



# JCL

## Chapter b3

# Modifying EXEC parameters

## **Job Control Language**

**Chapter a1. Introduction to JCL**

**Chapter a2. Coding JOB statements**

**Chapter a3. Coding EXEC statements**

**Chapter a4. Coding DD statements**

**Chapter a5. Analyzing job output**

**Chapter a6. Conditional processing**

## **Job Control Language**

**Chapter b1. Using special DD statements**

**Chapter b2. Introducing procedures**

**Chapter b3. Modifying EXEC parameters**

**Chapter b4. Modifying DD parameters**

**Chapter b5. Determining the effective JCL**

**Chapter b6. Symbolic parameters**

## **Job Control Language**

**Chapter c1. Nested procedures**

**Chapter c2. Cataloging procedures**

**Chapter c3. Using utility programs**

**Chapter c4. Sample utility application**

**Modifying EXEC parameters.**

## **Chapter b3**

# **Modifying EXEC parameters**

## Modifying EXEC parameters.

### Unit introduction.

**Before using a procedure it is essential to ensure that it meets all the processing requirements. This might involve some minor alterations to the procedure to ensure that it has all the requirements to execute the job.**

**This unit describes how to alter EXEC statement parameters at the time of job submission. The information contained in this unit can be applied to both in-stream and cataloged procedures.**

## Modifying EXEC parameters.

### Course objectives.

#### Be able to:

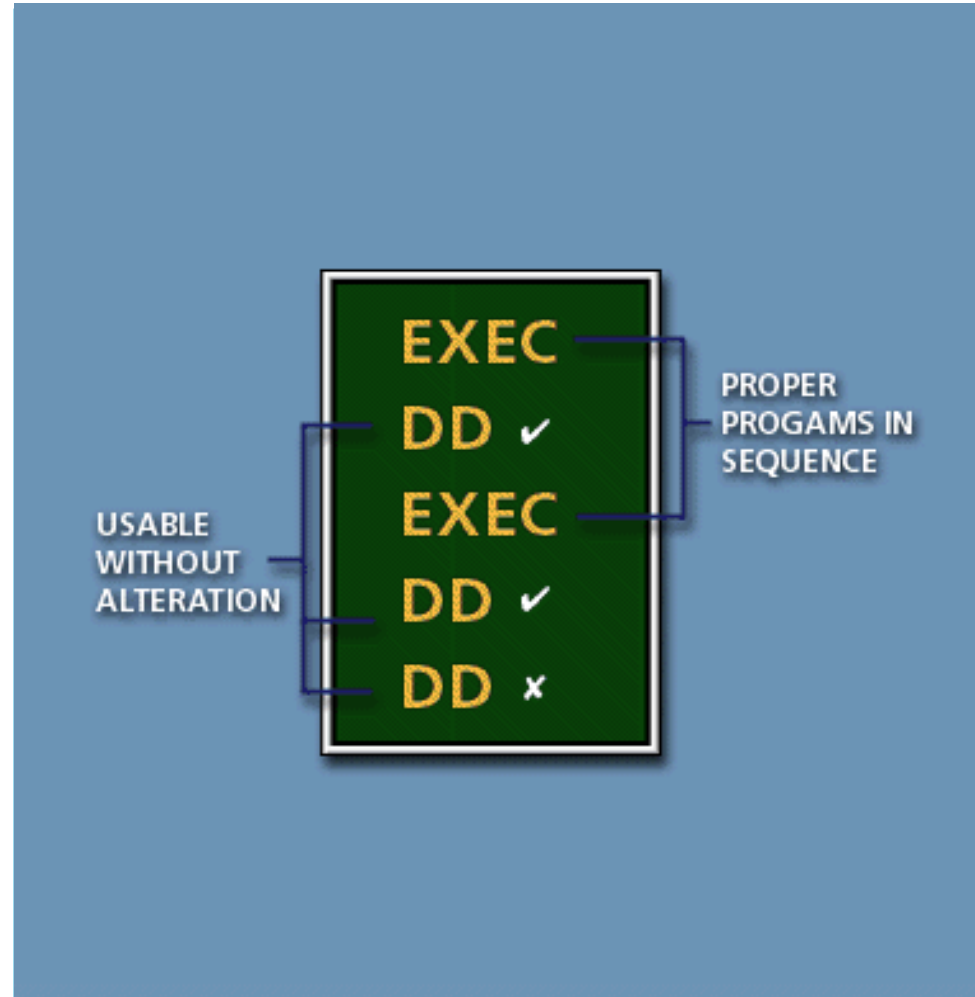
- **Specify what factors determine if a procedure meets the requirements of a job.**
- **Invoke a procedure, making temporary alterations to it if necessary.**
- **Add, override or nullify parameters on procedure step EXEC statements.**
- **Correctly sequence multiple changes to EXEC statement parameters.**

## Analyzing procedures.

### Identifying analysis criteria.

A procedure listing is obtained and examined to ensure that it meets the following two criteria:

- The procedure invokes the proper programs in the desired sequence.
- Most of the DD statements are usable without major alteration.





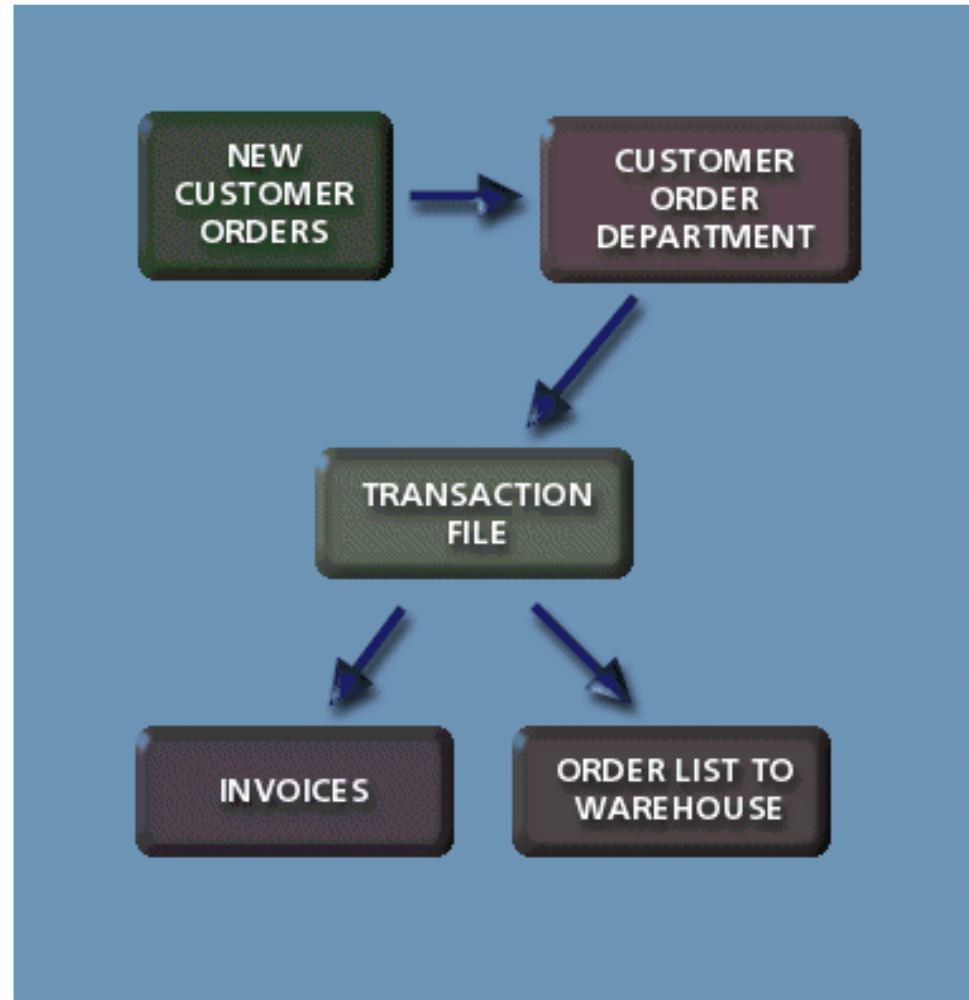
## Analyzing procedures.

### Identifying analysis criteria – an example.

Consider a case in which a company buys goods wholesale from several manufacturers and markets them retail to other customers.

Each week the customer order department creates a transaction file that contains new customer orders.

A list of orders is sent to the warehouse and an invoice is sent to each customer. The order list and associated invoices are printed once a week.



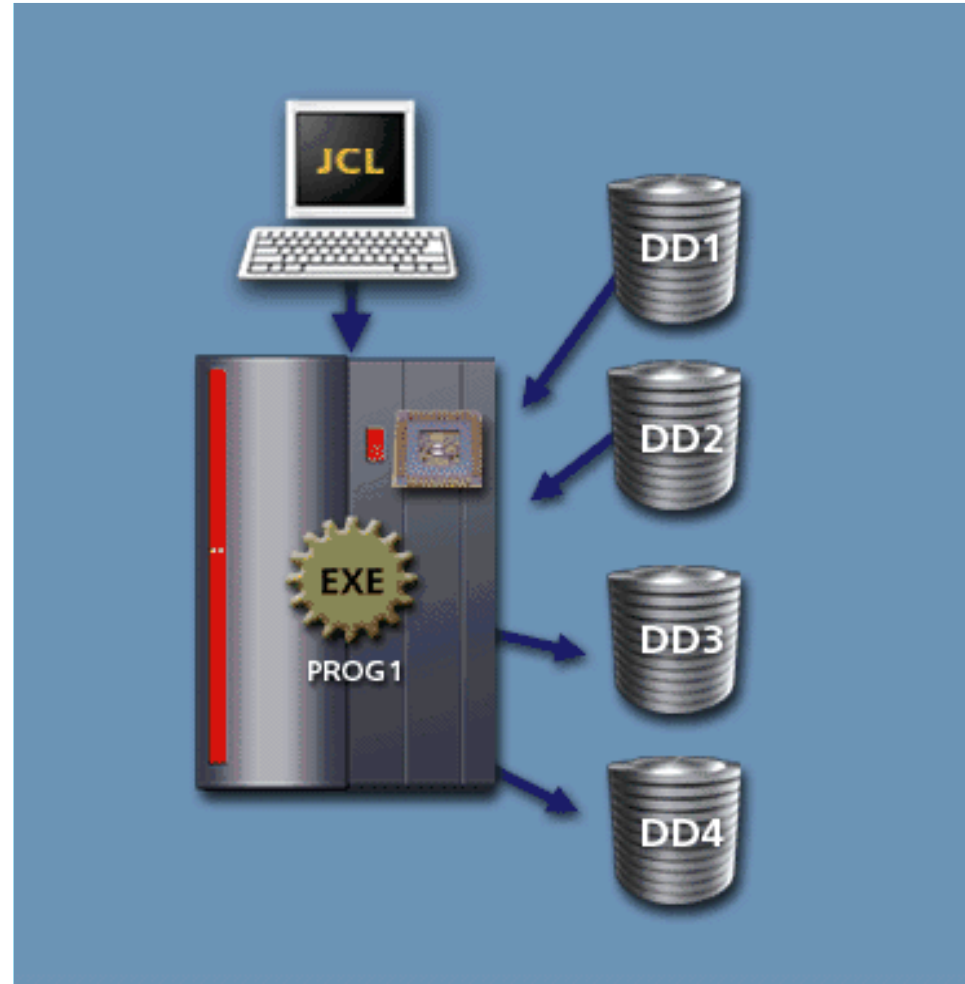
## Analyzing procedures.

