ThruPut Manager
Back to Back Converter/Interpreter	Supported
Dynamic ENQ Downgrade	Exploited
8-Character Job Classes	Exploited
EXEC JCL Keyword – PARMDD	Supported
JOB JCL Keywords – SYSTEMS and SYSAFF	Supported
OUTPUT JCL Keywords – MERGE, DDNAME and COPYCNT	Supported
JCLLIB JCL Keyword – PROCLIB	Exploited
DD JCL Keywords – EATTR, SYMBOLS, MAXGENS	Supported

New Batch Related Features for z/OS 2.1

Each of these z/OS 2.1 related features are described below, with expanded discussion of the relevant ThruPut Manager functionality.

Back to Back Converter/Interpreter

This new function allows for a more consistent look to the JCL listing, since all statements are processed at essentially the same time and all statements, even JECL statements, are assigned a number. **`TYPRUN=SCAN`** is more complete and the **`OUTPUT JESDS`** keyword is honored if a job fails with a JCL error. Many of these benefits already existed with ThruPut Manager, since analysis normally takes place right after conversion and that does include the Interpreter phase. This function is probably worth consideration because it does provide some performance benefit by moving the Converter out of the JES2 address space into one or more dedicated C/I address spaces and only driving the Interpreter once.

This function may be turned on by specifying the **`JOBDEF INTERPRET=JES`** initialization statement within JES2, but will not take effect unless all members of the JESplex are at the z/OS 2.1 level. Care must be exercised prior to enabling this feature if certain SMS ACS exits or SMF exits, such as **`IEFUJV`**, are being used to initialize an environment that is later used by other SMF exits, such as **`IEFUJI`** or **`IEFACTRT`**. These exits will no longer be driven in the same address space, perhaps not even on the same z/OS image.

With ThruPut Manager

ThruPut Manager supports Back to Back Converter/Interpreter and when this function is on the only difference is that the Interpreter is no longer driven in either the Analyzer or the Initiator address spaces and, instead, the necessary data is read from the JES2 SPOOL.

Dynamic ENQ Downgrade

Background

This new function provides the ability to downgrade an ENQ from **EXCLUSIVE (DISP=OLD/NEW/MOD)** to **SHARED (DISP=SHR)** in a controlled fashion. This may seem to be a straightforward function to use, but great care must be exercised when doing so. The risk is that existing JCL may have **DISP=OLD/NEW/MOD** coded in an early step and then **DISP=SHR** coded in a later step, but still be writing to the file in that later step. If the downgrade is allowed, that dataset might now be accessed by another job while it is still being updated. For this reason, the default operation for this function is to not allow the downgrade.

Controls for this function are provided in JCL on the **JOB** statement and at the job class level. Unfortunately, the parameters specified on the JES2 **JOBCLASS** are not very intuitive – the values are **AUTO**, **ALLOW** and **DISALLOW** with the default being **ALLOW**, which could lead to the belief that the downgrade action was allowed by default. This is not the case, however. **ALLOW** means honor what is specified on the **DSENQSHR** keyword on the **JOB** statement.

The following is the description of the **JOBCLASS DSENQSHR** keywords:

ALLOW	The system is allowed to change the serialization on the data set to shared control *if the JOB statement specifies* **ALLOW**.
AUTO	The system is allowed to change the serialization on the data set to shared control *if the JOB statement specifies* **ALLOW** *or* **USEJC**. Note: **USEJC** is the default value on the **JOB** statement, so specifying **AUTO** at the class level means that *all* jobs submitted with that class will be eligible for downgrade *unless their JCL is changed*.
DISALLOW	The system is not allowed to change the serialization on the data set to shared control. The default value of the **DSENQSHR** keyword on the **JOB** statement is **USEJC**, so the two defaults combine to mean **DISALLOW**.

With ThruPut Manager

ThruPut Manager supports this function by providing two new JAL descriptors so that JAL can be used to determine what actual rule will be in effect for **DSENQSHR** for the job being analyzed. They are:

$INCLASS_DSENQSHR and **$JCL_DSENQSHR**. Messages may be written and decisions made based on these descriptors.

