

# Dynamic Resource Definition (DRD)



© 2008 IBM Corporation

## Dynamic Resource Definition (DRD) Overview


- DRD provides type-2 commands to create, delete or update database, program, transaction and routing code definitions in an online IMS system
  - ◆ CREATE
    - Add the IMS control block representing the resource to the online IMS environment
  - ◆ DELETE
    - Remove the IMS control block from the IMS environment
  - ◆ UPDATE
    - Change the attributes of the resource
- Definitions are "permanent"
  - ◆ They are written to the IMS log and restored on warm and emergency restarts
  - ◆ They may be exported and imported from a Resource Definition Data Set (RDDS)
    - EXPORT – at IMS system checkpoint time
    - IMPORT - during IMS cold start
- QUERY commands may be used to show definitions

2

DRD is implemented with new and enhanced type-2 commands. The CREATE command is used to add (create) a new resource control block in a running IMS environment. The DELETE command is used to delete existing control blocks. The UPDATE command is used to update resource attributes.

The definitions added by DRD are typically permanent. They are always permanent across warm and emergency restarts. All definitions are written to the IMS log. This includes those created by the system definition (sysgen) process and those created by DRD. When an IMS system is warm started or emergency restarted, these definitions are read from the log. Definitions added by DRD may be permanent across cold starts. They may be exported for use by subsequent cold starts. Exports are done as part of system checkpoints. They are written to Resource Definition Data Sets (RDDS). When a cold start is done, IMS optionally reads the definitions from an RDDS. We will see later that the use of RDDSs is optional. If they are not used, cold starts will not restore definitions added by DRD.

Finally, QUERY commands may be used to show all resources including those created or updated by DRD.



Information On Demand

IMS Version 10

## Environment for DRD

- Required
  - ◆ IMS PROCLIB members with DRD enablement parameters
    - DFSDFXxx (new in IMS V10)
  - ◆ Common Service Layer with SCI and Operations Manager (OM)
    - Resource Manager (RM) is not required
  - ◆ OM client for entering DRD commands
    - SPOC, IMS Control Center, or other OM interface
- Optional, but recommended
  - ◆ Data sets to hold resource definitions
    - Resource Definition Data Sets (RDDs)
- Alternative to Online Change for MODBLKS
  - ◆ If DRD is enabled, online change cannot be used for MODBLKS

3

You must define DRD to IMS by including DRD parameters the new DFSDFXxx PROCLIB member.

Since type-2 commands are used, you must have a Common Service Layer (CSL) including SCI and OM. RM is not required. You also must have an OM client through which you may enter the commands. This can be the IMS TSO SPOC, the IMS Control Center, or any other OM client.

RDDs data sets are recommended. They are used for exporting and importing definitions across IMS cold starts. They also may be used by the extract utility to move the definitions to another system.

DRD is supported in all IMS environments, including DB/DC, DBCTL, and DCCTL. It is also supported in a data sharing environment and a shared queues environment. Care must be taken in a multi-IMS environment to define resources consistently across all IMSs

When DRD is implemented, online change cannot be used for MODBLKS. Online change remains available for ACBLIB and FORMAT.

When DRD is fully implemented, there is no longer a need for a MODBLKS system definition. MODBLKS system definitions may be used when DRD is implemented, but you may choose to eliminate it and define all MODBLKS resources (programs, transactions, routing codes, and databases) using DRD commands. The MODBLKS libraries associated with the system definition and online change processes may be deleted when only DRD is used to define these resources.



## Defining Resources with DRD

# Resources

- IMS resources defined by DRD

| RESOURCE TYPE                       | SYSGEN MACRO                            | CONTROL BLOCK                             |
|-------------------------------------|---|---|
| <b>Database</b>                     | <b>DATABASE</b><br>(DB/TM, DBCTL)       | <b>DDIR</b><br>(Database Directory)       |
| <b>Application Program</b><br>(PSB) | <b>APPLCTN</b><br>(DB/TM, DBCTL, DCCTL) | <b>PDIR</b><br>(Program Directory)        |
| <b>Transaction</b>                  | <b>TRANSACT</b><br>(DB/TM, DCCTL)       | <b>SMB</b><br>(Scheduler Message Block)   |
| <b>Routing Code</b>                 | <b>RTCODE</b><br>(DB/TM, DCCTL)         | <b>RCTE</b><br>(Routing Code Table Entry) |

- ◆ The control blocks created with DRD are the same as those created by system definition macros
- Terminology
  - ◆ *Runtime resource definitions* - internal control blocks that IMS maintains and uses
  - ◆ *Stored resource definitions* - information stored offline such as in an RDDS or MODBLKS

The resources that may be defined by DRD are shown here. The control blocks are the same as those that are created by the system definition process.

IMS publications use two new terms to describe the difference between definitions that IMS uses while running versus those that are stored outside of a running IMS system.

- Runtime resource definitions - these are the internal control blocks that IMS maintains and uses
- Stored resource definitions - this is resource information stored offline such as in an RDDS or MODBLKS

These are new terms, however, they describe a difference that has always existed in IMS. For example, the MODBLKS data set contains stored resource definitions that IMS uses to create runtime resource definitions when it is executed. The new terms are not limited to use with DRD.



## Commands Used in DRD

- Type-2 commands entered through OM interface to
  - ◆ CREATE, UPDATE, DELETE, and QUERY resources and descriptors

| Command | Short Form | Purpose             |
|---------|------------|---------------------|
| CREATE  | CRE        | Creates definitions |
| DELETE  | DEL        | Deletes definitions |
| UPDATE  | UPD        | Updates definitions |
| QUERY   | QRY        | Queries definitions |

6

There are four commands used with DRD.

- CREATE – This command is used to create definitions
- DELETE – This command is used to delete definitions
- UPDATE – This command is used to update definitions
- QUERY – This command is used to show definitions

The commands may have other uses. For example, the UPDATE command may be used to change a status and the QUERY command may be used to show a status. Statuses are discussed later.



## Resource Attributes and Statuses

- Attributes are defined for a resource
  - ◆ Example macro from System Definition

```
DATABASE DBD=ACCTMSTR,RESIDENT,ACCESS=UP
```
  - ◆ Example command with DRD

```
CREATE DB NAME(ACCTMSTR)SET(RESIDENT(Y),ACCTYPE(UPD))
```
  - ◆ RESIDENT and UPDATE access are attributes of the ACCTMSTR database
  
- Statuses are set or changed during execution of the system
  - ◆ Examples of commands

```
/STOP TRAN ABC
/DBR DB ACCTMSTR
```
  - ◆ Example of IMS changing a status
    - IMS stops a transaction if the application abends

7

Resources have attributes and statuses.

Attributes are defined for a resource. They are specified either on a system definition macro or on a DRD command.

Statuses are not part of definitions. They indicate the current availability of a resource. For example, a transaction or database may be stopped or started. The /STA DB, /STOP DB, /DBD DB and /DBR commands are examples of commands that change the status of a database. IMS may change the status of a resource without a command. For example, when an application program abends IMS may stop the transaction.

## CREATE Command

- Creates resource definitions or descriptors
  - ◆ Programs, transactions, routing codes, and databases
    - Appropriate attributes for each resource type may be set or defaulted
    - SET used to specify attribute values
- Examples:

```
CREATE PGM NAME(PGM1) SET(RESIDENT(Y))  
  
CREATE TRAN NAME(TRN1) SET(PGM(PGM1) CLASS(4))  
  
CRE      DB NAME(DB001) SET(RESIDENT(Y))
```

8

The CREATE command is used to create program, transaction, routing code and database definitions. It is also used to create descriptors for these resources, Descriptors are explained on the next page. The NAME parameter is used to specify the name of the resource or descriptor. You use the SET parameter to set the attribute values. Of course, default values may be used for any or all of the attributes. CREATE may be abbreviated to CRE.



## Descriptors

- A model for creating a resource or another descriptor
  - ◆ May be referenced in CREATE command with the LIKE parameter
  - ◆ Establishes defaults for attributes not SET in CREATE command
- IMS-defined descriptors
  - ◆ Provided with the IMS product
- User-defined descriptors
  - ◆ Defined by the user with CREATE commands
- Current system default descriptor
  - ◆ Each resource type will have one current default descriptor
    - IMS-defined or user-defined

9

A descriptor is a model or template (you will probably see both terms being used) that can be used by the user to define default attributes for resources being created. You can even use a descriptor as a model in creating another descriptor.

IMS provides four descriptors with the product – one for each resource type. Unless the user creates a descriptor with the DEFAULT(Y) attributes, these IMS descriptors are the “current system default descriptors.” The user may create a descriptor which can be specified in the CREATE command or can be set as the current default descriptor, replacing the IMS-defined descriptor. Whichever one is the default is used when the CREATE command does not SET all attributes.

## CREATE Command with LIKE Parameter

- LIKE parameter uses default attributes from another resource or descriptor
  - ◆ Examples:

```
CRE PGM NAME(PGM1) SET(RESIDENT(Y))
CRE PGM NAME(PGM2,PGM3) LIKE(RSC(PGM1))

CRE TRAN NAME(TRN1) SET(PGM(PGM1) MAXRGN(5)
    PARLIM(6) CLASS(4))
CRE TRAN NAME(TRN2,TRN3) LIKE(RSC(TRN1)
    SET(PGM(PGM23)))

CRE TRANDESC NAME(TRNCONV) SET(PGM(PGM1)
    CLASS(1) PARLIM(3) CONV(Y) SPASZ(2000))
CRE TRAN NAME(TRN5) LIKE(DESC(TRNCONV)
    SET(PGM(PGMC1) CLASS(5)))
```

10

The LIKE parameter may be used to specify a model for the resource or descriptor being created. LIKE refers either to another resource of the same type or to a descriptor for the type. When LIKE is used the new attributes for the new resource or descriptor have the same values as the referenced resource or descriptor unless they are specified with the SET parameter. The LIKE parameter allows you to use defaults other than those in the default descriptor.

The first two pairs of commands shown on this page show how the LIKE parameter may refer to another resource. The last pair of commands show how the LIKE parameter may refer to a descriptor.