### Identifying analysis criteria – an example.

A program PROG1 checks the weekly input transactions against entries in a master customer data set. Valid transactions are written to a new data set used as input for another program PROG2.

PROG1 refers to the following data sets:

- DD1: Input transactions.
- DD2: Master customer data set.
- DD3: Transaction exception report.
- DD4: Set of valid transactions that are passed to PROG2.



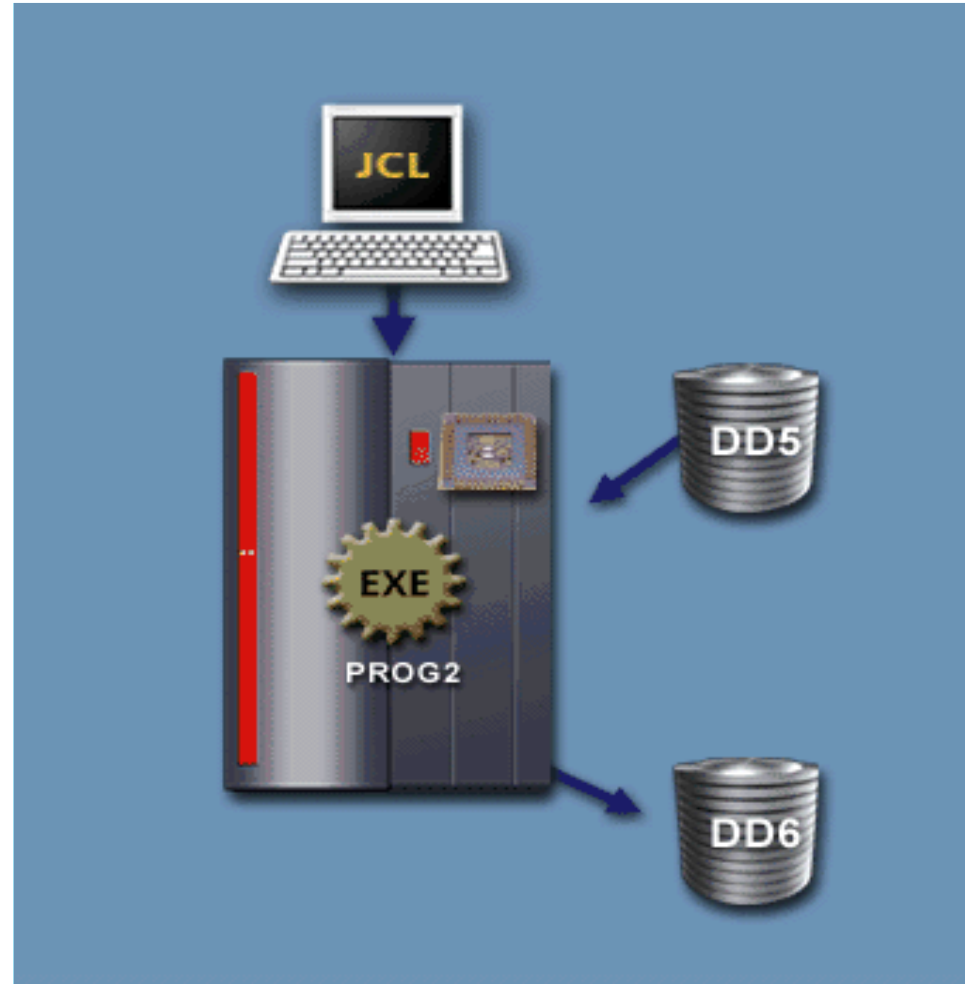
## Analyzing procedures.

### Identifying analysis criteria – an example.

PROG2 reads the valid transactions passed from PROG1 and creates an order list/invoice for each customer.

PROG2 refers to the following data sets:

- DD5: Set of valid transactions passed from PROG1.
- DD6: Order list/invoice for each customer.



## Analyzing procedures.

### Identifying analysis criteria – an example.

Shown on the right is the procedure named TRANSACT that is used to accomplish this task.

A procedure listing is obtained to determine:

- If it uses the required programs in the required sequence.
- If it uses appropriate data sets.

```
//PSTEP1 EXEC PGM=PROG1,TIME=(1,30)
//DD1 DD DSN=INTRAN,DISP=SHR
//DD2 DD DSN=MASTER,DISP=SHR
//DD3 DD SYSOUT=A
//DD4 DD DSN=&&VALID,
// UNIT=SYSDA,
// DISP=(NEW,PASS),
// SPACE=(TRK,(1,1))
//PSTEP2 EXEC PGM=PROG2,TIME=5
//DD5 DD DSN=&&VALID,
// DISP=(OLD,DELETE)
//DD6 DD SYSOUT=A
```

## Analyzing procedures.

### Are we on track?

**Based on the analysis of the JCL in the previous example, do you think this is an appropriate procedure for the task, as described?**

- A. No, because not all data sets are taken into consideration.**
- B. There is not enough information to decide.**
- C. Yes, because both criteria are met.**

## Analyzing procedures.

### Analysis explanation.

The listing for the procedure TRANSACT indicates the following:

The procedure executes 2 programs: PROG1 and PROG2.

PROG1 uses the following data sets:

- A cataloged data set INTRAN (DD1 DD statement).
- A cataloged data set MASTER (DD2 DD statement).
- SYSOUT class A output (DD3 DD statement).
- A temporary data set &&VALID that is passed to PROG2 (DD4 DD statement).

```
//PSTEP1 EXEC PGM=PROG1,TIME=(1,30)
//DD1 DD DSN=INTRAN,DISP=SHR
//DD2 DD DSN=MASTER,DISP=SHR
//DD3 DD SYSOUT=A
//DD4 DD DSN=&&VALID,
// UNIT=SYSDA,
// DISP=(NEW,PASS),
// SPACE=(TRK,(1,1))
//PSTEP2 EXEC PGM=PROG2,TIME=5
//DD5 DD DSN=&&VALID,
// DISP=(OLD,DELETE)
//DD6 DD SYSOUT=A
```

## Analyzing procedures.

### Analysis explanation.

PROG2 uses the following data sets:

- A temporary data set named &&VALID, which is passed from PROG1 (DD5 DD statement).
- SYSOUT class A output (DD6 DD statement).

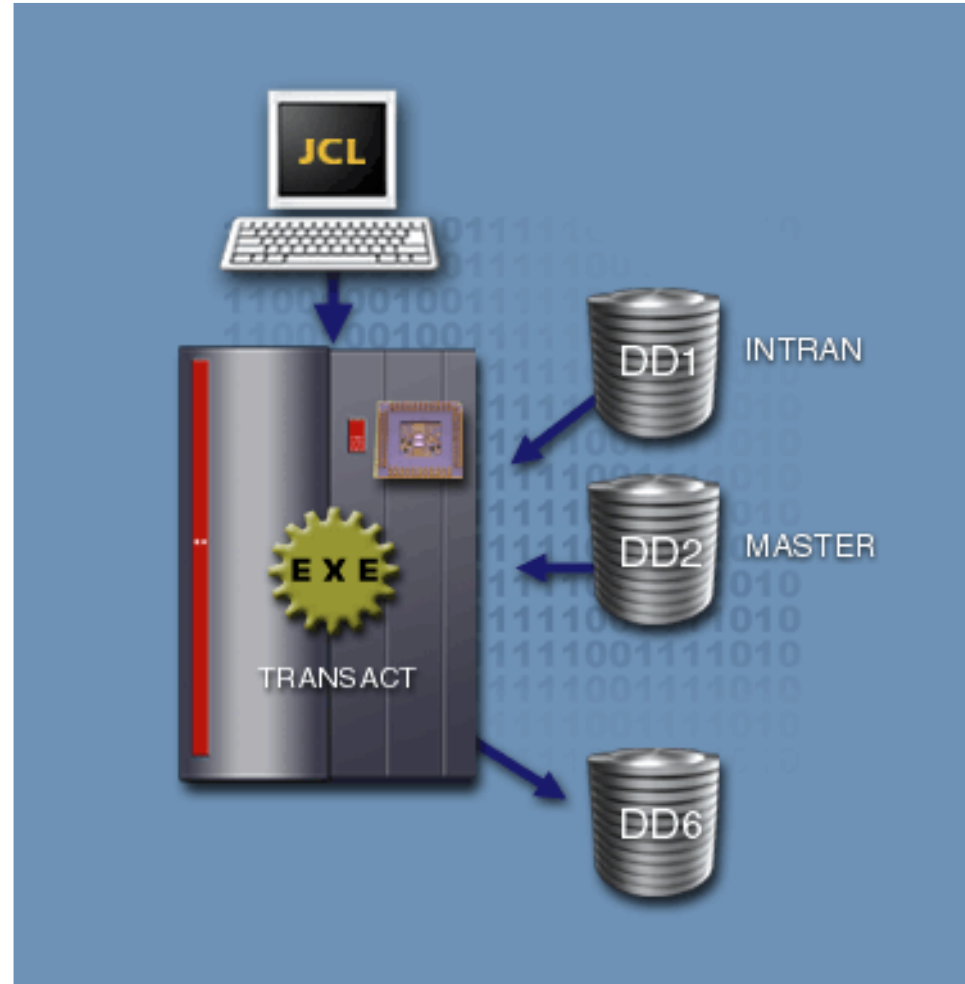
In addition, a TIME parameter is included on the PSTEP1 and PSTEP2 EXEC statements to restrict the amount of time the programs are permitted to use the central processor.

```
//PSTEP1 EXEC PGM=PROG1,TIME=(1,30)
//DD1 DD DSN=INTRAN,DISP=SHR
//DD2 DD DSN=MASTER,DISP=SHR
//DD3 DD SYSOUT=A
//DD4 DD DSN=&&VALID,
// UNIT=SYSDA,
// DISP=(NEW,PASS),
// SPACE=(TRK,(1,1))
//PSTEP2 EXEC PGM=PROG2,TIME=5
//DD5 DD DSN=&&VALID,
// DISP=(OLD,DELETE)
//DD6 DD SYSOUT=A
```

## Analyzing procedures.

### Analysis explanation.

The TRANSACT procedure listing is obtained and evaluated to ensure that the procedure executes all the required programs in the proper sequence and the appropriate data sets are used to accomplish the application.