$INCLASS_DSENQSHR

The **$INCLASS_DSENQSHR** descriptor defines the **DSENQSHR=** value for the job's INPUT Submission Class. It can be used in **EVALUATE** definition statements and JAL Logic statements. This descriptor can also be used as an insert in message definition statements.

$JCL_DSENQSHR

The **$JCL_DSENQSHR** descriptor represents the **DSENQSHR=** value coded on the jobcard or the system default. It can be used in **EVALUATE** definition statements and JAL Logic statements. This descriptor can also be used as an insert in message definition statements.

DCS SET DSENQSHR

ThruPut Manager offers the additional ability to actually change the **DSENQSHR** value for the job being analyzed, regardless of the setting on the input class. Any ThruPut Manager installation with DCS enabled may now use the **DCS SET DSENQSHR** JAL statement to specify **ALLOW**, in which case a downgrade is allowed, or **DISALLOW**, in which case it is not. A **JOB** statement value of **DISALLOW** can never be overridden by JAL.

The **DSENQSHR** keyword for the **DCS SET** JAL action statement indicates how the system will treat changes in dataset disposition.

8-Character Job Classes

Background

This new function allows for the creation and deletion of meaningfully named job classes at any time, in addition to the 36 single character classes already in existence. Classes are created with the **$ADD JOBCLASS** command. Since all 36 available 1-character classes already exist, it follows that any new class created with **$ADD**, or specified at JES2 initialization time, is at least 2 characters long. New classes created with the **$ADD** command are automatically included in the selection list by ThruPut Manager – that is, jobs submitted with that class will be analyzed, etc. If this is not desired, the **TM CLASS DEL** command should be used to exempt the class from ThruPut Manager processing.

The simple fact that a class name is now unpredictable introduces a new problem at job submission time – what if the class specified on the JOB statement doesn't exist? The unfortunate answer to that question is that the job is failed immediately, and this can now happen even for the old 1-character classes because they can be marked as "inactive," although they cannot be deleted. An "inactive" class is treated exactly the same way as a missing class when specified at job submission time.

Classes can be deleted using the **$DEL JOBCLASS** command, but only if the class is already "inactive." A class is marked "inactive" by using the **$T JOBCLASS(x),ACTIVE=NO** command, but this will only succeed if there are no jobs queued to that class.

Eight-character classes to be used by Workload Manager or ThruPut Manager initiators carry no special consideration; but, classes to be used by traditional JES2 Initiators may pose a problem because there is a limit of 8 classes per JES2 Initiator (if any one of them is an 8-character class). If more than 8 classes are required, the solution is to combine them into a job class group.

A group is implicitly defined when the first class is added to it. Classes are added to a group using the **$T JOBCLASS(x),GROUP=y** command.

A class can only belong to one group, but any number of classes can belong to the same group, and up to 8 groups (and/or classes) can be assigned to an initiator. So, we now have a way to have literally hundreds of classes associated with a single initiator, perhaps not the best idea, but certainly possible.

Traditionally, classes are assigned to JES2 initiators based on their perceived importance to the organization – the more important, the earlier in the list. While that still has merit for individual classes assigned to an initiator, it does not when the class is within a group and the group is assigned to an initiator. Classes within a group are processed in a round robin fashion; JES2 remembers its place in the group when it selects a job and then, when it's time to look for another job, starts looking at the next class in the group, only going back to the first class when the last class in the group has been scanned.

In other words, if you have a legitimate reason for having more than 8 classes in a specific order on a JES2 initiator, you cannot translate that to the 8-character class world.

Highlights with ThruPut Manager

ThruPut Manager provides a number of benefits to the new 8-character class environment.

Reduces number of classes needed

ThruPut Manager generally reduces the number of classes required; this is particularly true for ThruPut Manager AE and ThruPut Manager AE+. ThruPut Manager allows the use of 8-character classes for all of its specialty classes and thus does not require use of a 1-character class. For example, if today your analysis class is 9 and you have a General Services class of 2 you could change them to **ANALYSIS** and GS respectively thereby freeing up the 1-character classes 9 and 2 for other use.