## IMS-defined Descriptor

- IMS-defined descriptors
  - ◆ Provided with the IMS product – one for each resource type
    - Database descriptor           DFSDSDB1
    - Program descriptor           DFSDSPG1
    - Transaction descriptor       DFSDSTR1
    - Routing Code descriptor      DBFDSRT1
  - ◆ Define default attributes for resource types
    - When not specifically SET in CREATE command
    - When user-defined descriptor not set as default descriptor
- IMS-defined descriptors cannot be deleted or updated

*IMS-defined descriptor attribute values are shown in the appendix to this section.*

These are the four IMS-defined descriptors. They have similar names with two letters identifying the resource type to which the descriptor applies. These are DFSDSxx1, where xx is DB, PG, TR, or RT. These IMS descriptors cannot be updated or deleted except to make it the default if you have previously made a user-defined descriptor the default and want to go back to the IMS descriptor.

## User-defined Descriptor

- Descriptors created by users
  - ◆ Every descriptor has a value for every attribute
    - If not SET in CREATE command, attribute value defaults to value in current default descriptor

```
CREATE TRANDESC NAME(TRND001)
SET(PGM(PGM1) CLASS(1))
```

- ◆ A descriptor may become the default descriptor
  - DEFAULT(Y) parameter on CREATE or UPDATE command

```
CREATE TRANDESC NAME(TRND001)
SET(PGM(PGM1) CLASS(1)) DEFAULT(Y)
```

12

You can have as many user-defined descriptors as you like. User descriptors are created, updated, and deleted just like resources. They have the same attributes as resources. If, when creating a user descriptor, you do not SET the value of the attribute, then it will default to whatever that value is in the current default descriptor for that resource type.

When a user descriptor is created, you may make it the default descriptor by including the parameter DEFAULT(Y). Or, you may later make it the default using an UPDATE command.

The keyword for creating, updating, deleting, or querying descriptors are DBDESC, PGMDESC, TRANDESC, and RTCDESC. In the first example on this slide, the user has created a transaction descriptor named TRND001 with an attribute of CLASS(1). The program PGM1 will be invoked for transactions created with this descriptor unless this is overridden when they are created. The other attributes for TRND001 will be taken from whatever the current default transaction descriptor is, perhaps DFSDSTR1.

In the second example, the same descriptor is created, but it has been made the default transaction descriptor.



## UPDATE Command

- Updates existing resource definitions or descriptors
  - ◆ Resource may have been defined by system definition macro
- Examples:

```
UPDATE PGM NAME(PGM1) SET(RESIDENT(N))  
  
UPD     TRAN NAME(TRN1,TRN2) SET(PARLIM(4) MAXRGN(2))  
  
UPD     TRANDESC NAME(TRANCONV) SET(SPASZ(3000))  
  
UPD     TRANDESC NAME(TRND002) DEFAULT(Y)
```

- Cannot update attribute and status in the same command
  - ◆ The following command is not allowed:

```
UPD TRAN NAME(UJW001) START(SCHD) SET(MAXRGN(3))
```

13

The UPDATE command is used to update resources and descriptors. The parameters used with the CREATE command are also used with the UPDATE command. In the first example, the resident attribute for program PGM1 is set off. In the second example, the PARLIM and MAXRGN values for transactions TRN1 and TRN2 are changed. In the third example the SPASZ value of the TRANCONV transaction descriptor is changed. In the last example, the TRND002 transaction descriptor is made the default transaction descriptor.

An UPDATE command cannot update both an attribute and a status. The UPD TRAN command for transaction UJW001 attempts to change the status of to "start scheduling" and to the MAXRGN attribute to 3. This is invalid. The command will be rejected.

## DELETE Command

- Deletes existing resource definitions or descriptors
  - Cannot delete resources which are in use
- Examples:

```
DELETE PGM NAME(PGM1 )  
  
DELETE TRAN NAME( TRN1 , TRN2 )  
  
DEL      TRANDESC NAME( TRANCONV )
```

14

The DELETE command is used to delete existing resource definitions and descriptors.

To delete a database definition, it must not be in use. It cannot be allocated to the system. In addition, it cannot be referenced by the PSB for any program (PDIR) that is defined in the system.

To delete a program definition, it must not be in use. It cannot be currently scheduled. There cannot be any transactions defined that invoke this program. For FP(E) programs, there cannot be any routing codes that are associated with the program.

To delete a transaction definition, there must not be any instances on the local queue. You should stop queuing, release suspended transactions, and drain the queues. When the queues are empty, stop scheduling and exit all conversations. With shared queues, you may delete a transaction for which there are instances on the shared queues. Obviously, you should be careful about deleting such a transaction for all of the systems in the IMSplex.

To delete a routing code, it must not be active in any IFP region. You do not have to delete routing codes created by IMS for an FPE transaction. Deleting the transaction will also delete the routing code.

The first example deletes program PGM1. The second example deletes transactions TRN1 and TRN2. The third example deletes the TRANCONV transactions descriptor.

## Quiescing for UPDATE and DELETE Commands

- IMS will not delete a resource which is in use
  - ◆ If work is in progress for resource, command will fail
- IMS will not update certain attributes of a resource which is in use
  - ◆ For example, CONV parameter cannot be changed for a transaction
    - If work is in progress for the transaction, command will fail
  - ◆ On the other hand, PARLIM can be changed while a transaction is in use
    - If work is in progress for the transaction, command will succeed
- You may want to stop the resource before you attempt to update or delete it
  - *Command Reference* publications have complete instructions under "Usage notes" for each command

15

IMS will not allow you to delete a resource which is in use. If you attempt to do so, the command will fail.

When a resource is in use, some attributes can be updated and some cannot. For example, you can change the following with an UPDATE TRAN command without stopping the transaction: CLASS, CPRI, LCT, LPRI, MAXRGN, NPRI, PARLIM, PLCT, SEGNO, and SEGSZ. On the other hand you cannot change AOCMD, CMTMODE, CONV, DCLWA, DIRROUTE, EDITRTN, EDITUC, EMHBSZ, FP, INQ, MSGTYPE, MSNAME, PLCTTIME, PGM, RECOVER, REMOTE, RESP, SERIAL, SIDL, SIDR, SPASZ, SPATRUNC, TRANSTAT, or WFI while the transaction is in use. The Command References publications contain complete instructions for the use of these commands and when resources must be stopped under the heading "Usage notes" for each command.

When updating or deleting a resource, IMS quiesces the resource (set a quiesce flag) to prevent other commands from coming in and changing the resource at the same time.

## QUERY Command to Show Attributes

- Queries resource definitions or descriptor definitions
  - ◆ Including attributes defined with SET(...) parameter on CRE or UPD command
- Examples:

```
QUERY PGM NAME(PGM1) SHOW(BMPTYPE,RESIDENT)

QRY   TRAN NAME(TRN1,TRN2) SHOW(ALL)

QRY   TRANDESC NAME(TRANCONV) SHOW(ALL)
```

16

QUERY commands may be used to show attributes of resources and descriptors. The SHOW parameter may specify attributes to be shown or SHOW(ALL) may be used to show all of the attributes.

In the first example, only the BMPTYPE and RESIDENT attributes will be returned in the response.

In the second example, all attributes of transactions TRN1 and TRN2 will be returned.

In the third example, all attributes of the TRANCONV transaction descriptor will be returned.





# IMS Restarts

## Restoring Definitions on Restarts

- IMS logs all definitions
  - ◆ All resource and descriptor definitions are written during system checkpoints
  - ◆ CREATE, UPDATE, and DELETE commands are logged
- Definitions are restored during warm and emergency restarts
  - ◆ IMS reads all resource and descriptor definitions from the log
    - Built from system checkpoint and command log records
- Definitions may be restored during cold starts
  - ◆ Resource and descriptor definitions may be exported and imported
    - Export writes definitions to a Resource Definition Data Set (RDDS)
    - Import reads definitions from RDDS for cold starts
- Definitions are lost across cold starts if export and import are not used

18

IMS logs all resource and descriptor definitions during system checkpoints. If any changes are made by commands the commands are logged.

When a warm start is done, the definitions are restored from the shutdown checkpoint. When an emergency restart is done, the definitions are restored from the last system checkpoint and updated from the log records for any commands.

Resource Definition Data Sets are optional. When they are used, all definitions are written to one of them during the restart of the IMS system. If any changes are made to the definitions, all definitions are written to another RDDS during the next system checkpoint. During an IMS cold start, when DRD is enabled, you can import resource and descriptor definitions from an RDDS or from the active MODBLKS library. Or, you can choose not to import any definitions. In this case, you would have to define all your resources dynamically using the CREATE command.