Analyzing procedures.

Are we on track?

Code a statement to invoke TRANSACT.

//JSTEP EXEC \_\_\_\_\_

## Analyzing procedures.

**Are we on track?**

**A procedure must meet which of the following criteria?**

- A. Most of the DD statements are usable as they are or might need some minor alteration.**
- B. The programs in the procedure are located in the same library.**
- C. It invokes the proper programs in the correct sequence.**
- D. The DD statements point to a single storage volume.**

## Analyzing procedures.

### Glossary.

#### Cataloged data set

**A non-temporary data set for which the system has recorded the unit and volume on which the data set resides.**

#### SYSOUT

**A keyword that defines a print data set. It instructs the system to queue the output on a direct-access volume.**

#### Temporary data set

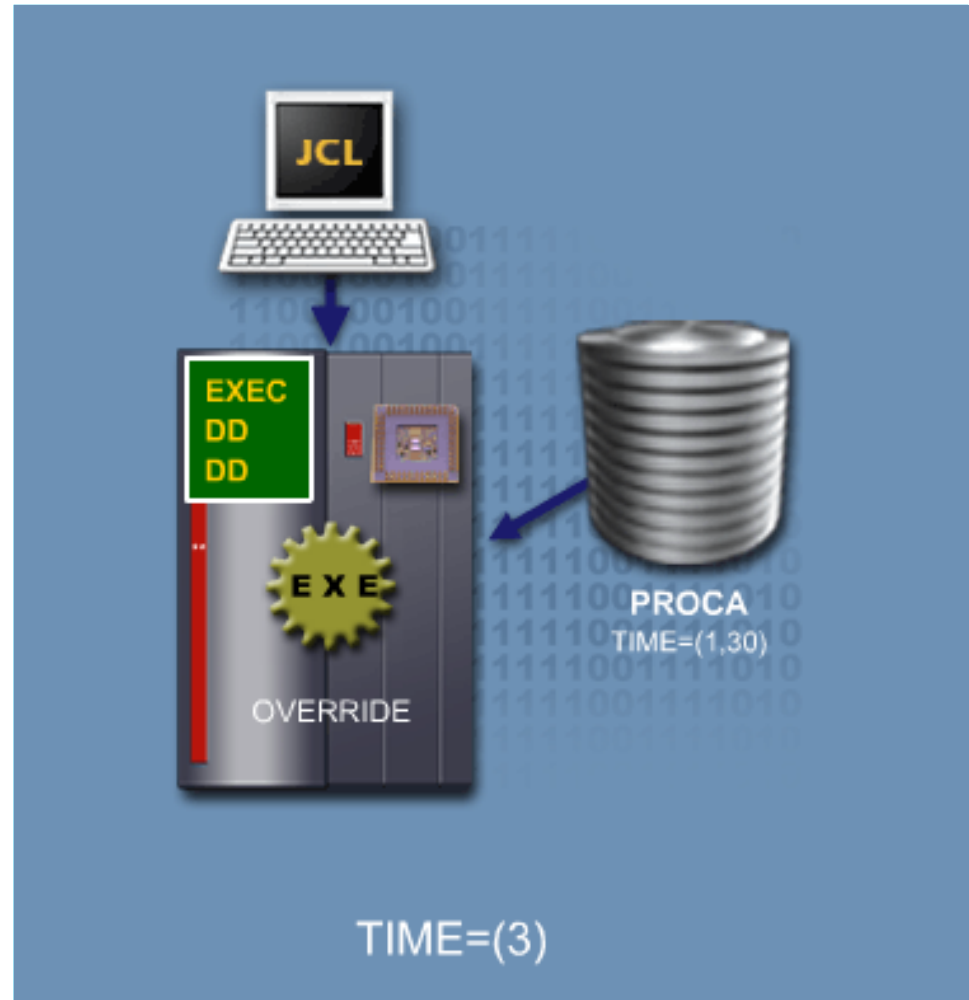
**A data set which stores data needed only for the duration of the job.**

## Changing EXEC parameters.

### Coding changes.

A procedure listing helps a programmer to analyze the procedure for its usability. In some cases a procedure might satisfy all the basic requirements, but might need some minor alterations.

This can be done by changing the EXEC and DD parameters when the procedure is invoked. However, these alterations are applicable only for one invocation. They do not permanently modify the procedure definition.



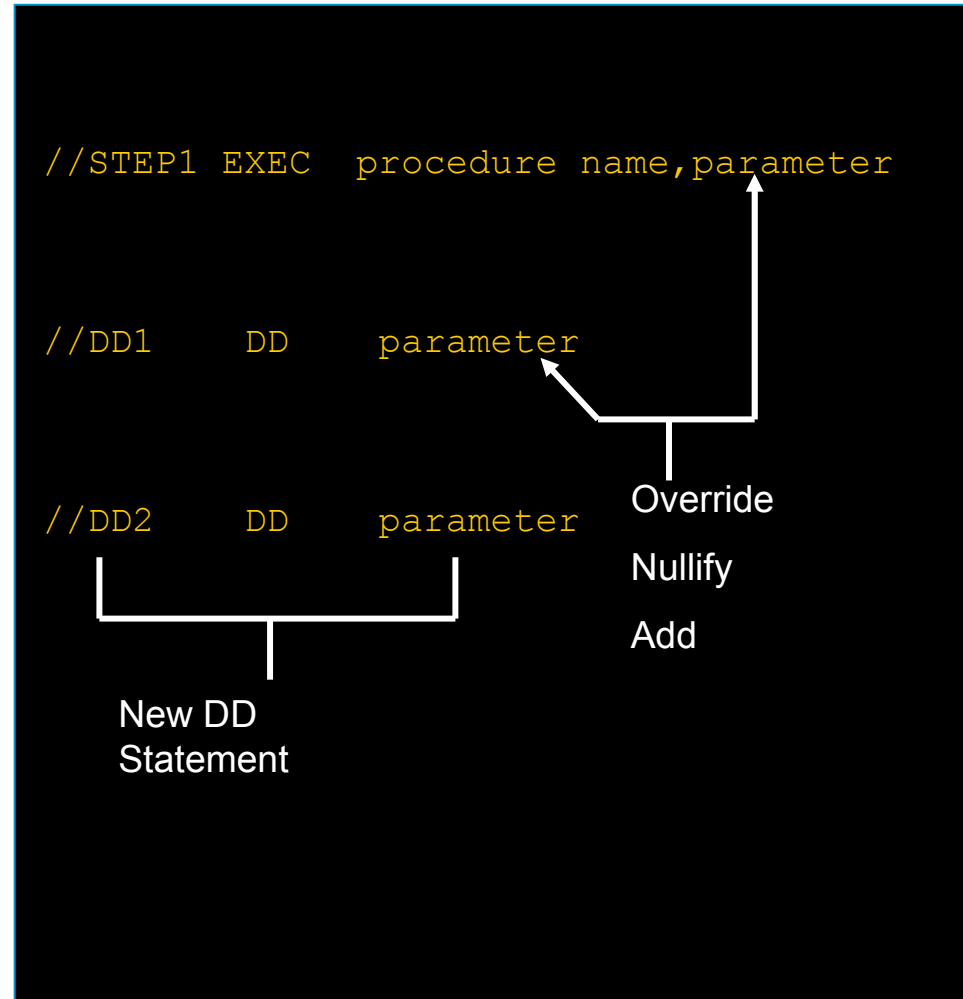
## Changing EXEC parameters.

### Coding changes.

Changes can be made to procedure EXEC statement parameters such as TIME, ACCT, and PARM.

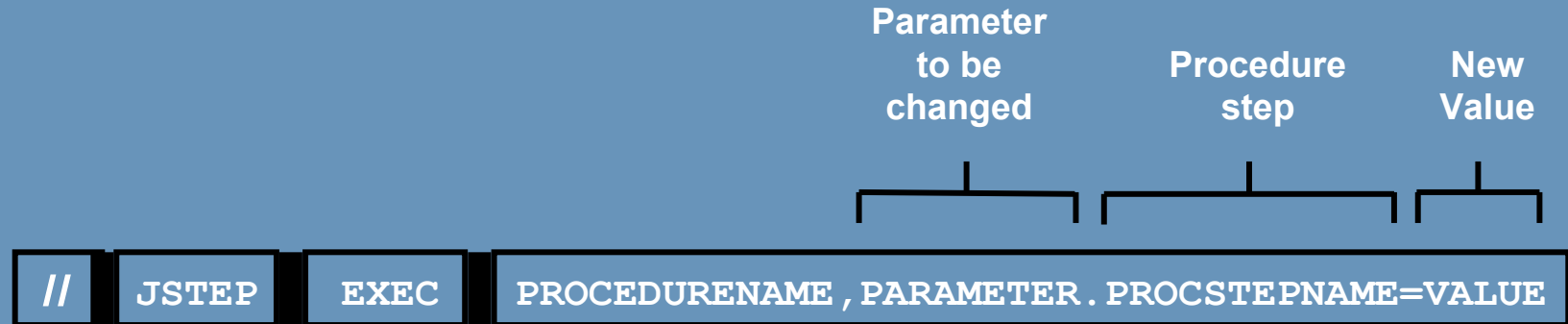
The programmer can change these parameters in the following ways:

- Override the parameters on the procedure EXEC statement.
- Nullify parameters on the procedure EXEC statement.
- Add parameters to the procedure EXEC statement.