JAL handles classification of jobs

Another benefit is the buffer between the end user and the installation provided by the Job Action Language, which allows the introduction of the new descriptive classes without requiring any JCL changes; the new classes can simply be assigned in JAL thereby removing the risk of a user error when selecting a class.

Classes maintained in JES2 Checkpoint

All classes, including the TM selection class list, are now maintained in the JES2 checkpoint and therefore *apply to the entire JESplex*. ThruPut Manager classes are all defined or modified using the **TM CLASS** command only. **TMPARM** statement is allowed for legacy reasons, but classes defined on the **TMPARM** are ignored.

TM CLASS is updated

The syntax of the **TM CLASS** command has not changed other than it now accepts larger class name fields. The display format of the **TM CLASS** command has changed. The 1-character and 8-character class format shown below:

```
/TM CLASS
DTM3233I TM CLASS LIST
ANALYZE=9      DEFERRED=DX
SELECT=ABCEFGHIJKLMNOPRSTUVWYZ012345678
NO ON_DEMAND CLASS
SLM - PCS=1
      GENERAL SERVICES=2
```

1-character class format

```
/TM CLASS
DTM3233I TM CLASS LIST
Analysis.....  ANALYSIS
Deferred.....  D,X
Selectable...  A,B,C,D,DPROD,E,ERNIE,F,G,H,I,J,K,L,M,N,O,P,R,S,T,U,W,X,Y,Z
               0,1,2,3,4,5,6,7,8,9
Exempt.......  Q
On_Demand....  None
PCS..........  1
General Srvcs  GS
Default        None
```

8-character class format

Messages accommodate new class size

ThruPut Manager messages that contain class information have been modified to allow for the larger value. In most cases, the field has been moved to the end of the message. In other cases, the message has been reformatted, as in the JOB command example below.

```
/JOB D 1737
DTM0220I JOB DISPLAY
JOB01737 $DCSTEST AWAITING EXECUTION  Q  PRIO  9 ANY
```

1-character class format

```
/JOB D 1738
DTM0220I JOB DISPLAY
JOB01738 $DCSTEST   AWAITING EXEC FOR 35 MINS
                 CLASS   : ERNIE
                 PRIORITY: 9
                 SYSAFF  : ANY
```

8-character class format

Specification of a DEFAULT Job Class

ThruPut Manager adds functionality to allow the specification of a **DEFAULT** job class to be used in situations where an invalid class has been specified either in JCL or JAL. If a **DEFAULT** job class has been specified, it will always be used in cases where an invalid class is encountered, instead of failing the job. The **DEFAULT** job class is specified on the **TM CLASS** command and maintained in the JES2 checkpoint.

```
TM CLASS SET DEFAULT(class)
```

This support does require the addition of JES2 Exits 02 and 52. JAL descriptors are available to indicate that the original input class was invalid.

For legacy reasons, as explained earlier, the classes on a **TMPARM** statement are ignored, and a message is issued, in favor of the **TM CLASS** command.

EXEC JCL Keyword – PARMDD

Background

The **PARMDD** keyword on the **EXEC** JCL statement allows the passing of a parameter to the executing program that exceeds the previous limit of 100 characters. The keyword identifies a DD statement that defines a dataset containing the parameter data. The program is unaware of the source of the parameter data.

With ThruPut Manager

ThruPut Manager supports this function by providing a new JAL keyword, **PARMDD**, on **$PROGRAM** JAL statement indicating whether this keyword was specified or not (**YES** or **NO**).

$PROGRAM PARMDD

PARMDD is a keyword on **$PROGRAM** JAL statement that specifies a DD statement that contains the **PARM** data. This DD is read and the data is automatically passed to the calling program. The maximum amount of data is 32760 bytes. **PARMDD** is mutually exclusive with the **PARM** and **PARMFLD** keywords.