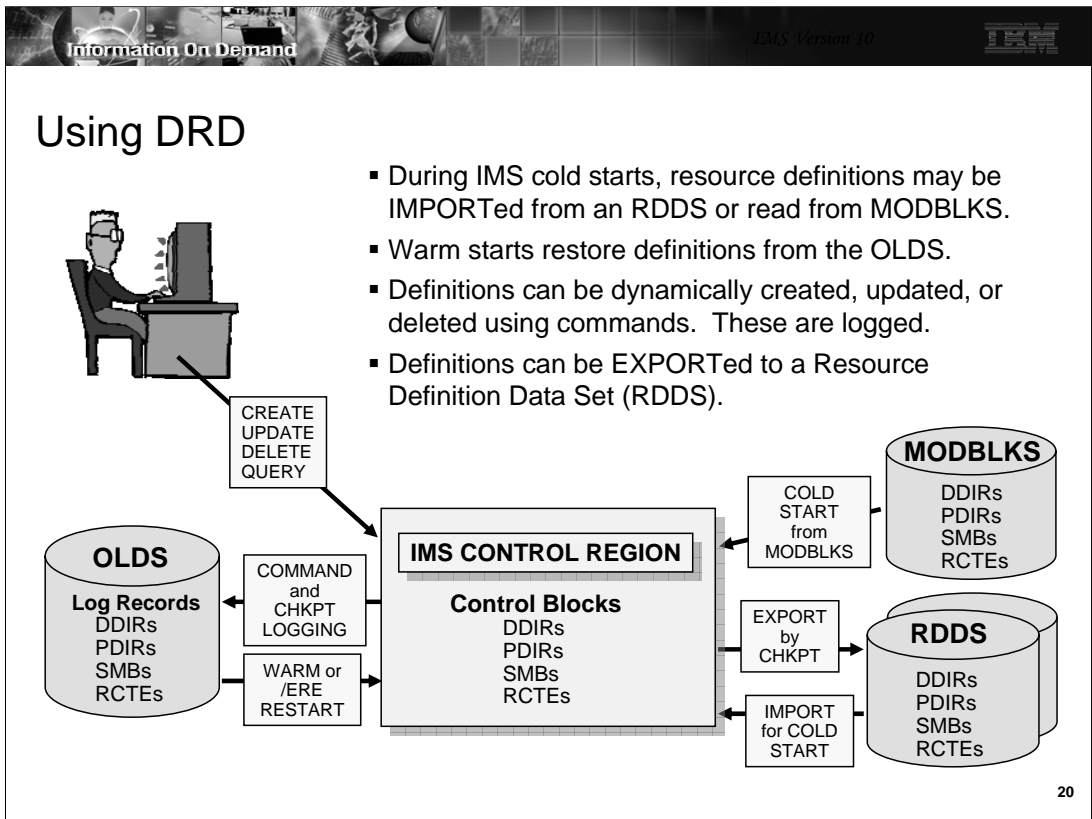
## Resource Definition Data Sets

- Set of BSAM data sets containing definitions of resources and descriptors
  - ◆ Used in a “round-robin” fashion
- Each IMS has its own set of RDDSs
  - ◆ Cannot be shared across IMSplex
- Target for “exporting” definitions during system checkpoint
  - ◆ All definitions are exported during initialization checkpoint
  - ◆ If any definitions have changed between later system checkpoints, all definitions are exported
- Source for “importing” definitions
  - ◆ Load resource and descriptor definitions

19

The Resource Definition Data Sets are a set of BSAM data sets used by IMS to export resource and descriptor definitions and import these definitions. A minimum of two is required, but more may be defined. They are used in a round-robin fashion with the most recent one never being the target of an EXPORT. IMS does not write over the last good set of definitions because a write error could destroy the last good copy of the definitions.

Each IMS has its own set of RDDSs. They are not shared by multiple IMSs. Each RDDS has a header record which identifies the IMS which initialized it. We will discuss the RDDSs in more detail later.

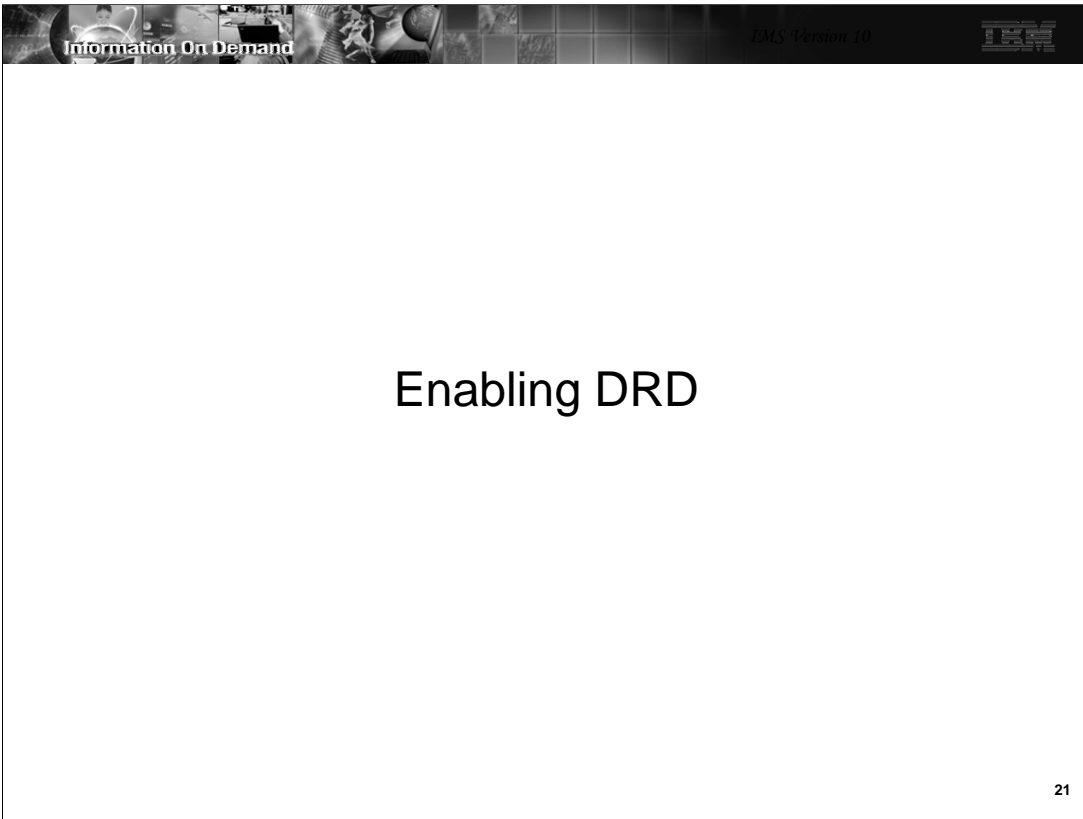


During an IMS cold start, IMS imports (loads) all of its resource definitions. In a non-DRD environment, these are always from the active MODBLKS library as determined by MODSTAT or OLCSTAT. In a DRD environment, IMS may “import” resource definitions from MODBLKS, or both resource and descriptor definitions from a new type of data set called the Resource Definition Data Set (RDDS). The process is called AUTOIMPORT and where the definitions are loaded from is determined by an execution time option in DFSDFxxx. MODBLKS, of course, does not contain any descriptors.

During an IMS warm start, definitions are rebuilt from the restart checkpoint on the OLDS. During an IMS emergency restart, definitions are rebuilt from the checkpoint log records plus additional log records created when the attributes or status of those definitions are changed. This is true for both DRD and non-DRD systems.

After restart is complete, in a DRD system, all definitions are “exported” to an RDDS (enabling them to be imported again at the next cold start). This process is called AUTOEXPORT and occurs at every system checkpoint, including the initialization and shut down checkpoints. Definitions are also logged during system checkpoint for restart purposes.

During execution, an operator using the OM interface may enter commands to create, update, or delete resource and descriptor definitions and status. CREATE and DELETE are available only in a DRD environment. UPDATE is available in either environment, although it is not as complete in terms of what can be updated. These updates are also logged for use during restart.



This section describes how to enable DRD.

## PROCLIB Members

- DFSPBxxx
  - ◆ **CSLG=xxx**
    - Suffix to DFSCGxxx – CSL proclib member
  - ◆ **DFSDF=xxx**
    - Suffix to DFSDFxxx – IMS Definition proclib member – new in V10
- DFSCGxxx
  - ◆ This proclib member can be replaced with CSL section of DFSDFxxx
  - ◆ CSL proclib member
    - DRD requires a CSL environment with SCI and OM; RM not required

◆ **MODBLKS=DYN**

- Enables DRD

MODBLKS= parameter may be specified in DFSCGxxx or DFSDFxxx member

22

The CSLG=xxx parameter in DFSPBxxx points to the CSL global proclib member (DFSCGxxx) and the DFSDF=xxx parameter points to the IMS System Definition proclib member which is new in IMS Version 10.

The entire contents of the CSL member can be moved to the CSL section of DFSDFxxx. If both DFSCGxxx and a CSL Section in DFSDFxxx exist, the DFSCGxxx member takes precedence. It is therefore necessary to remove the CSLG parameter from DFSDFxxx if you are coding a CSL section in DFSDFxxx.

DFSCGxxx contains the new parameter MODBLKS=DYN which enables DRD. MODBLKS defaults to MODBLKS=OLC. When MODBLKS=DYN is specified, online change cannot be used for the resources stored in MODBLKS (programs, transactions, routing codes, and databases). Online change for ACBLIB and FORMAT may be used with either MODBLKS=OLC or MODBLKS=OLC.

## PROCLIB Members

- DFSDFxxx
  - ◆ New “IMS Definition” PROCLIB member
    - **DFSDF=xxx** in DFSPBxxx
  - ◆ Divided into Sections
    - <**SECTION=COMMON\_SERVICE\_LAYER**>
      - May define **MODBLKS=** and CSL parameters as alternative to DFSCGxxx
        - If both exist, DFSCGxxx parameters override DFSDFxxx parameters
      - MODBLKS=DYN**
        - Enables DRD
    - <**SECTION=DYNAMIC\_RESOURCES**>
      - Defines DRD parameters

23

DFSDFxxx is a new proclib member in IMS Version 10. It is required for DRD but can also be used in place of DFSCGxxx and DFSSQxxx. The user may define four sections in this member.

DRD is enabled by specifying MODBLKS=DYN in the COMMON-SERVICE\_LAYER section.

If MODBLKS=DYN is coded in the CSL section, the Dynamic Resources section defines all the DRD parameters .

## DRD Section of DFSDFxxx

- Parameters in DFSDFxxx determine the automatic import and export functions of DRD

```
<SECTION=DYNAMIC_RESOURCES>
  RDDSDSN=( dsn1 , dsn2 , . . . )
  AUTOEXPORT=AUTO | RDDS | NO
  AUTOIMPORT=AUTO | RDDS | MODBLKS | NO
  IMPORTERR=ABORT | CONTINUE
  RDDSERR=ABORT | NOIMPORT
```




24

The DFSDFxxx member has multiple sections. The DYNAMIC\_RESOURCES section is used to define DRD export and import parameters.

- RDDSDSN defines the RDDS data set names. These must be defined to import from an RDDS (obviously) or to export. If defined, there must be at least two but may be more.
- AUTOEXPORT defines the option for exporting definitions at system checkpoint.
- AUTOIMPORT defines the option for importing definitions at IMS cold start
- IMPORTERR determines what action to take if IMPORT fails for other than an allocation or read error
- RDDSERR determines what action to take if there is an allocation or read error

The meanings of these parameters will be discussed in the next several slides.