## Changing EXEC parameters.

### Coding changes to EXEC statement parameters.



The general form for coding changes to EXEC statement parameters is as follows (it is shown above):

- To modify EXEC statement parameters for any procedure step, append the procedure step to the parameter.
- If the stepname is omitted, the parameter applies to all steps of the procedure, with the exception of the PARM parameter.
- If the stepname is omitted when adding or overriding a PARM parameter, the PARM value only applies to the first step in the procedure.
- Any PARM parameters in subsequent steps within the procedure are nullified.

## Changing EXEC parameters.

# Changing EXEC statement parameters – an example.

To illustrate an EXEC statement override, review the procedure definition for TRANSACT, as shown on the right.

The time allocated for TRANSACT is 1 minute 30 seconds.

If the transaction file for the week were much larger than usual, you might want to change the time allocated for the procedure to 3 minutes. You would code the following override statement:

```
//JSTEP EXEC TRANSACT,TIME.PSTEP1=3
```

```
//PSTEP1 EXEC PGM=PROG1,  
//          TIME=(1,30)  
//DD1      DD  DSN=INTRAN,DISP=SHR  
//DD2      DD  DSN=MASTER,DISP=SHR  
//DD3      DD  SYSOUT=A  
//DD4      DD  DSN=&&VALID,  
//          DISP=(NEW,PASS),  
//          UNIT=SYSDA,  
//          SPACE=(TRK,(1,1))  
//PSTEP2 EXEC PGM=PROG2,TIME=5  
//DD5      DD  DSN=&&VALID,  
//          DISP=(OLD,DELETE)  
//DD6      DD  SYSOUT=A
```



## Changing EXEC parameters.

### Implementing coding changes.

General syntax for changes to EXEC statement parameters is as follows:

```
//JSTEP      EXEC  procedurename,  
//          parameter.procstepname=value
```

Implementing coding changes to EXEC statements involves the following steps:

- Follow the name of the procedure with a coma.
- Give the name of the EXEC statement parameter to be overridden, nullified or added, followed by a period.
- Give the name of the procedure step, followed by an equal sign.
- Give the new value for the parameter if you are overriding or adding a value. Do not code a value if the parameter is to be nullified.



**Changing EXEC parameters.**

**Are we on track?**

**Changing parameters at the time you invoke a procedure has what effect?**

**A. Changes apply to the number of invocations you specify in the JCL.**

**B. Changes apply to all procedures containing the edited parameters.**

**C. Changes apply once only to the current invocation.**

**D. Changes apply to all future invocations of the procedure.**

## Changing EXEC parameters.

**Are we on track?**

**Which of the following changes can you make at the time you execute a procedure?**

- A. Temporarily add operands such as ACCT to procedure EXEC statements.**
- B. Alter the library copy of the JCL contained in cataloged procedure.**
- C. Override the PGM parameter on procedure EXEC.**

## Override statements.

# Overriding statement parameters.

An override statement is used to change an existing parameter value.

Consider the TRANSACT procedure which definition is shown on the right. Note that the time that PROG1 can run is 1 minute 30 seconds.

Assume that for a particular week, the transaction file to be processed is too large and the time that PROG1 can run needs to be increased to 3 minutes.

```
//PSTEP1 EXEC PGM=PROG1, TIME=(1,30)
//DD1 DD DSN=INTRAN, DISP=SHR
//DD2 DD DSN=MASTER, DISP=SHR
//DD3 DD SYSOUT=A
//DD4 DD DSN=&&VALID,
// DISP=(NEW,PASS),
// UNIT=SYSDA,
// SPACE=(TRK,(1,1))
//PSTEP2 EXEC PGM=PROG2, TIME=5
//DD5 DD DSN=&&VALID,
// DISP=(OLD,DELETE)
//DD6 DD SYSOUT=A
```

## Override statements.

# Overriding statement parameters.

To override the original time parameter, the TRANSACT can be invoked with the following EXEC statement:

```
//JSTEP EXEC TRANSACT,TIME.PSTEP1=3
```

The resulting JCL would behave as the procedure on the right. Note the new parameter in the resulting JCL.

However, this override is only temporary. The procedure definition does not change. The next time the procedure is invoked, it will revert to the original definition.

```
//PSTEP1 EXEC PGM=PROG1,TIME=3
//DD1 DD DSN=INTRAN,DISP=SHR
//DD2 DD DSN=MASTER,DISP=SHR
//DD3 DD SYSOUT=A
//DD4 DD DSN=&&VALID,
// DISP=(NEW,PASS),
// UNIT=SYSDA,
// SPACE=(TRK,(1,1))
//PSTEP2 EXEC PGM=PROG2,TIME=5
//DD5 DD DSN=&&VALID,
// DISP=(OLD,DELETE)
//DD6 DD SYSOUT=A
```

## Override statements.

### Are we on track?

**Following is a portion of the JCL for a procedure named MYPROC.**

```
//PSTEP1      EXEC  PGM=PROGA,TIME=(3,30)
//DD1        DD    DSN=A,DISP=SHR
//DD2        DD          ...
//PSTEP2      EXEC  PGM=PGMB,TIME=5
//DD3        DD          ...
```

**Code a statement to invoke the procedure named MYPROC. Assume you want to restrict the amount of time PROGA is permitted to use the CPU to 2 minutes.**

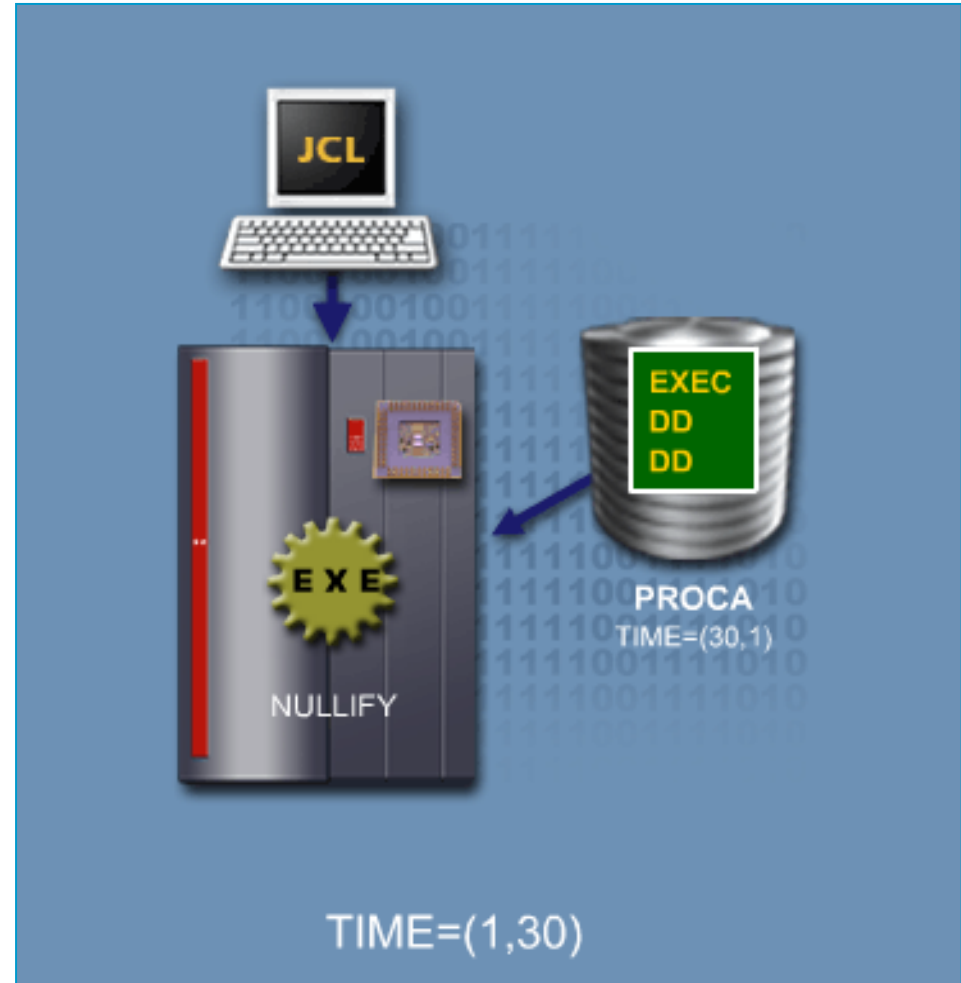
**//JSTEP EXEC \_\_\_\_\_**

## Nullification statements.

# Nullifying EXEC statement parameters.

A procedure can be modified by nullifying an EXEC statement parameter.

Most installations have values that are assigned to EXEC statement parameters automatically. For example, a default value may be assigned for the TIME parameter. The default values may be overridden when the procedure is defined. To return to the installation's default value, the programmer can code a statement that nullifies the parameter.



## Nullification statements.

### Nullifying EXEC statement parameters.

The format for nullifying an EXEC statement parameter is:

```
//JSTEP EXEC procedurename,  
//           parameter.procstepname=
```

Note that in the format, the programmer specifies the parameter and the procedure step in which it appears.

Also note that no value is assigned to the parameter in the nullifying EXEC statement.

## Nullification statements.

# Nullifying EXEC statement parameters – an example.

Consider the TRANSACT procedure.

The procedure definition for PROG1 (in the procedure step PSTEP1) has specified a CPU time of 1 minute 30 seconds for processing a transaction file. This processing time may not be adequate for a larger file.

If the default time is adequate, the programmer might want to execute the procedure taking the system default time for PROG1.

```
// PSTEP1 EXEC PGM=PROG1, TIME=(1, 30)
// DD1 DD DSN=INTRAN, DISP=SHR
// DD2 DD DSN=MASTER, DISP=SHR
// DD3 DD SYSOUT=A
// DD4 DD DSN=&&VALID,
// DISP=(NEW, PASS) ,
// UNIT=SYSDA,
// SPACE=(TRK, (1, 1) )
// PSTEP2 EXEC PGM=PROG2, TIME=5
// DD5 DD DSN=&&VALID,
// DISP=(OLD, DELETE)
// DD6 DD SYSOUT=A
```



## Nullification statements.

### Nullifying EXEC statement parameters – an example.

To do this, the programmer needs to nullify the TIME specified in the procedure definition on the PSTEP1 EXEC statement.