JOB JCL Keywords – SYSTEMS and SYSAFF

With ThruPut Manager

The new **SYSAFF=** and **SYSTEMS= JOBCARD** keywords are supported without any changes. Both of these keywords internally set the system affinity mask so the **$INSYSAFF** descriptor will continue to work as before.

OUTPUT JCL Keywords – MERGE, DDNAME, COPYCNT

Background

The new keywords on the **OUTPUT** JCL statement (**MERGE**, **DDNAME** and **COPYCNT**) replace keywords on the **/*OUTPUT** JECL statement and, in the case of the **DDNAME** keyword, provide similar functionality to the JES3 **DDNAME** JECL statement.

With ThruPut Manager

All these new keywords and functionality are fully supported by the ThruPut Manager SYSOUT Services (SOS) feature.

JCLLIB JCL Keyword – PROCLIB

Background

This replaces the **/*JOBPARM PROCLIB** JECL statement.

With ThruPut Manager

ThruPut Manager adds the **$PROC**edure descriptor in JAL, which works in much the same way as the **$PROGRAM** descriptor does. It allows installations to examine the use of their PROCedure(s) and where they came from.

Three new JAL list descriptors are added:

- **$LIST_PROC_NAME** – The procedure name
- **$LIST_PROC_TYPE** – The procedure TYPE
- **$LIST_PROC_LIBRARY** – The procedure data set name

These descriptors can be can be inserted in messages also they can be used when processing the ThruPut Manager analysis performance SMF record, creating reports on the various PROCedure(s) and Libraries being used.

Also three new DAL descriptors are added to describe the current step's procedure, type and library:

- **$JCL_PROC_NAME** – The procedure name
- **$JCL_PROC_TYPE** – The procedure TYPE
- **$JCL_PROC_LIBRARY** – The procedure data set name

Keywords – EATTR, SYMBOLS, MAXGENS

Background

The new keywords on the DD JCL statement **EATTR**, **SYMBOLS**, and **MAXGEN** are added in z/OS 2.1.

With ThruPut Manager

New DAL descriptors are created to reflect new keywords on the DD JCL statement **EATTR**, **SYMBOLS**, and **MAXGEN**:

- **$JCL_EATTR**
- **$JCL_SYMBOLS**
- **$JCL_SYMBOLS_LOGDD**
- **$JCL_MAXGENS**

ThruPut Manager Installation Considerations

Background

Much of the new functionality cannot be exploited until all images are at the z/OS 2.1 level and it follows that those images must also be running ThruPut Manager Version 7.1.

Installing Version 7.1

Installing Version 7.1 does require the addition of JES2 Exits 02 and 52, which means the JES2 initialization parameters require updating. Two EXIT statements must be added as follows:

```
EXIT(2) ROUTINE=DTMJ2X02,STATUS=ENABLED,TRACE=NO

EXIT(52) ROUTINE=DTMJ2X52,STATUS=ENABLED,TRACE=NO
```

It is also recommended to remove all reference to classes from the **TMPARM** statement, since these will be ignored and may just become a source of confusion. Specifically remove **ANALYSIS=**, **DEFER=**, **EXEMPT=**, and **SELECT=** keywords, if used. Also ThruPut Manager AE customers may have classes specified in the legacy SLM keyword, the entire SLM keyword should be replaced with the **TMINITS** keyword.

At the earliest opportunity after JES2 and ThruPut Manager have completed initialization on the first image in the JESplex, the **TM CLASS D** command should be issued to verify the classes are defined as expected. The **TM CLASS** command should then be used to make any necessary corrections. If the new **DEFAULT** class is to be used, it can be defined at this time. Enter **TM CLASS ?** to obtain the full syntax of the command.

If 8-character classes are to be deployed, review the existing JES2 initiator class structure. If any have more than 8 classes defined it may not be possible to simply replace the existing classes with their descriptive 8-character equivalents.

MVS solutions inc.

8300 Woodbine Avenue, 4th Floor
Markham ON Canada L3R 9Y7
Tel: (905) 940-9404 Fax: (905) 940-5308
Email: marketing@mvssol.com www.mvssol.com