## Resource Definition Data Sets (RDDS)

**RDDSDSN=( dsn1 , dsn2 , . . . )**

- Defines the RDDSs to IMS
  - ◆ A set of BSAM data sets that contain MODBLKS resource and descriptor definitions for one IMS
    - None or a minimum of 2 RDDSs must be defined
      - IMS abends during initialization with U0071 if only 1 defined
    - Each IMS has its own set of RDDSs
      - Header record contains IMSID and last update timestamp
    - Dynamically allocated by IMS when needed
      - Must be predefined and cataloged
        - Variable blocked with block size up to 32760
        - **DCB=( RECFM=VB , BLKSIZE=32760 , LRECL=32756 )**
    - Size requirements are about 1.5 times the size of MODBLKS data set

RDDSDSN defines the RDDS data set names to IMS. For example, they might be defined as:

```
RDDSDSN=( IMS10.RDDS1 , IMS10.RDDS2 , IMS10.RDDS3 , IMS10.RDDS4 )
```

Technically they do not have to be defined to a DRD system, however, if they are not, then IMS cannot import or export definitions. IMS can import from MODBLKS but cannot export, meaning everything that changes cannot be restored during a cold start. If you do define RDDSDSN, then you must define at least two. IBM recommends at least three so that you have a spare in case of an RDDS error. If you define only one, IMS initialization will abend with a U0071.

Each RDDS is pre-allocated as a BSAM data set. When IMS initialization opens these data sets for the first time, it will create a header record with its own IMSID (the RDDSs are not shared by multiple IMSs). As EXPORT writes to these data sets, the export time (last update time) is also written in the header record.

IMS dynamically allocates these data sets using the data set names in this parameter. Therefore they must also be cataloged. The block size of the RDDS may be up to 32760. It is a variable blocked sequential data set. Sample DCB parameters are shown for the maximum block size. The size of each RDDS should be about 1.5 times the size of one of your MODBLKS data sets. Larger would be wise if you intend to add many new definitions.



## Exporting Resource and Descriptor Definitions

**AUTOEXPORT=**AUTO | RDDS | NO

### **AUTO** (default)

- Export all resource and descriptor definitions to oldest RDDS
- When export completes, a message is issued identifying the RDDS used
  - DFS3371I SUCCESSFUL AUTOMATIC EXPORT TO RDDSDSN=dsn
- No export if no RDDSs are defined
  - DFS3374I AUTOMATIC EXPORT DISABLED

### **RDDS**

- Same meaning as AUTO

### **NO**

- Definitions not exported to RDDS

26

AUTOEXPORT defines what IMS is to do at all IMS checkpoints.

AUTO and RDDS have the same meaning. Definitions will be exported to the oldest RDDS (the one with the oldest time stamp in the header record).

During system checkpoint, definitions are exported only if at least one of them has changed (created, deleted, updated) since the last export. The exceptions to this are that export always occurs during the initialization checkpoint. When export does run, ALL definitions are exported, even those that didn't change.

If export fails trying to write to one, it will try another but will NOT try writing to the most recent. This would be overwriting the latest definitions and failure there could pose significant problems. If only one is available (that is, it can't write to another RDDS), then IMS will issue a DFS3372E AUTOMATIC EXPORT FAILED, RC=8 message.

If you specify AUTOEXPORT=AUTO or RDDS, and no RDDSs have been defined, IMS initialization will abend with a U0071.

## Export Errors

- If an RRDS error occurs during export processing
  - ◆ IMS issues a DFS3368E message with appropriate reason code
    - DFS3368E ERROR PROCESSING RDDS, RC=rc, RDDSDSN=dsn
  - ◆ Continues with next available RDDS
  
- If RDDSs are defined but none are available
  - ◆ IMS issues a DFS3370E message
    - DFS3370E NO SYSTEM RESOURCE DEFINITION DATA SETS ARE AVAILABLE FOR AUTOMATIC EXPORT
      - Message highlighted on console – deleted when successful export
    - Redefine and allocate bad RDDSs
  - ◆ Export processing suspended until next checkpoint
  - ◆ Try again at next checkpoint

27

If an error occurs during export processing, IMS will attempt exporting to the next available RDDS. Message DFS3368E identifying the bad RDDS and a reason code is issued.

If there are no RDDSs available for export (remember, the latest one will not be used), IMS will abort export processing and issue message DFS3370E.

In either case, the user should delete and reallocate the bad RDDS

## Importing Definitions for Cold Start

**AUTOIMPORT=AUTO | RDDS | MODBLKS | NO**

### **AUTO (default)**

- ◆ Import resource and descriptor definitions from most recent RDDS if
  - Two or more system RDDSs are defined
  - And - all the defined RDDSs can be allocated and read
  - And - at least one of the RDDSs contains valid definitions
- ◆ Import resource definitions from active MODBLKS data set in JCL if
  - No system RDDSs are defined in DFSDFxxx
  - Or - all of the defined system RDDSs are empty
  - MODBLKS suffix is determined by OLCSTAT or MODSTAT
- ◆ Automatic import is not performed if
  - Two or more system RDDSs are defined, but IMS cannot allocate and read all of them
    - IMS action depends on RDDSERR parameter
  - Or - no RDDSs are defined and no MODBLKS data set is defined

28

AUTOIMPORT defines what IMS is to do during cold start. AUTOIMPORT is invoked only during IMS cold start. Warm and emergency restart load control blocks from the logs. Options are to import (load) definitions from MODBLKS, from the most recent RDDS, or to not load them at all. If they are loaded from MODBLKS, then MODSTAT or OLCSTAT is used to determine which library (A or B) to use. Care must be taken to ensure the correct version of MODBLKS is being used.

AUTO is the default for AUTOIMPORT and is recommended. AUTO lets IMS decide from where to import the definitions. When AUTO is used IMS will import from the most recent RDDS if it can be determined and it contains definitions. If all of the RDDSs cannot be read, the most recent RDDS cannot be determined. If there are no RDDSs or they are all empty, definitions will be read from MODBLKS. The MODBLKS suffix is determined the same way it is without DRD. If there are RDDS data sets, but one or more cannot be read definitions are not loaded from either an RDDS or MODBLKS. What IMS does depends on the IMPORTERR parameter. If there are no RDDS or MODBLKS data sets defined, definitions are not loaded.

If any transaction definitions include Transaction Edit Routines, they are also loaded at this time.

## Importing Definitions for Cold Start

**AUTOIMPORT=**AUTO | RDDS | MODBLKS | NO

### **RDDS**

- ◆ Import resource and descriptor definitions from most recent RDDS
  - Same conditions as AUTO except will not import from MODBLKS if all defined RDDSs are empty

### **MODBLKS**

- ◆ Import resource definitions from MODBLKS data set in JCL
  - MODBLKS suffix is determined by MODSTAT or OLCSTAT

### **NO**

- ◆ Do not import definitions from either RDDS or MODBLKS
  - All resources have to be created dynamically

29

If AUTOIMPORT equals RDDS or MODBLKS, IMS will import from whichever is specified but the same conditions apply as for AUTO.

If AUTOIMPORT=NO, then IMS will come up without any MODBLKS resources defined. If this happens, then all definitions would have to be dynamically created. This might be OK for a test system, but probably not for production.

## Import Errors

### **RDDSERR=ABORT | NOIMPORT**

- ◆ Specifies action to take if IMS cannot allocate and read all defined RDDSs
  - I/O error, allocation error, etc.

#### **ABORT**

- IMS cold start terminates with U3368 abend
- Message indicates reason

#### **NOIMPORT**

- Cold start completes but no resources defined

30

RDDSERR determines the action IMS will take in the event of an error trying to allocate or read the RDDSs – for example, if there is a read error. It does NOT apply if the RDDS is successfully read but there are errors in the definitions.

One option is to ABORT with a U3368 abend. Messages will indicate the cause of the failure. The other option is to continue without importing. This is similar to the AUTOIMPORT=NO option.



## Import Errors

### **IMPORTERR=ABORT | CONTINUE**

- ◆ Specifies action to take due to invalid resource or descriptor definition or failure to load Transaction Input Edit Routine
  - This is a “should not happen” error

#### **ABORT**

- IMS cold start terminates with U3397 abend
- Message identifies resource(s) with bad definitions

#### **CONTINUE**

- Resource in error marked bad and given NOTINIT status
- Must manually correct error after restart completes (CREATE or UPDATE)

31

IMPORTERR says what to do if some definitions are imported but later determined to be invalid (a “should not happen” condition) or if a transaction definition specifies a Transaction Input Edit routine and that edit routine cannot be loaded. Again, there are two options:

-ABORT causes IMS to abend with a U3397. DFS3398E identifies the “bad” definitions.

-CONTINUE lets IMS continue without the bad definitions. They acquire a status of NOTINIT and must be corrected later with UPDATE or DELETE/CREATE.

## A Procedure to Enable DRD

- Update IMS proclib members
  - ◆ DFSCGxxx (set MODBLKS=DYN)
  - ◆ DFSDFxxx (define RDDSDSNs, set other DRD parameters – AUTOIMPORT=AUTO)
- Define and catalog RDDSDSNs
  - ◆ DSNs defined in DFSDFxxx
- Cold start IMS
  - ◆ Load definitions from active MODBLKS
    - RDDSDSNs will be empty
  - ◆ Exports definitions to RDDSDSNs at first system checkpoint
- Warm start and emergency restart
  - ◆ Rebuild definitions from IMS checkpoint records
- Next cold start
  - ◆ Import definitions from most recent RDDSDSN (AUTOIMPORT=AUTO)
    - RDDSDSNs no longer empty

32

DRD is “enabled” simply by coding MODBLKS=DYN (meaning MODBLKS changes are dynamic – not by online change) in either DFSCGxxx or the CSL Section of DFSDFxxx. Technically it is not necessary to code any other parameters, however, if RDDSDSNs are not defined, export and importing from an RDDSDSN cannot be done. Import could only be done from MODBLKS and OLC would be disabled. Creates, updates, and deletes could be done but they couldn’t be exported.

Assuming this is not what you want, you must create a DFSDFxxx proclib member with a DRD section containing the desired RDDSDSN, AUTOIMPORT, AUTOEXPORT, and DCLWA parameters and the parameters for handling errors. Since all of these parameter except RDDSDSN have reasonable defaults, it may not be necessary to code anything except RDDSDSN.

You can then cold start IMS. Assuming AUTOIMPORT=AUTO, IMS will detect that the RDDSDSNs are empty and import the definitions from the active MODBLKS. At the completion of cold start, AUTOEXPORT=AUTO or RDDSDSN will cause IMS to export these MODBLKS definitions to an RDDSDSN.