The following EXEC statement which invokes TRANSACT would nullify the time parameter:

```
//JSTEP1 EXEC TRANSACT,TIME.PSTEP1=
```

The resulting JCL would behave as the procedure on the right.

```
//PSTEP1 EXEC PGM=PROG1
//DD1 DD DSN=INTRAN,DISP=SHR
//DD2 DD DSN=MASTER,DISP=SHR
//DD3 DD SYSOUT=A
//DD4 DD DSN=&&VALID,
// DISP=(NEW,PASS),
// UNIT=SYSDA,
// SPACE=(TRK,(1,1))
//PSTEP2 EXEC PGM=PROG2,TIME=5
//DD5 DD DSN=&&VALID,
// DISP=(OLD,DELETE)
//DD6 DD SYSOUT=A
```

## Nullification statements.

### Are we on track?

**Following is a portion of the JCL for a procedure named MYPROC.**

```
//PSTEP1      EXEC PGM=PROGA  
//DD1        DD      ...  
//PSTEP2      EXEC PGM=PROGB,TIME=5  
//DD2        DD      ...
```

**Assume that for this use of the procedure, you want to return to the installation-defined CPU time limit for PROGB. Code the statement to invoke MYPROC.**

```
//JSTEP EXEC _____
```

## Addition statements.

### Adding parameters to a procedure.

Addition statements are used to add parameters to a procedure. The programmer can code additions on the EXEC statement that invokes the procedure.

Consider the TRANSACT procedure. Assume that a programmer wants to supply the current date for PROG1.

This parameter can be coded in the following way while invoking the procedure:

```
//JSTEP EXEC procedurename,parameter.procstepname=value
```

```
//JSTEP EXEC TRANSACT,PARM.PSTEP1='01/29/99'
```



**Addition**

## Addition statements.

### Adding parameters to a procedure.

Parameters can be added to one or more procedure steps. In the TRANSACT procedure, the current date can be supplied to PROG1 and PROG2 by means of the PARM parameter.

The code would be as follows:

```
//JSTEP EXEC TRANSACT,  
//          PARM.PSTEP1='01/29/99',  
//          PARM.PSTEP2='01/29/99'
```

Note the following points when sequencing multiple EXEC statement additions:

- The additions are in procedure step sequence.
- A comma separates the name of the procedure from the first parameter addition, and the parameter additions from each other.

## Addition statements.

# Adding parameters to a procedure.

The TRANSACT procedure definition is shown on the right.

```
//PSTEP1 EXEC PGM=PROG1,TIME=(1,30)
//DD1      DD  DSN=INTRAN,DISP=SHR
//DD2      DD  DSN=MASTER,DISP=SHR
//DD3      DD  SYSOUT=A
//DD4      DD  DSN=&&VALID,
//          DD  DISP=(NEW,PASS),
//          DD  UNIT=SYSDA,
//          DD  SPACE=(TRK,(1,1))
//PSTEP2 EXEC PGM=PROG2,TIME=5
//DD5      DD  DSN=&&VALID,
//          DD  DISP=(OLD,DELETE)
//DD6      DD  SYSOUT=A
```

## Addition statements.

# Adding parameters to a procedure.

The following EXEC statement supplies the current date for PROG1 and PROG2:

```
//JSTEP EXEC TRANSACT,  
//          PARM.PSTEP1=' 01/29/91',  
//          PARM.PSTEP2=' 01/29/91'
```

The resulting JCL would behave as the procedure on the right.

```
//PSTEP1 EXEC PGM=PROG1, TIME=(1,30),  
//          PARM=' 01/29/91'  
//DD1      DD DSN=INTRAN, DISP=SHR  
//DD2      DD DSN=MASTER, DISP=SHR  
//DD3      DD SYSOUT=A  
//DD4      DD DSN=&&VALID,  
//          DISP=(NEW,PASS),  
//          UNIT=SYSDA,  
//          SPACE=(TRK,(1,1))  
//PSTEP2 EXEC PGM=PROG2, TIME=5,  
//          PARM=' 01/29/91'  
//DD5      DD DSN=&&VALID,  
//          DISP=(OLD,DELETE)  
//DD6      DD SYSOUT=A
```

## Addition statements.

### Are we on track?

**Following is a portion of the JCL for a procedure named MYPROC.**

```
//PSTEP1          EXEC PGM=PROGA,TIME=(1,30)
//DD1             DD      ...
//PSTEP2          EXEC PGM=PROGB,TIME=5
```

**Assume that for this invocation of MYPROC, you wish to add the date 01/11/99 to PROGA and PROGB. (You do so through the PARM parameter.)**

**Code the statement to invoke MYPROC.**

## Addition statements.

Are we on track?

Match the change types below with their descriptions.

- |                    |   |
|--------------------|---|
| <b>1. Override</b> | <b>A. Append existing code with new parameters.</b>     |
| <b>2. Nullify</b>  | <b>B. Change a parameter back to its default value.</b> |
| <b>3. Addition</b> | <b>C. Change an existing value.</b>                     |



## Addition statements.

**Are we on track?**

**Which of the following actions can you perform when a procedure is invoked for use?**

- A. Override the PGM=parameter on one or more procedure EXEC statements.**
- B. Override operands, such as ACCT, on procedure EXEC statements.**
- C. Nullify dataset specifications on procedure DD statements.**
- D. Add data specifications on procedure DD statements.**
- E. Permanently alter the JCL in a cataloged procedure.**

## Sequencing multiple changes.

### Combining changes.

**It is possible to use EXEC statement overrides, nullifications, and additions for one or more procedure steps at the same time. This can be done by combining the changes on the EXEC statement that invokes the procedure.**

**For example, the programmer can change the time restrictions and can also supply the current date for a particular PSTEP.**

## Sequencing multiple changes.

# Sequencing multiple changes.

Many changes can be made to EXEC statement parameters for one or more procedure steps by combining them on the EXEC statement you use to invoke the procedure.

The following rules must be followed while sequencing multiple changes:

- **Specify alterations in procedure step sequence. The alterations for one step must be specified before the alterations for a subsequent step.**
- **Within any one step, alterations can be specified in any sequence.**
- **Alterations should be separated from each other by a comma.**
- **Multiple changes must be coded in procedure step sequence.**

## Sequencing multiple changes.

# Sequencing multiple changes – an example.

Consider the TRANSACT procedure.  
The following alterations are to be made to the EXEC statement operands in the procedure:

- Increase the time restriction for PSTEP1 to 3 minutes.
- Revert to the installation-defined TIME default for PSTEP2.
- Add a PARM parameter value of 01/29/99 for the EXEC statements in PSTEP1 and PSTEP2.

```
//JSTEP EXEC TRANSACT,TIME.PSTEP1=3,  
//          PARM.PSTEP1=' 01/29/99' ,  
//          TIME.PSTEP2=,  
//          PARM.PSTEP2=' 01/29/99'
```

## Sequencing multiple changes.

### Are we on track?

**For the invocation of MYPROC, assume you want to make the following changes:**

- a. Revert to the installation-defined CPU time limit for PROGB.**
- b. Restrict the amount of time PROGA can use the CPU to 2 minutes.**
- c. Add a PARM value of '5/10/99' to PROGA.**

**Put the following items in the correct sequence, following the sequencing rules for multiple changes to produce EXEC statement parameters.**

**A. //                            PARM.PSTEP1='5/10/99'**

**B. //                            TIME.PSTEP1=2**

**C. //JSTEP            EXEC MYPROC,**

**D. //                            TIME.PSTEP2=**



## Modifying EXEC parameters.

### Unit summary.

Now that you have completed this unit, you should be able to:

- **Specify what factors determine if a procedure meets the requirements of a job.**
- **Invoke a procedure, making temporary alterations to it if necessary.**
- **Add, override or nullify parameters on procedure step EXEC statements.**
- **Correctly sequence multiple changes to EXEC statement parameters.**

# **JCL**

## **Chapter b3 Modifying EXEC parameters**

## **Job Control Language**

**Chapter a1. Introduction to JCL**

**Chapter a2. Coding JOB statements**

**Chapter a3. Coding EXEC statements**

**Chapter a4. Coding DD statements**

**Chapter a5. Analyzing job output**

**Chapter a6. Conditional processing**



## Job Control Language

**Chapter b1. Using special DD statements**

**Chapter b2. Introducing procedures**

**Chapter b3. Modifying EXEC parameters**

**Chapter b4. Modifying DD parameters**

**Chapter b5. Determining the effective JCL**

**Chapter b6. Symbolic parameters**

## **Job Control Language**

**Chapter c1. Nested procedures**

**Chapter c2. Cataloging procedures**

**Chapter c3. Using utility programs**

**Chapter c4. Sample utility application**

**Modifying EXEC parameters.**

## **Chapter b3**

# **Modifying EXEC parameters**

**5**

Copyright © 2006 CA. All trademarks, trade names, services marks and logos referenced herein belong to their respective companies.

**Modifying EXEC parameters.**

## **Unit introduction.**

**Before using a procedure it is essential to ensure that it meets all the processing requirements. This might involve some minor alterations to the procedure to ensure that it has all the requirements to execute the job.**

**This unit describes how to alter EXEC statement parameters at the time of job submission. The information contained in this unit can be applied to both in-stream and cataloged procedures.**

**Modifying EXEC parameters.**

**Course objectives.**

**Be able to:**

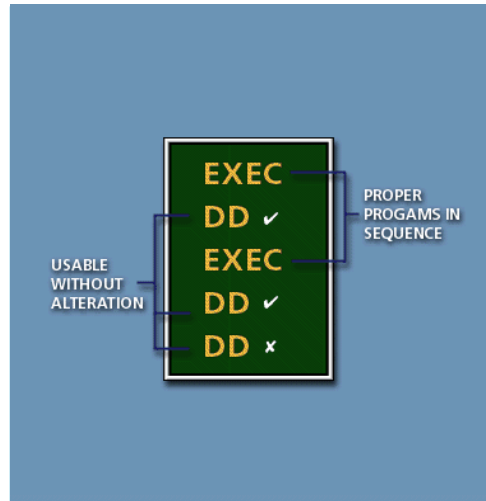
- **Specify what factors determine if a procedure meets the requirements of a job.**
- **Invoke a procedure, making temporary alterations to it if necessary.**
- **Add, override or nullify parameters on procedure step EXEC statements.**
- **Correctly sequence multiple changes to EXEC statement parameters.**

## Analyzing procedures.

### Identifying analysis criteria.

A procedure listing is obtained and examined to ensure that it meets the following two criteria:

- The procedure invokes the proper programs in the desired sequence.
- Most of the DD statements are usable without major alteration.



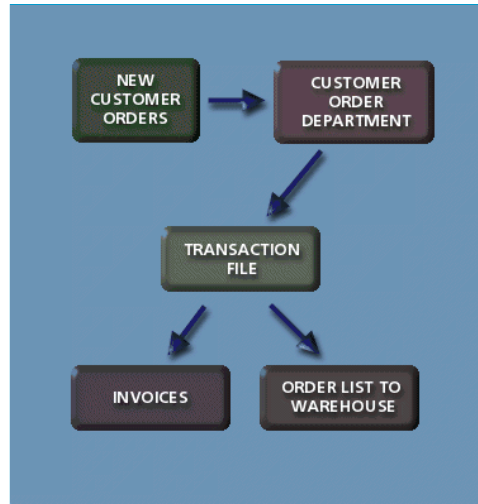
### Analyzing procedures.

#### Identifying analysis criteria – an example.

Consider a case in which a company buys goods wholesale from several manufacturers and markets them retail to other customers.

Each week the customer order department creates a transaction file that contains new customer orders.

A list of orders is sent to the warehouse and an invoice is sent to each customer. The order list and associated invoices are printed once a week.



Continued... ➡

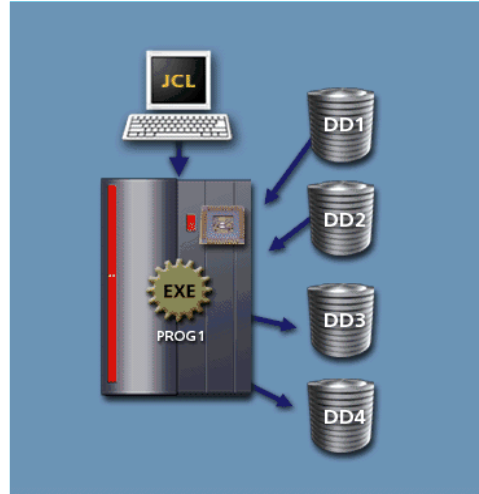
### Analyzing procedures.

#### Identifying analysis criteria – an example.

A program PROG1 checks the weekly input transactions against entries in a master customer data set. Valid transactions are written to a new data set used as input for another program PROG2.

PROG1 refers to the following data sets:

- DD1: Input transactions.
- DD2: Master customer data set.
- DD3: Transaction exception report.
- DD4: Set of valid transactions that are passed to PROG2.



10

Copyright © 2006 CA. All trademarks, trade names, services marks and logos referenced herein belong to their respective companies.

Continued... ➔

Master customer data set ensures that each transaction applies to a valid customer. Any invalid transactions are written to a transaction exception report.



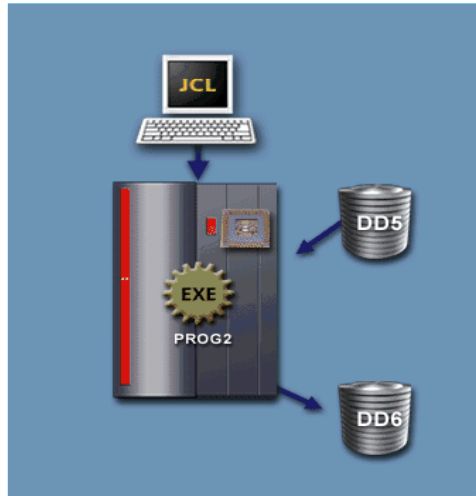
## Analyzing procedures.

### Identifying analysis criteria – an example.

PROG2 reads the valid transactions passed from PROG1 and creates an order list/invoice for each customer.

PROG2 refers to the following data sets:

- DD5: Set of valid transactions passed from PROG1.
- DD6: Order list/invoice for each customer.



## Analyzing procedures.

### Identifying analysis criteria – an example.

Shown on the right is the procedure named TRANSACT that is used to accomplish this task.

A procedure listing is obtained to determine:

- If it uses the required programs in the required sequence.
- If it uses appropriate data sets.

```
//PSTEP1 EXEC PGM=PROG1,TIME=(1,30)
//DD1 DD DSN=INTRAN,DISP=SHR
//DD2 DD DSN=MASTER,DISP=SHR
//DD3 DD SYSOUT=A
//DD4 DD DSN=%%VALID,
// UNIT=SYSDA,
// DISP=(NEW,PASS),
// SPACE=(TRK,(1,1))
//PSTEP2 EXEC PGM=PROG2,TIME=5
//DD5 DD DSN=%%VALID,
// DISP=(OLD,DELETE)
//DD6 DD SYSOUT=A
```

**Analyzing procedures.**

**Are we on track?**

**Based on the analysis of the JCL in the previous example, do you think this is an appropriate procedure for the task, as described?**

- A. No, because not all data sets are taken into consideration.**
- B. There is not enough information to decide.**
- C. Yes, because both criteria are met.**

The correct answer is C.

## Analyzing procedures.

### Analysis explanation.

The listing for the procedure TRANSACT indicates the following:

The procedure executes 2 programs: PROG1 and PROG2.

PROG1 uses the following data sets:

- A cataloged data set INTRAN (DD1 DD statement).
- A cataloged data set MASTER (DD2 DD statement).
- SYSOUT class A output (DD3 DD statement).
- A temporary data set &&VALID that is passed to PROG2 (DD4 DD statement).

```
//PSTEP1 EXEC PGM=PROG1,TIME=(1,30)
//DD1 DD DSN=INTRAN,DISP=SHR
//DD2 DD DSN=MASTER,DISP=SHR
//DD3 DD SYSOUT=A
//DD4 DD DSN=&&VALID,
// UNIT=SYSDA,
// DISP=(NEW,PASS),
// SPACE=(TRK,(1,1))
//PSTEP2 EXEC PGM=PROG2,TIME=5
//DD5 DD DSN=&&VALID,
// DISP=(OLD,DELETE)
//DD6 DD SYSOUT=A
```

## Analyzing procedures.

### Analysis explanation.

PROG2 uses the following data sets:


- A temporary data set named &&VALID, which is passed from PROG1 (DD5 DD statement).
- SYSOUT class A output (DD6 DD statement).

In addition, a TIME parameter is included on the PSTEP1 and PSTEP2 EXEC statements to restrict the amount of time the programs are permitted to use the central processor.

15

Copyright © 2006 CA. All trademarks, trade names, services marks and logos referenced herein belong to their respective companies.

```
//PSTEP1 EXEC PGM=PROG1,TIME=(1,30)
//DD1 DD DSN=INTRAN,DISP=SHR
//DD2 DD DSN=MASTER,DISP=SHR
//DD3 DD SYSOUT=A
//DD4 DD DSN=&&VALID,
// UNIT=SYSDA,
// DISP=(NEW,PASS),
// SPACE=(TRK,(1,1))
//PSTEP2 EXEC PGM=PROG2,TIME=5
//DD5 DD DSN=&&VALID,
// DISP=(OLD,DELETE)
//DD6 DD SYSOUT=A
```

Continued... 

TIME[.procstepname]= {[minutes][,seconds]}

{1440 }

{NOLIMIT }

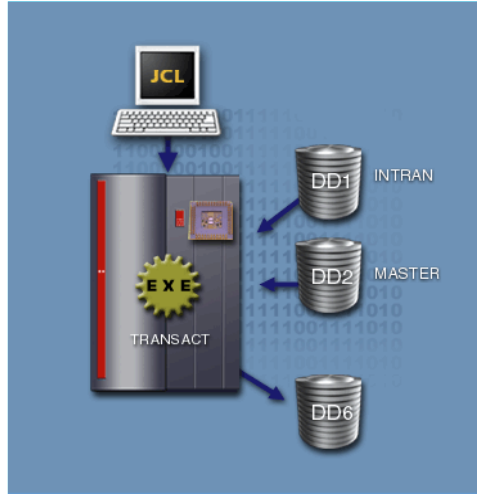
{MAXIMUM }

{0 }

### Analyzing procedures.

#### Analysis explanation.

The TRANSACT procedure listing is obtained and evaluated to ensure that the procedure executes all the required programs in the proper sequence and the appropriate data sets are used to accomplish the application.



It is assumed that data sets INTRAN and MASTER already exist and are cataloged.

Analyzing procedures.

Are we on track?

Code a statement to invoke TRANSACT.

//JSTEP EXEC \_\_\_\_\_

17

Copyright © 2006 CA. All trademarks, trade names, services marks and logos referenced herein belong to their respective companies.

The correct answer is TRANSACT.

**Analyzing procedures.**

**Are we on track?**

**A procedure must meet which of the following criteria?**

- A. Most of the DD statements are usable as they are or might need some minor alteration.**
- B. The programs in the procedure are located in the same library.**
- C. It invokes the proper programs in the correct sequence.**
- D. The DD statements point to a single storage volume.**

The correct answer is A. and C.



Analyzing procedures.

## **Glossary.**

### **Cataloged data set**

**A non-temporary data set for which the system has recorded the unit and volume on which the data set resides.**

### **SYSOUT**

**A keyword that defines a print data set. It instructs the system to queue the output on a direct-access volume.**

### **Temporary data set**

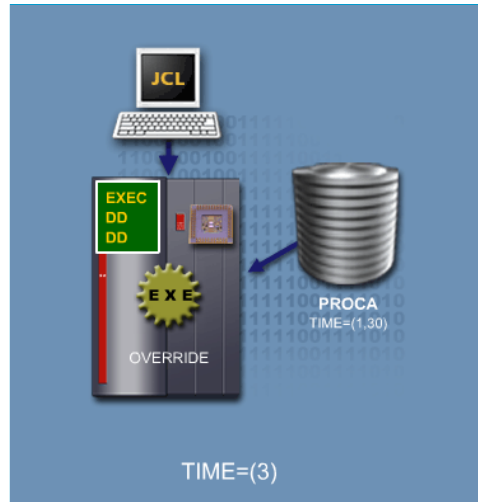
**A data set which stores data needed only for the duration of the job.**

## Changing EXEC parameters.

### Coding changes.

A procedure listing helps a programmer to analyze the procedure for its usability. In some cases a procedure might satisfy all the basic requirements, but might need some minor alterations.

This can be done by changing the EXEC and DD parameters when the procedure is invoked. However, these alterations are applicable only for one invocation. They do not permanently modify the procedure definition.



20

Copyright © 2006 CA. All trademarks, trade names, services marks and logos referenced herein belong to their respective companies.