After cold starting IMS the first time successfully, and before the next cold start, you can remove the MODBLKS DD statements from the control region JCL. This is not, however, necessary and you may want to leave them in until you are certain you will never want to import from them again.

At the next IMS cold start, the latest version of the definition in RDDSDSN will be imported. At the next warm or emergency restart, the definitions will be rebuilt from the logs.



## Processing Flow with DRD Enabled

- During IMS initialization
  - ◆ All RDDSs are dynamically allocated, opened, and read
  - ◆ Empty RDDSs
    - Initialized with header record containing IMSID
  - ◆ Non-empty RDDSs
    - Header records with timestamps are read and put into an array
- During IMS cold start
  - ◆ Definitions are imported from
    - MODBLKS
    - Or - RDDS with most recent export timestamp
- During IMS warm restart
  - ◆ Definitions are rebuilt from checkpoint log records
- During IMS emergency restart
  - ◆ Definitions are rebuilt from restart checkpoint plus x'22' log records

33

This slide shows the general flow when DRD is enabled.

During IMS initialization:

- All RDDSs defined in DFSDFxxx are dynamically allocated and read
- If they are empty, IMS initializes them with a header record containing the IMSID
- If they are not empty, IMS builds an array in storage of the available (i.e., successfully allocated and read) RDDSs. These RDDSs will be used in round-robin fashion by export

During IMS cold start:

- Definitions are imported from the most recent RDDS or from MODBLKS

During warm start:

- Definitions are rebuilt from the checkpoint log records

During emergency restart:

- Definitions are rebuilt from restart checkpoint log records plus and updates made since that checkpoint (x'22' log records)

## Processing Flow with DRD Enabled

- After all restart types
  - ◆ All definitions are exported to RDDS (unless AUTOEXPORT=NO)
- During normal operations
  - ◆ Creates, updates, and deletes are logged (x'22')
  - ◆ User may scratch and reallocate "bad" RDDSs
- During system checkpoint (including shutdown checkpoint)
  - ◆ Resource and descriptor definitions are logged
  - ◆ If any definitions have changed since last checkpoint
    - All definitions are exported to the RDDS with the oldest (or no) export time
  - ◆ RDDS is closed and deallocated

34

After any type of restart completes. AUTOEXPORT will export definitions to the oldest RDDS. These (or later exports) then become available at the next IMS cold start.

During normal operations, all creates, updates, and deletes are logged in x'22' log records. If an RDDS cannot be read, an error message is issued. You may scratch and reallocate this RDDS. This will add it back to the RDDSs that are used by IMS. You cannot add new RDDS data sets while IMS is executing. Only those defined in the DFSDfxxx member at start up will be used.

At initialization time, IMS reads the RDDSDSN parameter in the DFSDfxxx member. It opens the specified data sets and reads their headers.

At system checkpoint times, if any definitions have change since the last checkpoint, all definitions are exported to the next RDDS. The RDDS is closed and deallocated after the definitions are written to it.

# RDDS Extraction Utility

## RDDS Extraction Utility

- RDDS Extraction Utility
  - ◆ Creates Stage 1 macros or CREATE commands from definitions in RDDS
    - May be used to copy definitions to another system
    - May be used to fall back to non-DRD use
    - May be used to create documentation of system definitions
  - ◆ Stage 1 macros
    - DATABASE, APPLCTN, TRANSACT, and RTCODE macros are written
    - Does not write descriptor information
    - Does not write other stage 1 macros (IMSCTRL, IMSCTF, etc.)
  - ◆ CREATE commands
    - CREATE commands for DB, DBDESC, PGM, PGMDESC, TRAN, TRANDESC, RTC, and RTCDESC are written

36

The RDDS Extraction Utility (DFSURDD0) is a batch utility which can be used to convert the resource definitions in an RDDS into either stage 1 macro statements or type-2 CREATE commands.

This provides a convenient way to take the definitions for one system and copy them to another system. It also may be used fall back from the use of DRD to the use of MODBLKS. Finally, the creation of stage 1 macros may be used as documentation of the current definitions.

If stage 1 macros are produced, descriptor information in the RDDS is not created. Of course, it only produces stage 1 macros for MODBLKS resources (DATABASE, APPLCTN, TRANSACT, and RTCODE macros).

If CREATE commands are produced, commands for creating descriptors are included.

## RDDS Extraction Utility

- Sample JCL

```
//MYJOB JOB CLASS=J,MSGCLASS=A,MSGLEVEL=(1,1)
//JOB LIB DD DSN=IMS10.SDFSRESL,DISP=SHR
//S1 EXEC PGM=DFSURDD0,MEMLIMIT=4G
//DFSRDDS DD DSN=IMS10.RDDS01,DISP=SHR
//SYSOUT DD DSN=MY.RDDS.OUTPUT,DISP=(,CATLG,DELETE),
// UNIT=SYSDA,VOL=SER=ABC001,
// SPACE=(CYL,(1,1),RLSE),
// DCB=(LRECL=80,RECFM=FB,BLKSIZE=800)
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
OUTPUT=MAC
/*
//
```

- ◆ SYSIN control statement:

- **OUTPUT=MAC** creates stage 1 macros
- **OUTPUT=CMD** creates CREATE commands
  - May be used as input to the Batch SPOC utility

37

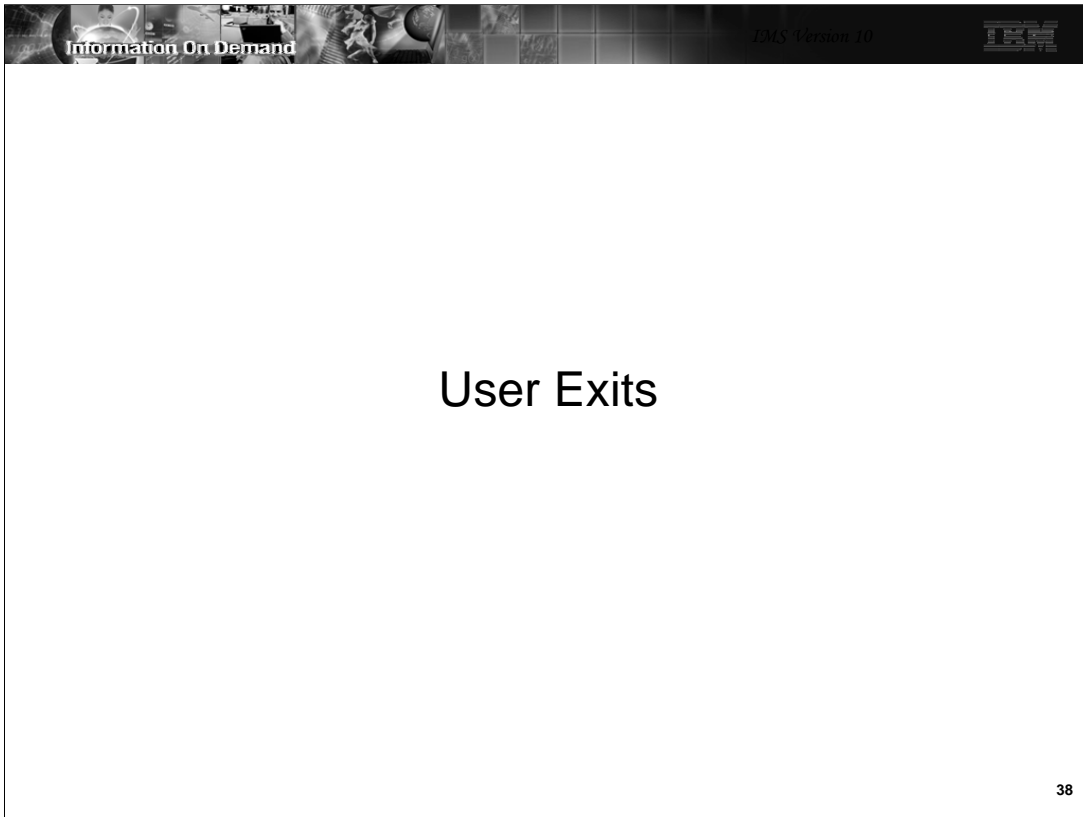
This shows sample JCL for the execution of the RDDS Extraction Utility (DFSURDD0). The utility obtains 64-bit storage. The MEMLIMIT parameter on the EXEC statement should be used to ensure that adequate storage above the bar is available for the utility. MEMLIMIT should be set to 4G or higher.

The DFSRDDS is used to specify the RDDS input data set. You are responsible for choosing the correct RDDS. Typically, this will be the one most recently written.

The output is written to the SYSOUT data set.

The SYSIN data set is used for the control statement. Either OUTPUT=MAC or OUTPUT=CMD must be specified. There is not a default.

The output with OUTPUT=CMD may be used as input to the Batch SPOC utility. This utility is new in IMS 10. It is discussed in the Operations Enhancements section.



There are some user exits affected by DRD.



## Transaction Input Edit Routines

- Defined on CREATE TRAN or UPD TRAN
  - ◆ SET(EDITRTN(name))
- If found in nucleus
  - ◆ If other transactions reference it
    - Use nucleus copy
  - ◆ If no other transactions reference it
    - Dynamically load it from STEPLIB concatenation
    - Do not use nucleus copy
  - ◆ If dynamic load fails
    - Use nucleus copy
- If not found in nucleus
  - ◆ Dynamically load it from STEPLIB concatenation
    - If load fails, CREATE or UPDATE command fails

39

Transaction edit routines are defined during create or update by the SET(EDITRTN(rtn-name)). Since the name of these edit routines is not known at IMS initialization, the rules for loading it following a create or update are:

- if the edit routing is found in the nucleus and other transactions use it, use the nucleus copy
- if no other transactions reference it, load it from the STEPLIB concatenation – do not use the nucleus copy
- if it is not found in STEPLIB, then use the nucleus copy

If the edit routing is not found in the nucleus

- load it from STEPLIB
- if it fails, then the create or update command fails

## Destination Creation Exit (DFSINSX0)

- Originally called “Output Creation Exit”
  - ◆ Was ETO-only exit; then ETO and Shared Queues
  - ◆ Now called in any environment when destination is unknown
- IMS V10
  - ◆ Exit enhanced and name changed
  - ◆ When message with unknown destination arrives, exit is invoked
    - Can create transaction destination in any environment (DRD or non-DRD)
    - Can create program to process transaction (DRD only)
    - Can create LTERM destination (ETO only)
    - Can reject message
  - ◆ May also be invoked with QUEUE command
  - ◆ To enable exit, include in STEPLIB concatenation

40

DFSINSX0 was originally called the Output Creation Exit and was implemented for ETO to dynamically create an LTERM destination. When shared queues came along, it was enhanced to allow the exit to also create a transaction destination. However, IMS could use this only for purposes of queuing a message to the shared queue. It could not schedule that transaction locally.

With IMS 10 the name of the exit has change to Destination Creation Exit. And it has been enhanced in terms of what it can create and for what purpose (i.e., for queuing and/or scheduling). It can create both transactions and the programs to schedule those transactions. It can also create LTERMs but only in an ETO environment.

The exit is invoked when a message with an unknown destination arrives in IMS, and also when the new QUEUE command references a TRAN or LTERM which is not defined. The exit is only invoked for messages. It will not be invoked when you attempt to execute a BMP which is not defined.

The exit can create both transactions and programs. It can create them on the IMS where the exit is invoked, on one other IMS, or on all IMSs. SCI is used to send the create requests to other IMS systems.

To enable DFSINXS0, all you have to do (besides coding it) is to include it in STEPLIB. Details for writing this exit can be found in the Exit Routine Reference.



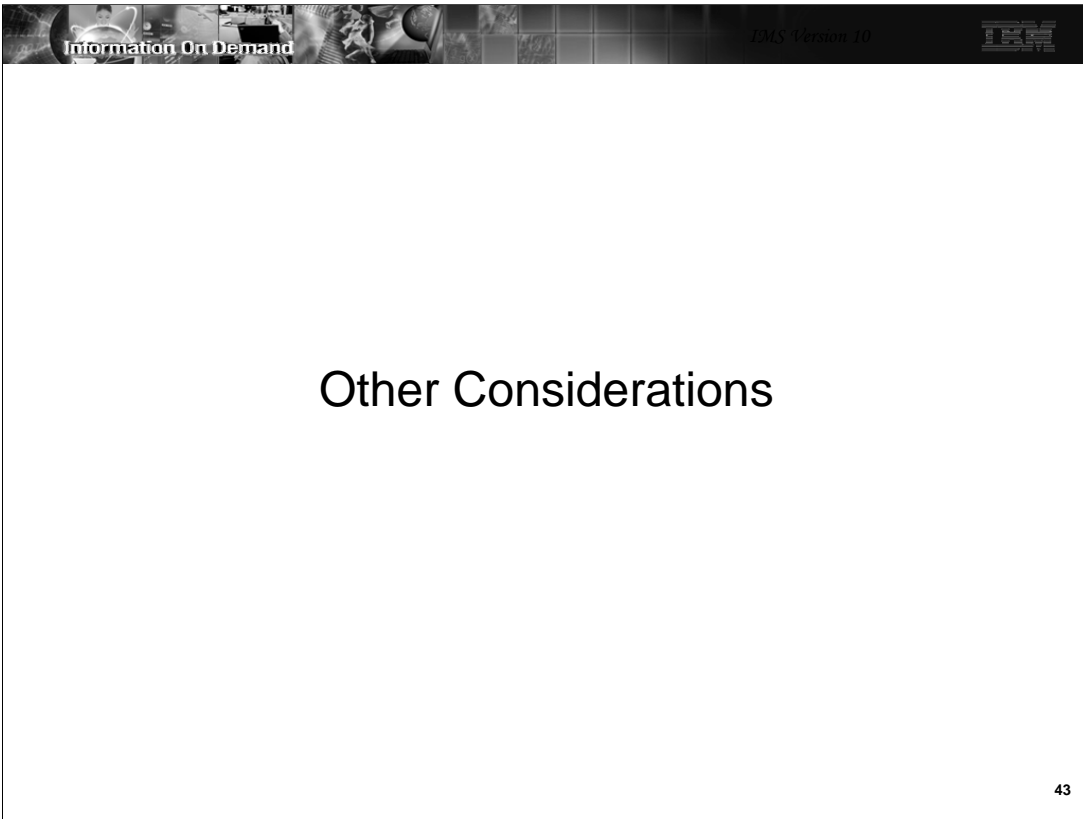
# Security

## Security Considerations

- All DRD commands are type-2 and can only be entered through the Operations Manager interface
- Command security for type-2 commands can only be performed by the Operations Manager
  - ◆ In OM Initialization proclib member (CSLOIxxx)
    - `CMDSEC=N|R|E|A`
      - N do not perform command authorization in OM (default)
      - R call RACF for authorization
      - E call user security exit for authorization
      - A call both RACF (first) and the user security exit

42

All DRD commands are type-2 commands. That means they are only submitted through the OM interface. Therefore, only OM can perform authorization processing for these commands. In the OM initialization proclib member (DFSOIxxx), the CMDSEC parameter determines what type of security checking should be done for commands submitted through OM. There is not change in the regard for the DRD commands. This slide is just a quick review of what is already available.



There are some other considerations when enabling DRD.

## IMS System Generation Considerations

- DRD can eliminate the following macros from the system definition
  - ◆ DATABASE
  - ◆ APPLCTN
  - ◆ TRANSACT
  - ◆ RTCODE
    - These system definition macros may be used with DRD, but are not required
  
- IMSCTRL
  - ◆ MSVID= only applicable when using MODBLKS (not DRD)
  - ◆ Can still use /MSVERIFY command to verify MSC resource definitions

44

There are a few sysgen considerations when implementing DRD, although nothing is required.

-You can eliminate the DATABASE, APPLCTN, TRANSACT, and RTCODE macros. If you do not eliminate them, their definitions will reside in the MODBLKS data set, but IMS execution will not read them from MODBLKS

-The MSC Verification ID (MSVID) is not applicable when DRD is enabled. You can, however, continue to use /MSV to verify your MSC definitions.

## Elimination of MODBLKS

- MODBLKS
  - ◆ Can eliminate `//MODBLKSx DD` statements and libraries
    - Resources can be dynamically created
    - Resources can be imported from Resource Definition Data Set (RDDS)

After successfully importing from MODBLKS the first time, and exporting to an RDDS, the MODBLKS data sets can be eliminated from the control region JCL.

## Operational Considerations

- Each resource or descriptor is created, updated, or deleted individually
  - ◆ Some may be successful and some may fail
  - ◆ If successful
    - Response will have CC=0 for that resource
  - ◆ If not successful
    - Response will have CC>0 for that resource and indicate reason for failure

```
CREATE PGM NAME ( PGM1 , PGM2 )
```

| PgmName | MbrName | CC | CCText                  |
|---------|---------|----|-------------------------|
| PGM1    | IMS1    | 11 | RESOURCE ALREADY EXISTS |
| PGM2    | IMS1    | 0  |                         |

46

Each resource or descriptor is created, updated, or deleted individually. If some fail, a non-zero return code and condition code indicate the reason for the failure of each resource. In this example, CREATE for PGM1 failed because PGM1 already existed. However, CREATE for PGM2 as successful.

## Operational Considerations

- Type-1 commands affected by DRD
  - ◆ /DIS MODIFY
    - Indicates whether DRD is enabled
  - ◆ /MOD PREPARE
    - Cannot specify MODBLKS if DRD enabled
    - “ALL” is OK – MODBLKS omitted by IMS if DRD enabled

47

/DIS MODIFY indicates whether DRD is enabled or not.

/MOD PREPARE MODBLKS will fail if DRD is enabled. /MOD PREPARE ALL can be issued – the MODBLKS library be ignored.

## Operational Considerations

- Type-2 commands affected when DRD enabled
  - ◆ INIT OLC
    - TYPE(MODBLKS) returns CC=1 (not applicable)
      - TYPE(ALL) OK
  - ◆ QRY OLC
    - Does not display MODBLKS information
  - ◆ QRY MEMBER
    - Indicates whether DRD is enabled or not
      - LclAttr = DYNMODBLKS
  - ◆ QRY DB and QRY TRAN
    - Displays additional attributes
  - ◆ UPD DB and UPD TRAN
    - Can SET additional attributes

48

Some type-2 commands are affected when DRD is enabled.

INIT OLC TYPE(MODBLKS) will return a completion code of 1 which indicates that it is not applicable. If the command is processed by multiple IMSs and only some have DRD enabled, the command will be processed by those IMSs without DRD and ignored by those with DRD.

QRY OLC does not display MODBLKS information if processed by a DRD system.

QRY member indicates whether DRD is enabled for the IMS processing the command. It will have a Local Attribute of DYN or MODBLKS.

QRY DB and QRY TRAN can SHOW additional attributes.

UPD DB and UPD TRAN can SET additional attributes.



## Operational Considerations

- Commands that don't work if DRD not enabled
  - ◆ CREATE DB|PGM|TRAN|RTC
  - ◆ DELETE DB|PGM|TRAN|RTC
  - ◆ UPDATE DB with SET(...)
    - Except SET(ACCTYPE(...)) and SET(LOCK(...))
  - ◆ UPDATE PGM with SET(...)
    - Except SET(LOCK(...))
  - ◆ UPDATE TRAN with SET(...)
    - Except those attributes allowed to be SET in V9 and TRANSTAT
  - ◆ All commands that reference descriptors

49

These commands are disabled if DRD is enabled.

For UPD DB, you can SET ACCTYPE and SET LOCK but you cannot set RESIDENT.

For UPD PGM, you can only SET LOCK – no other attributes

For UPD TRAN, you can only SET those attributes supported by V9

And of course, and commands that reference descriptors and those that import or export definitions.

## Shared Queues Considerations

- Transactions can be updated and deleted even when message exists on the shared queue
  - ◆ Must be careful to create or update transaction to be consistent with messages on shared queue
    - For example
      - SPASZ, RESP, FP, EMHBSZ, MSGTYPE, SEGNO, SEGSZ, SERIAL, EDITRTN
  
- Programs can be created and updated when there are associated transactions on the shared queue
  - ◆ Must be careful to define program to be consistent with messages on shared queue for that program
    - FP, BMPTYPE, CONV, SERIAL

50

There are some considerations when using shared queues. Most have already been discussed.