Continued... ➔

Some of the minor alterations to a procedure may involve, changing the data set names or the time for which the program uses the CPU.

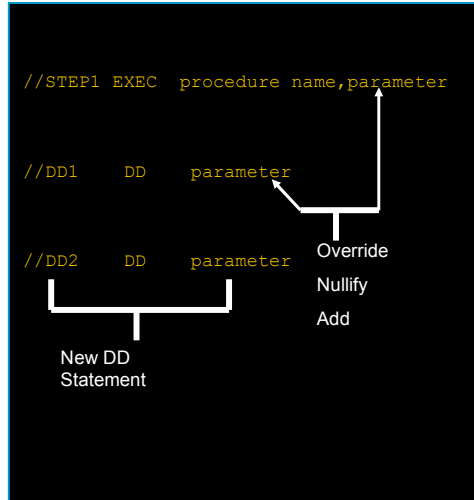
## Changing EXEC parameters.

### Coding changes.

Changes can be made to procedure EXEC statement parameters such as TIME, ACCT, and PARM.

The programmer can change these parameters in the following ways:

- Override the parameters on the procedure EXEC statement.
- Nullify parameters on the procedure EXEC statement.
- Add parameters to the procedure EXEC statement.



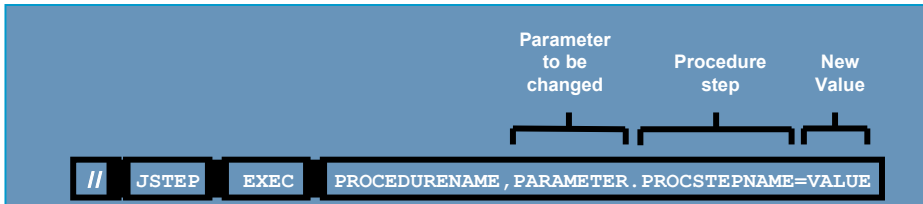
21

Copyright © 2006 CA. All trademarks, trade names, services marks and logos referenced herein belong to their respective companies.

These parameters specify such things as CPU processing time, account information or date parameters.

## Changing EXEC parameters.

### Coding changes to EXEC statement parameters.



The general form for coding changes to EXEC statement parameters is as follows (it is shown above):

- To modify EXEC statement parameters for any procedure step, append the procedure step to the parameter.
- If the stepname is omitted, the parameter applies to all steps of the procedure, with the exception of the PARM parameter.
- If the stepname is omitted when adding or overriding a PARM parameter, the PARM value only applies to the first step in the procedure.
- Any PARM parameters in subsequent steps within the procedure are nullified.

## Changing EXEC parameters.

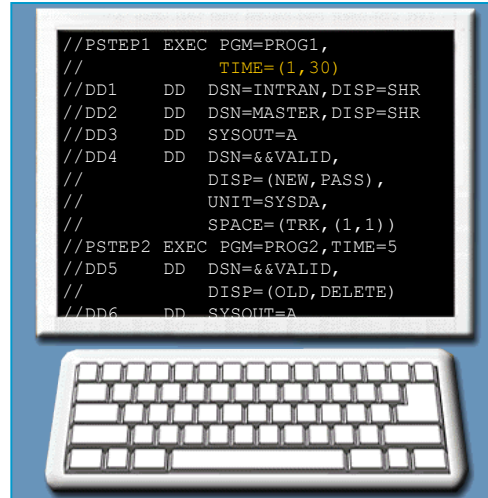
### Changing EXEC statement parameters – an example.

To illustrate an EXEC statement override, review the procedure definition for TRANSACT, as shown on the right.

The time allocated for TRANSACT is 1 minute 30 seconds.

If the transaction file for the week were much larger than usual, you might want to change the time allocated for the procedure to 3 minutes. You would code the following override statement:

```
//JSTEP EXEC TRANSACT,TIME.PSTEP1=3
```



### Changing EXEC parameters.

#### Implementing coding changes.

General syntax for changes to EXEC statement parameters is as follows:

```
//JSTEP      EXEC  procedurename,  
//          parameter.procstepname=value
```

Implementing coding changes to EXEC statements involves the following steps:

- Follow the name of the procedure with a coma.
- Give the name of the EXEC statement parameter to be overridden, nullified or added, followed by a period.
- Give the name of the procedure step, followed by an equal sign.
- Give the new value for the parameter if you are overriding or adding a value. Do not code a value if the parameter is to be nullified.

Changes are specified in the EXEC statement that invokes the procedure.

You can override, nullify, or add EXEC statement parameters such as PARM or TIME on the EXEC statement to invoke the procedure. One exception is the PGM parameter, which is the only EXEC statement parameter that cannot be overridden or nullified. The only way to execute a procedure with a different program is to assign a value to a symbolic PGM parameter.

**Changing EXEC parameters.**

**Are we on track?**

**Changing parameters at the time you invoke a procedure has what effect?**

**A. Changes apply to the number of invocations you specify in the JCL.**

**B. Changes apply to all procedures containing the edited parameters.**

**C. Changes apply once only to the current invocation.**

**D. Changes apply to all future invocations of the procedure.**

25

Copyright © 2006 CA. All trademarks, trade names, services marks and logos referenced herein belong to their respective companies.

The correct answer is C.

**Changing EXEC parameters.**

**Are we on track?**

**Which of the following changes can you make at the time you execute a procedure?**

**A. Temporarily add operands such as ACCT to procedure EXEC statements.**

**B. Alter the library copy of the JCL contained in cataloged procedure.**

**C. Override the PGM parameter on procedure EXEC.**

26

Copyright © 2006 CA. All trademarks, trade names, services marks and logos referenced herein belong to their respective companies.

The correct answer is A.



## Override statements.


### Overriding statement parameters.

An override statement is used to change an existing parameter value.

Consider the TRANSACT procedure which definition is shown on the right. Note that the time that PROG1 can run is 1 minute 30 seconds.

Assume that for a particular week, the transaction file to be processed is too large and the time that PROG1 can run needs to be increased to 3 minutes.

```
//PSTEP1 EXEC PGM=PROG1, TIME= (1, 30)
//DD1 DD DSN=INTRAN, DISP=SHR
//DD2 DD DSN=MASTER, DISP=SHR
//DD3 DD SYSOUT=A
//DD4 DD DSN=&&VALID,
// DISP= (NEW, PASS) ,
// UNIT=SYSDA,
// SPACE= (TRK, (1, 1) )
//PSTEP2 EXEC PGM=PROG2, TIME=5
//DD5 DD DSN=&&VALID,
// DISP= (OLD, DELETE)
//DD6 DD SYSOUT=A
```

Continued... 

## Override statements.

### Overriding statement parameters.

To override the original time parameter, the TRANSACT can be invoked with the following EXEC statement:

```
//JSTEP EXEC TRANSACT,TIME.PSTEP1=3
```

The resulting JCL would behave as the procedure on the right. Note the new parameter in the resulting JCL. However, this override is only temporary. The procedure definition does not change. The next time the procedure is invoked, it will revert to the original definition.

```
//PSTEP1 EXEC PGM=PROG1,TIME=3
//DD1 DD DSN=INTRAN,DISP=SHR
//DD2 DD DSN=MASTER,DISP=SHR
//DD3 DD SYSOUT=A
//DD4 DD DSN=&&VALID,
// DISP=(NEW,PASS),
// UNIT=SYSDA,
// SPACE=(TRK,(1,1))
//PSTEP2 EXEC PGM=PROG2,TIME=5
//DD5 DD DSN=&&VALID,
// DISP=(OLD,DELETE)
//DD6 DD SYSOUT=A
```

**Override statements.**

**Are we on track?**

**Following is a portion of the JCL for a procedure named MYPROC.**

```
//PSTEP1      EXEC PGM=PROGA, TIME=(3,30)
//DD1         DD  DSN=A, DISP=SHR
//DD2         DD  ...
//PSTEP2      EXEC PGM=PGMB, TIME=5
//DD3         DD  ...
```

**Code a statement to invoke the procedure named MYPROC.  
Assume you want to restrict the amount of time PROGA is  
permitted to use the CPU to 2 minutes.**

**//JSTEP EXEC \_\_\_\_\_**

29

Copyright © 2006 CA. All trademarks, trade names, services marks and logos referenced herein belong to their respective companies.

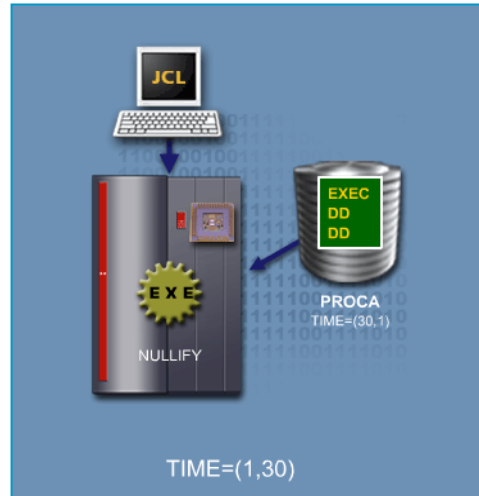
The correct answer is MYPROC,TIME.PSTEP1=2

### Nullification statements.

## Nullifying EXEC statement parameters.

A procedure can be modified by nullifying an EXEC statement parameter.

Most installations have values that are assigned to EXEC statement parameters automatically. For example, a default value may be assigned for the TIME parameter. The default values may be overridden when the procedure is defined. To return to the installation's default value, the programmer can code a statement that nullifies the parameter.



30

Copyright © 2006 CA. All trademarks, trade names, services marks and logos referenced herein belong to their respective companies.

To nullify a parameter means to change it back to the installation-defined default.

**Nullification statements.**

**Nullifying EXEC statement parameters.**

**The format for nullifying an EXEC statement parameter is:**

```
//JSTEP EXEC procedurename,  
//           parameter.procstepname=
```

**Note that in the format, the programmer specifies the parameter and the procedure step in which it appears.**

**Also note that no value is assigned to the parameter in the nullifying EXEC statement.**

### Nullification statements.

## Nullifying EXEC statement parameters – an example.

Consider the TRANSACT procedure.

The procedure definition for PROG1 (in the procedure step PSTEP1) has specified a CPU time of 1 minute 30 seconds for processing a transaction file. This processing time may not be adequate for a larger file.