-Transactions can be created, updated, or deleted even if there are messages queued for that transaction on the shared queue. You must be careful when updating or creating a transaction in a SQ environment to set the attributes consistently across all IMSs in the SQ group.

-Programs can be created, updated, or deleted even if there are messages queued for that program on the shared queue. You must be careful when updating or creating a program in a SQ environment to set the attributes consistently across all IMSs in the SQ group.

The slide indicates some of the attributes that can cause problems if not defined consistently.

## IMSplex Considerations

- CREATE, DELETE, and UPDATE are not mirrored across IMSplex
  - ◆ Can be successful on some IMSs and fail on others
    - Response indicates success or failure for each IMS
  - ◆ User responsibility to verify success everywhere
  
- When RM and Resource Structure available
  - ◆ CREATE TRAN creates entry in RM structure to enforce resource name consistency for Transactions, LTERMs, and MSNAMEs
    - DELETE TRAN does NOT delete transaction entry from structure
      - Can only delete by deleting structure
      - QUERY TRAN after delete still displays the transaction, but with all blank attributes since the transaction is not defined locally
  - ◆ DFSINSX0 creates entry in RM structure when transaction is created

51

Creates, deletes, and updates are not automatically mirrored across an IMSplex. When routing the command to all IMSs, it may be successful on some and fail on others. It is a user responsibility to verify success everywhere, or to take corrective action.

CREATE TRAN and DFSINSX0 creates an entry in the Resource Structure but DELETE TRAN does not delete it.

## XRF Considerations

- “Created” and “Updated” resources picked up by XRF alternate in the log stream and created or updated on the alternate
- XRF active and alternate use different RDDSs

The XRF alternate reads the active log and processes the x'22' log records.

When using Resource Definition Data Sets the XRF alternate and active systems do not share these data sets. You must define different sets for the two systems.



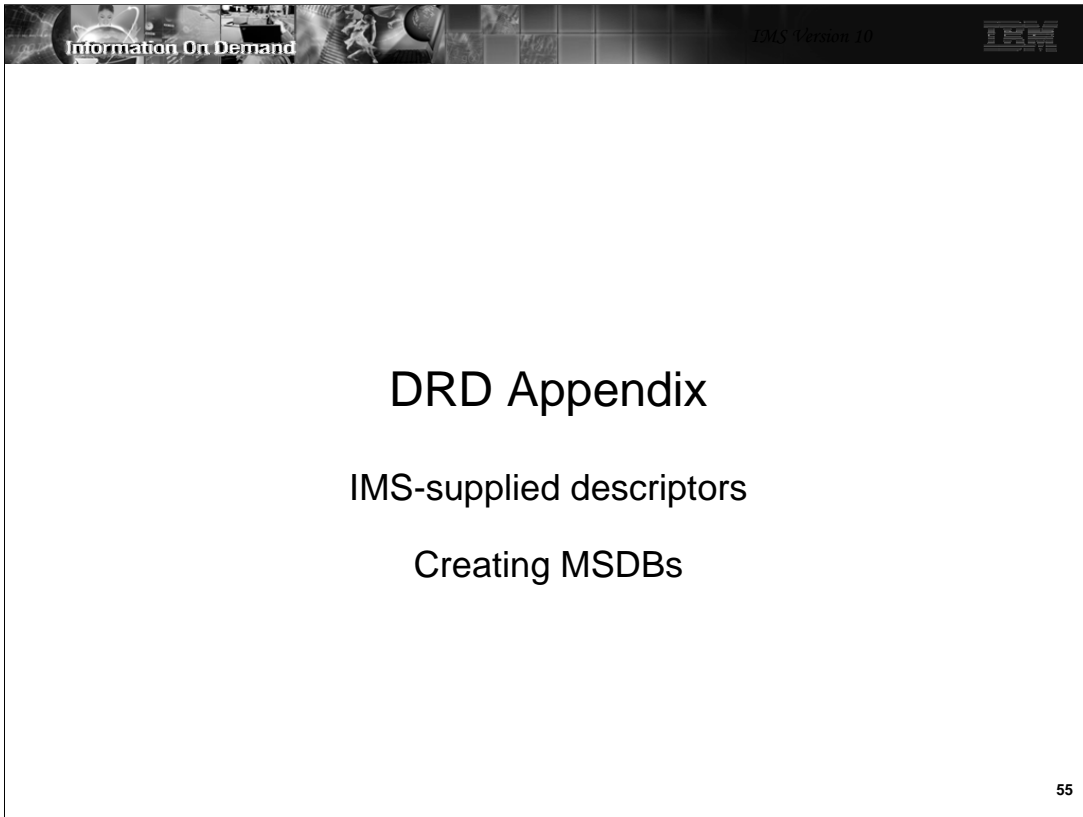
## Summary

- DRD is an alternative to online change for MODBLKS
- DRD is supported in all IMS environments
  - ◆ DB/DC, DB/CTL, DC/CTL
  - ◆ Data sharing, shared queues
- Definitions can be dynamically created, deleted, and updated
  - ◆ CREATE, DELETE, UPDATE commands
  - ◆ DB, DBDESC, PGM, PGMDESC, TRAN, TRANDESC, RTC, RTCDESC
- Resources and descriptors can be modeled after existing resource or descriptor
  - ◆ CRE .... LIKE(DESC(desc-name)) < or > LIKE(RSC(rsc-name))
  - ◆ Determines defaults for attributes not explicitly SET in CREATE command



## Summary

- Migration to DRD requires a cold start
- Definitions persist across warm and emergency restarts
- Export and import may be used to carry definitions across a cold start
- Extract may be used to copy definitions to another system



This section shows the descriptors that are supplied with IMS. They are the default descriptors unless you change the defaults. It also has a procedure for creating MSDBs with DRD.

## DFSDSDB1 – IMS-defined Database Descriptor

| DATABASE macro keyword with default | CREATE DB attribute name with default        |
|-------------------------------------|--|
| DBD=(name)                          | NAME(name1,name2,...)                        |
| (not) RESIDENT                      | RESIDENT( <u>N</u>  Y)                       |
| ACCESS=EX                           | ACCTYPE( <u>UPD</u> ,EXCL,READ,BRWS)         |
|                                     | DEFAULT( <u>N</u>  Y) – for descriptors only |

- DRD and system definition have different defaults for access type
  - ◆ DRD ACCTYPE default is UPD
  - ◆ System definition ACCESS default is EX
- To keep exclusive as the default
  - ◆ Create a user descriptor with ACCTYPE=EXCL and DEFAULT(Y)

56

This table shows the default attributes in DFSDSDB1. You cannot update these attributes. To change them, you would need to define a new DB descriptor with the different attributes.

The left column of the table shows the attribute and its default in the DATABASE macro. The right column shows it as it appears in the CREATE command. There are some differences in the keyword (spelling), the parameters, and the defaults.

-There are no defaults for NAME. It must be specified in the CREATE command.

-The Non-resident attribute is implied by the absence of the RESIDENT keyword in the DATABASE macro. It is specifically coded in the CREATE command although it defaults to the same value, NO.

-ACCTYPE replaces ACCESS. Note that the default is UPD, which is different from the default for the DATABASE ACCESS parameter, which is EX. To return to EXclusive, you would have to define a DB descriptor with ACCTYPE=EXC and then use that descriptor either with LIKE or set it as the default.

-DEFAULT(N|Y) applies only to descriptors. If DEFAULT(Y) is specified, this descriptor will replace DFSDSDB1 (or another DB descriptor) as the current system default descriptor for databases.



**DFSDSPG1 – IMS Program Descriptor**

| APPLCTN macro keyword w/ default                  | CREATE PGM attribute name with default  |
|---|---|
| PSB=name < or ><br>GPSB=name                      | NAME(name1,name2,...) < and ><br>GPSB=( <u>N</u>  Y) – if Y, uses NAME specified above  |
| (not) RESIDENT                                    | RESIDENT( <u>N</u>  Y)  |
| (not) DOPT  | DOPT( <u>N</u>  Y)  |
| FPATH= <u>NO</u><br>FPATH=(YES,size)              | FP( <u>N</u>  E)<br>“size” not supported for PGM<br>– EMH buffer size defined in CREATE TRAN command                                  |
| LANG= <u>ASSEM</u>                                | LANG( <u>ASSEM</u>  COBOL JAVA PL/I PASCAL)   |
| PGMTYPE=( <u>TP</u>   BATCH,<br>OVLY,<br>1 class) | BMPTYPE=( <u>N</u> ,Y)<br>OVLY no longer supported by IMS<br>“class” not supported for PGMs<br>- class defined in CREATE TRAN command |
| SCHDTYP= <u>SERIAL</u>                            | SCHDTYPE( <u>PARALLEL</u>  SERIAL)  |
| SYSID=(remote,local)                              | Not supported for PGM<br>- defined in CREATE TRAN command   |
|   | DEFAULT(Y) – for descriptors only   |

❖ BMPTYPE(N) includes MSG, JMP, IFP, CICS DRA, ODBA

57

This table shows the default attributes in DFSDSPG1. You cannot update these attributes. To change them, you would need to define a new PGM descriptor with the different attributes.

The left column of the table shows the attribute and its default in the APPLCTN macro. The right column shows it as it appears in the CREATE command. There are some differences in the keyword (spelling), the parameters, and the defaults.

- There are no defaults for the NAME or GPSB name. They must be specified in the CREATE command.
- FP(N|E) identifies this program as an IFP or a full function. If FP(E) is coded, this is equivalent to coding FPATH(YES) on the APPLCTN macro. Note that the “size” parameter, which sets the size of the EMH Buffer, is not code on the CREATE PGM command. It will be set instead on the CREATE TRAN command.
- BMPTYPE replaces PGMTYPE. BMPTYPE(Y) indicates this program is either a BMP (message or non-message driven) or a JBP. BMPTYPE(N) defines everything else (MSG, JMP, IFP, CICS DRA, ODBA). OVLY (overlay) is no longer supported by IMS. Class, which can be specified on the APPLCTN macro and defaults to “1” if not coded, is not set at all on the CREATE PGM command. Instead, the class is set by the CREATE TRAN command.
- The SCHDTYPE default has changed from SERIAL to PARALLEL. To return to SERIAL as the default, create a new PGMDESC with SCHDTYPE(SERIAL) and set DEFAULT(Y).
- SYSID, which on the APPLCTN macro applies to all transactions defined following this macro, is not coded on the CREATE PGM command. Local and remote SYSIDs are coded on the CREATE TRAN command.

**DFSDSTR1 – IMS Transaction Descriptor**

| <b>TRANSACT macro keyword with default</b>   | <b>CREATE TRAN attribute name with default</b>  |
|--|---|
| Previous APPLCTN macro                       | PGM(name)   |
| CODE=name                                    | NAME(name1,name2,...)   |
| AOI=N  | AOCMD( <u>N</u>  CMD TRAN Y)  |
| DCLWA=YES                                    | DCLWA(Y N) – if not specified, defaults to value in DFSDFxxx; if not in DFSDFxxx, default is DCLWA(Y)     |
| EDIT=(UC,name) – no default for edit routine | EDITUC( <u>Y</u>  N)<br>EDITRTN(name)   |
| FPATH=NO<br>FPATH=(Y,size)                   | FP( <u>N</u> )<br>FP(E P) – FP exclusive or FP potential<br>EMHBSZ(12-30720) – default is EMHL size or 12 |
| INQUIRY=(NO,RECOVER)                         | INQ( <u>N</u>  Y)<br>RECOVER( <u>Y</u>  N)  |
| MAXRGN=0                                     | MAXRGN( <u>0</u> ,1-999)  |

- ❖ FP(E) – fast path exclusive transaction; PGM(name) must be FP(E)
- ❖ FP(P) – fast path potential transaction; PGM(name) must be FP(N)

58

This table shows the default attributes in DFSDSTR1. You cannot update these attributes. To change them, you would need to define a new transaction descriptor with the different attributes.

The left column of the table shows the attribute and its default in the TRANSACT macro. The right column shows it as it appears in the CREATE command. There are some differences in the keyword (spelling), the parameters, and the defaults.

-There are no defaults for PGM or NAME. They must be specified on the CREATE command. On the CREATE command, PGM identifies the program with which this transaction is associated.

-PGM - In sysgen, the transaction is associated with the previous APPLCTN macro in the stage 1 input. With DRD, the name of the program is defined by the PGM keyword parameter. The exception is for remote transactions where PGM is not defined.

-DCLWA determines whether DC Log Write Ahead is set for this transaction. On the TRANSACT macro, this defaults to YES. For DRD, if DCLWA is not specified on the CREATE command, it defaults to the value in the default descriptor. The value in the default descriptor, however, has its own default. In DFSDSTR1, and in user-defined descriptors where it is not SET, it will default to the value of DCLWA in the DRD section of DFSDFxxx. If no value is set in DFSDFxxx, and none is set when defining the default descriptor, the descriptor assumes a default of Y(es).

-There are two parameters to replace the EDIT parameter on the TRANSACT macro. EDITUC(Y|N) determines upper case translation (default is “yes” in both cases), and EDITRTN identifies a Transaction Input Edit Routine (no default).

-Full function transactions are defined as FP(N).

-Fast path exclusive transactions are defined as FP(E) and the PGM parameter must refer to a program defined as FP(E).

-Fast path potential transactions are defined as FP(P) and the PGM parameter must refer to a program defined as FP(N).

-For both types of FP transactions, the EMH buffer size is defined by the EMHBSZ parameter (remember, this was not specified on the CREATE PGM command).

## DFSDSTR1 – IMS Transaction Descriptor

| TRANSACT macro keyword with default | CREATE TRAN attribute name with default  |
|-------------------------------------|--|
| MODE=MULT                           | CMTMODE( <b>SNGL</b>  MULT)  |
| MSGTYPE=(MULTSEG,NONRESPONSE,1)     | MSGTYPE( <b>MULTSEG</b>  SNGLSEG)<br>RESP( <b>N</b>  Y)<br>CLASS( <b>1</b>  1-999)   |
| PARLIM=65535                        | PARLIM( <b>65535</b>  0-65535)   |
| PROCLIM=(65535,65535)               | PLCT( <b>65535</b>  number)<br>PLCTTIME( <b>6553500</b>  hundredths of a second)     |
| PRTY=(1,1,65535)                    | NPRI( <b>1</b>  0-14)<br>LPRI( <b>1</b>  0-14)<br>LCT( <b>65535</b>  smaller number) |
| ROUTING=NO                          | DIRROUTE( <b>N</b>  Y)   |
| SCHD=1                              | None - All scheduling is SCHD=3 in V10   |
| SEGNO=0                             | SEGNO( <b>0</b> ,number)   |
| SEGSIZE=0                           | SEGSZ( <b>0</b> ,number)   |

- ❖ Note: PLCTTIME is in hundredths of a second rather than seconds. This applies to both the CREATE and UPDATE commands.

59

-CMTMODE replaces MODE and the default changes to single (SNGL)

-MSGTYPE is replaced by three keywords but the defaults remain the same. Remember that “class” could not be specified on the CREATE PGM command, so it must be set here or default to “1”.

-PROCLIM is replaced by two keywords – PLCT for processing limit count – and PLCTTIME for processing limit count time. The defaults are the same but note that the “time” specification has changed from “seconds” to “hundredths of a second.” The meaning has not changed – it has just become more granular.

-PRTY has been replaced by three keywords – NPRI, LPRI, and LCT for normal and limit priority and for limit count. The defaults and the meanings have not changed.

-The SCHD parameter on the TRANSACT macro is no longer supported in IMS. All transactions are now scheduled as though they were coded with SCHD=3

Information On Demand IMS Version 10

## DFSDSTR1 – IMS Transaction Descriptor

| TRANSACT macro keyword with default                      | CREATE TRAN attribute name with default   |
|--|---|
| SERIAL=NO  | SERIAL( <u>N</u>  Y)  |
| SPA=(size,STRUNC)  | CONV( <u>N</u>  Y)<br>SPASZ(size)<br>SPATRUNC(R S) – if not specified, defaults to value specified in DFSDCxxx; if not in DFSDCxxx, default is SPATRUNC(S)      |
| SYSID=(remote sysid,local sysid) – default is not remote | REMOTE( <u>N</u>  Y)<br>SIDR(1-2036) – no default<br>SIDL(1-2036) – no default<br><b>MSNAME(logical link path name) – can use this instead of SIDR and SIDL</b> |
| (not) WFI  | WFI( <u>N</u> ,Y)   |
|  | TRANSTAT( <u>N</u>  Y)  |
|  | DEFAULT(Y) – for descriptors only   |

❖ MSNAME – if this parameter is coded instead of SIDR and SIDL, the remote and local sysids take on the values of the logical link identified by the MSNAME

60

-SPA has been replaced by three keywords. CONV(Y) defines this transaction as conversational. If conversational, SPASZ sets the spa size, and SPATRUNC sets the SPA truncation rules.

-SYSID has been replaced by three keywords. REMOTE(Y) defines this transaction as remote (MSC). If remote, the SIDR defines the remote SYSID and DISL defines the local SYSID.

-MSNAME is a new parameter that can be used instead of SIDR and SIDL. IF MSNAME is coded, the remote and local SYSIDs are taken from those values in the MSNAME macro. Note that MSC definitions cannot be dynamically created at this time – they must still be defined in the sysgen. However, they can be UPDATED. This is discussed in the Systems Management Enhancements presentation.

-WFI transactions defaulted to not WFI in the sysgen by absence of the WFI keyword. With DRD, WFI(Y|N) determines whether or not a transaction wait-for-input. The default remains not-WFI.

-TRANSTAT – this is a new parameter for transactions in V10. It determines whether statistics are to be kept for individual transaction executions. Prior to V10, or with TRANSTAT(N) – the default – statistics are kept only by PSB schedule (x'08'to x'07'). With TRANSTAT(Y) they are kept for each sync internal. TRANSTAT is discussed in more detail in the System Enhancements section of the class.



## DBFDSRT1 – IMS Routing Code Descriptor

| RTCODE macro keyword with default | CREATE RTC attribute name with default |
|-----------------------------------|--|
| Previous APPLCTN macro            | PGM(name)                              |
| CODE=name                         | NAME(name1,name2,...)                  |
| INQUIRY=NO                        | INQ( <u>N</u> ,Y)                      |
|                                   | DEFAULT(Y) – for descriptors only      |

61

This table shows the default attributes in DBFDSRT1. You cannot update these attributes. To change them, you would need to define a new transaction descriptor with the different attributes.

The left column of the table shows the attribute and its default in the RTCODE macro. The right column shows it as it appears in the CREATE command. There are some differences in the keyword (spelling), the parameters, and the defaults.

-There are no defaults for PGM or NAME. They must be specified on the CREATE command. On the CREATE command, PGM identifies the FPE program with which this routing code is associated.

## Creating MSDBs

- The following procedure can be used to add an MSDB
  - ◆ Do a DBDGEN and ACBGEN for the MSDB
    - Copy staging library to inactive library
  - ◆ Execute full ACBLIB online change
    - Member level OLC (new in V10) does not support MSDBs
  - ◆ Insert segments into the MSDBINIT data set
    - Use MSDB Maintenance Utility (DBFDBMA0)
  - ◆ Issue CREATE DB for the MSDB
    - MSDB cannot be used yet because it has not been loaded
  - ◆ Shut down IMS
  - ◆ Warm start IMS with MSDBLOAD keyword
    - /NRE MSDBLOAD

62

When creating an MSDB, special procedures are required.

-As for all DB types, a DBDGEN and an ACBGEN must be done followed by an online change to make the BHDR (MSDB version of the DBCB) available to IMS. Add the BHDR to ACBLIB by doing a full ACBLIB OLC. Note that you cannot use V10's Member Online Change to add an MSDB BHDR to ACBLIB. It requires a full OLC.

-Insert the MSDB segments into the MSDBINIT data set using the MSDB Maintenance Utility (DBFDBMA0). MSDBs do not support ISRT and so cannot be loaded online.

-Issue the CREATE DB NAME(msdb-name) command to create the DDIR. Since the BHDR will be in ACBLIB, IMS will know that this is an MSDB. However, unlike FF and DEDBs, the database cannot be used yet. IMS loads MSDBs into ECSA only during a restart.

-If it is necessary to use the MSDB immediately, you must shut down IMS normally and warm start it with the MSDBLOAD keyword. IMS will then load the MSDB from the MSDBINIT data set.