If the default time is adequate, the programmer might want to execute the procedure taking the system default time for PROG1.

```
//PSTEP1 EXEC PGM=PROG1, TIME=(1,30)
//DD1 DD DSN=INTRAN, DISP=SHR
//DD2 DD DSN=MASTER, DISP=SHR
//DD3 DD SYSOUT=A
//DD4 DD DSN=&&VALID,
// DISP=(NEW, PASS),
// UNIT=SYSDA,
// SPACE=(TRK, (1,1))
//PSTEP2 EXEC PGM=PROG2, TIME=5
//DD5 DD DSN=&&VALID,
// DISP=(OLD, DELETE)
//DD6 DD SYSOUT=A
```

### Nullification statements.

## Nullifying EXEC statement parameters – an example.

To do this, the programmer needs to nullify the TIME specified in the procedure definition on the PSTEP1 EXEC statement.

The following EXEC statement which invokes TRANSACT would nullify the time parameter:

```
//JSTEP1 EXEC TRANSACT,TIME.PSTEP1=
```

The resulting JCL would behave as the procedure on the right.

```
//PSTEP1 EXEC PGM=PROG1
//DD1 DD DSN=INTRAN,DISP=SHR
//DD2 DD DSN=MASTER,DISP=SHR
//DD3 DD SYSOUT=A
//DD4 DD DSN=&&VALID,
// DISP=(NEW,PASS),
// UNIT=SYSDA,
// SPACE=(TRK,(1,1))
//PSTEP2 EXEC PGM=PROG2,TIME=5
//DD5 DD DSN=&&VALID,
// DISP=(OLD,DELETE)
//DD6 DD SYSOUT=A
```

**Nullification statements.**

**Are we on track?**

**Following is a portion of the JCL for a procedure named MYPROC.**

```
//PSTEP1      EXEC PGM=PROGA  
//DD1        DD      ...  
//PSTEP2      EXEC PGM=PROGB,TIME=5  
//DD2        DD      ...
```

**Assume that for this use of the procedure, you want to return to the installation-defined CPU time limit for PROGB. Code the statement to invoke MYPROC.**

```
//JSTEP EXEC _____
```

34

Copyright © 2006 CA. All trademarks, trade names, services marks and logos referenced herein belong to their respective companies.

The correct answer is MYPROC,TIME.PSTEP2=



**Addition statements.**

**Adding parameters to a procedure.**

**Addition statements are used to add parameters to a procedure. The programmer can code additions on the EXEC statement that invokes the procedure.**

**Consider the TRANSACT procedure. Assume that a programmer wants to supply the current date for PROG1.**

**This parameter can be coded in the following way while invoking the procedure:**

```
//JSTEP EXEC procedurename,parameter.procstepname=value
```

```
//JSTEP EXEC TRANSACT,PARM.PSTEP1='01/29/99'
```



**Addition**

**Addition statements.**

**Adding parameters to a procedure.**

Parameters can be added to one or more procedure steps. In the TRANSACT procedure, the current date can be supplied to PROG1 and PROG2 by means of the PARM parameter.

The code would be as follows:

```
//JSTEP EXEC TRANSACT,  
//          PARM.PSTEP1='01/29/99',  
//          PARM.PSTEP2='01/29/99'
```

Note the following points when sequencing multiple EXEC statement additions:

- The additions are in procedure step sequence.
- A comma separates the name of the procedure from the first parameter addition, and the parameter additions from each other.

36

Copyright © 2006 CA. All trademarks, trade names, services marks and logos referenced herein belong to their respective companies.

Note that in the example, the additions are done in step sequence. PSTEP1 additions are specified before PSTEP2 additions. Also a comma separates TRANSACT from the first parameter addition. Both the parameter additions are also separated by a comma.

The value of the PARM parameter in the above example is a date string. However, any character string can be assigned to a PARM parameter.

### Addition statements.

## Adding parameters to a procedure.

The TRANSACT procedure definition is shown on the right.

```
//PSTEP1 EXEC PGM=PROG1, TIME= (1, 30)
//DD1 DD DSN=INTRAN, DISP=SHR
//DD2 DD DSN=MASTER, DISP=SHR
//DD3 DD SYSOUT=A
//DD4 DD DSN=&&VALID,
// DISP= (NEW, PASS) ,
// UNIT=SYSDA,
// SPACE= (TRK, (1, 1))
//PSTEP2 EXEC PGM=PROG2, TIME=5
//DD5 DD DSN=&&VALID,
// DISP= (OLD, DELETE)
//DD6 DD SYSOUT=A
```



### Addition statements.

## Adding parameters to a procedure.

The following EXEC statement supplies the current date for PROG1 and PROG2:

```
//JSTEP EXEC TRANSACT,  
//          PARM.PSTEP1='01/29/91',  
//          PARM.PSTEP2='01/29/91'
```

The resulting JCL would behave as the procedure on the right.

```
//PSTEP1 EXEC PGM=PROG1,TIME=(1,30),  
//          PARM='01/29/91'  
//DD1 DD DSN=INTRAN,DISP=SHR  
//DD2 DD DSN=MASTER,DISP=SHR  
//DD3 DD SYSOUT=A  
//DD4 DD DSN=&&VALID,  
//          DISP=(NEW,PASS),  
//          UNIT=SYSDA,  
//          SPACE=(TRK,(1,1))  
//PSTEP2 EXEC PGM=PROG2,TIME=5,  
//          PARM='01/29/91'  
//DD5 DD DSN=&&VALID,  
//          DISP=(OLD,DELETE)  
//DD6 DD SYSOUT=A
```



**Addition statements.**

**Are we on track?**

**Following is a portion of the JCL for a procedure named MYPROC.**

```
//PSTEP1      EXEC PGM=PROGA, TIME=(1,30)
//DD1         DD   ...
//PSTEP2      EXEC PGM=PROGB, TIME=5
```

**Assume that for this invocation of MYPROC, you wish to add the date 01/11/99 to PROGA and PROGB. (You do so through the PARM parameter.)**

**Code the statement to invoke MYPROC.**

39

Copyright © 2006 CA. All trademarks, trade names, services marks and logos referenced herein belong to their respective companies.

The correct answer is:

```
//JSTEP EXEC MYPROC,
//          PARM.PSTEP1='01/11/99',
//          PARM.PSTEP2='01/11/99'
```

**Addition statements.**

**Are we on track?**

**Match the change types below with their descriptions.**

- |                    |   |
|--------------------|---|
| <b>1. Override</b> | <b>A. Append existing code with new parameters.</b>     |
| <b>2. Nullify</b>  | <b>B. Change a parameter back to its default value.</b> |
| <b>3. Addition</b> | <b>C. Change an existing value.</b>                     |

40

Copyright © 2006 CA. All trademarks, trade names, services marks and logos referenced herein belong to their respective companies.

The correct answer is 1-C, 2-B, and 3-A.

**Addition statements.**

**Are we on track?**

**Which of the following actions can you perform when a procedure is invoked for use?**

**A. Override the PGM=parameter on one or more procedure EXEC statements.**

**B. Override operands, such as ACCT, on procedure EXEC statements.**

**C. Nullify dataset specifications on procedure DD statements.**

**D. Add data specifications on procedure DD statements.**

**E. Permanently alter the JCL in a cataloged procedure.**

41

Copyright © 2006 CA. All trademarks, trade names, services marks and logos referenced herein belong to their respective companies.

The correct answer is B., C., and D.

**Sequencing multiple changes.**

### **Combining changes.**

**It is possible to use EXEC statement overrides, nullifications, and additions for one or more procedure steps at the same time. This can be done by combining the changes on the EXEC statement that invokes the procedure.**

**For example, the programmer can change the time restrictions and can also supply the current date for a particular PSTEP.**



### Sequencing multiple changes.

#### Sequencing multiple changes.

Many changes can be made to EXEC statement parameters for one or more procedure steps by combining them on the EXEC statement you use to invoke the procedure.

The following rules must be followed while sequencing multiple changes:

- **Specify alterations in procedure step sequence. The alterations for one step must be specified before the alterations for a subsequent step.**
- **Within any one step, alterations can be specified in any sequence.**
- **Alterations should be separated from each other by a comma.**
- **Multiple changes must be coded in procedure step sequence.**

43

Copyright © 2006 CA. All trademarks, trade names, services marks and logos referenced herein belong to their respective companies.

Parameters with a stepname appended must appear before any parameters coded without appended stepnames.

## Sequencing multiple changes.

### Sequencing multiple changes – an example.

Consider the TRANSACT procedure.  
The following alterations are to be made to the EXEC statement operands in the procedure:

- Increase the time restriction for PSTEP1 to 3 minutes.
- Revert to the installation-defined TIME default for PSTEP2.
- Add a PARM parameter value of 01/29/99 for the EXEC statements in PSTEP1 and PSTEP2.

```
//JSTEP EXEC TRANSACT, TIME.PSTEP1=3,  
//          PARM.PSTEP1='01/29/99',  
//          TIME.PSTEP2=  
//          PARM.PSTEP2='01/29/99'
```

44

Copyright © 2006 CA. All trademarks, trade names, services marks and logos referenced herein belong to their respective companies.

The JCL statements on the right show how to combine these alterations on the EXEC statement that invokes the procedure.

Note that the nullify for the TIME parameter of PSTEP2 is immediately followed by a comma, before the next parameter.

Changes for procedure step 1 are coded before changes to procedure step 2.

Sequencing multiple changes.

**Are we on track?**

**For the invocation of MYPROC, assume you want to make the following changes:**

- a. Revert to the installation-defined CPU time limit for PROGB.**
- b. Restrict the amount of time PROGA can use the CPU to 2 minutes.**
- c. Add a PARM value of '5/10/99' to PROGA.**

**Put the following items in the correct sequence, following the sequencing rules for multiple changes to produce EXEC statement parameters.**

- A. //                    PARM.PSTEP1='5/10/99'**
- B. //                    TIME.PSTEP1=2**
- C. //JSTEP    EXEC MYPROC,**
- D. //                    TIME.PSTEP2=**

45 Copyright © 2006 CA. All trademarks, trade names, service marks, and registered trademarks are the property of their respective companies.

The correct order is C., B., A., and D.

### Modifying EXEC parameters.

#### Unit summary.

Now that you have completed this unit, you should be able to:

- **Specify what factors determine if a procedure meets the requirements of a job.**
- **Invoke a procedure, making temporary alterations to it if necessary.**
- **Add, override or nullify parameters on procedure step EXEC statements.**
- **Correctly sequence multiple changes to EXEC statement parameters.**