

Unit 7 – IMSGEN + Online Change & Dynamic Resource Definition (DRD)

The IMSGEN (also called Sysgen) is a process that IMS Systems Programmers perform in order to define and tailor IMS to meet the needs of their business. As input, a series of macros are coded that define the attributes of IMS resources; output is executable IMS code, a series of non-executable table load modules that describe applications resources, and optionally, members that populate Macro and PROC libraries. The IMS *Online Change* function and utility permits many of the system changes introduced by an IMSGEN to be incorporated into a running IMS system without the need for an IMS restart. In IMS V10, the DRD (Dynamic Resource Definition) function was added that permits an alternate approach to IMSGEN for defining application resources such as databases and transactions.

What you should be able to do

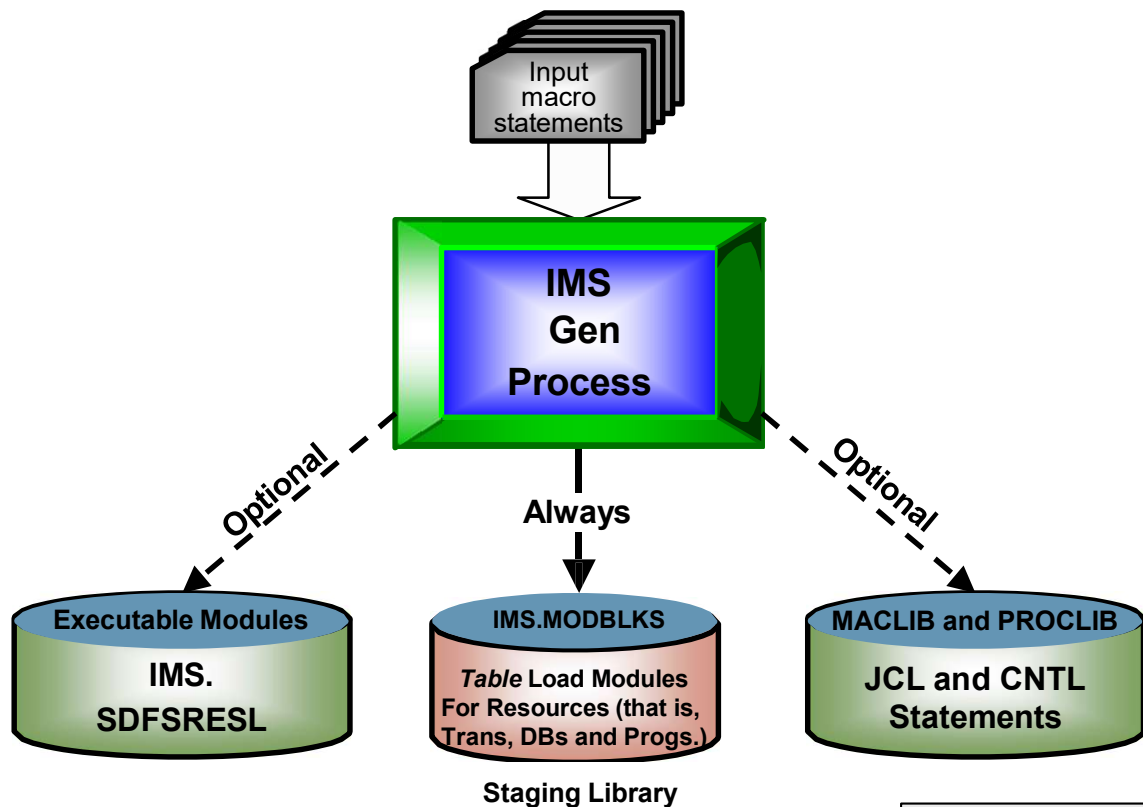
After completing this unit, you should be able to:

- List the functions of IMSGEN
- Understand the role of the different types of IMSGEN and when each of these different types would be appropriate
- Identify the purpose of the different IMSGEN macros
- Describe the difference between the IMSGEN *Stage 1* and *Stage 2*
- Understand how *Staging*, *Active* and *Inactive* libraries are update and modified by the IMS Online Change commands and utility

This chapter doesn't cover the totally changed processing when

**under “IMS Managed ACB” / CATALOG .
This will be part of the class KC220.**

IMSGEN summary

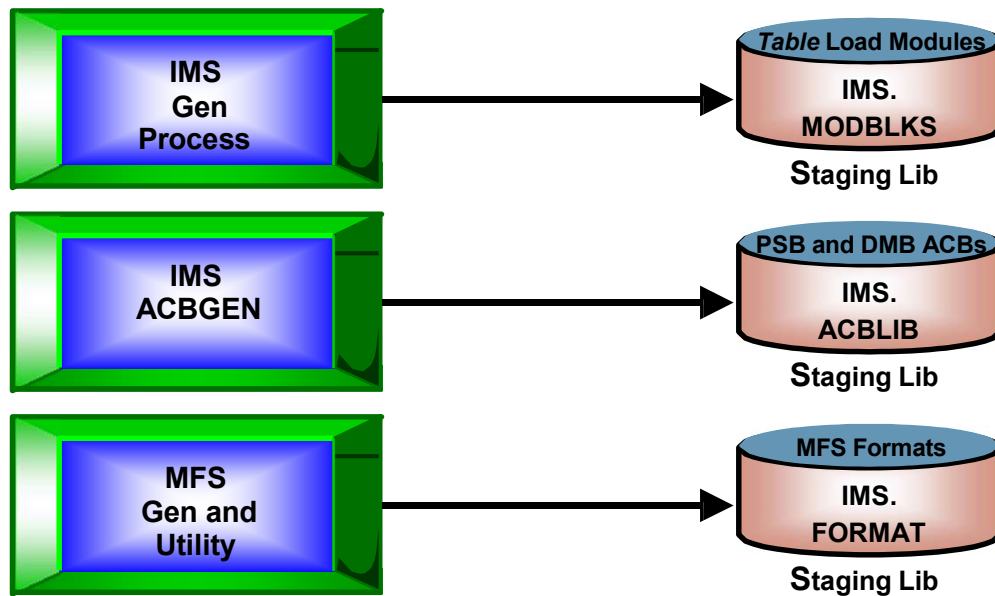


KC110 unit 7 page 2

Notes:

As we have seen, the purpose of the IMSGEN has been to GENERate and customize an IMS system to meet our business needs. All IMSGEN types create new MODBLKS data set members that provide IMS with execution-time information on databases, programs, and transactions (and FastPath Route Codes) that will be used by our applications. The IMS.MODBLKS data set that is updated by IMSGEN is an IMS Staging Library.

All IMS staging libraries

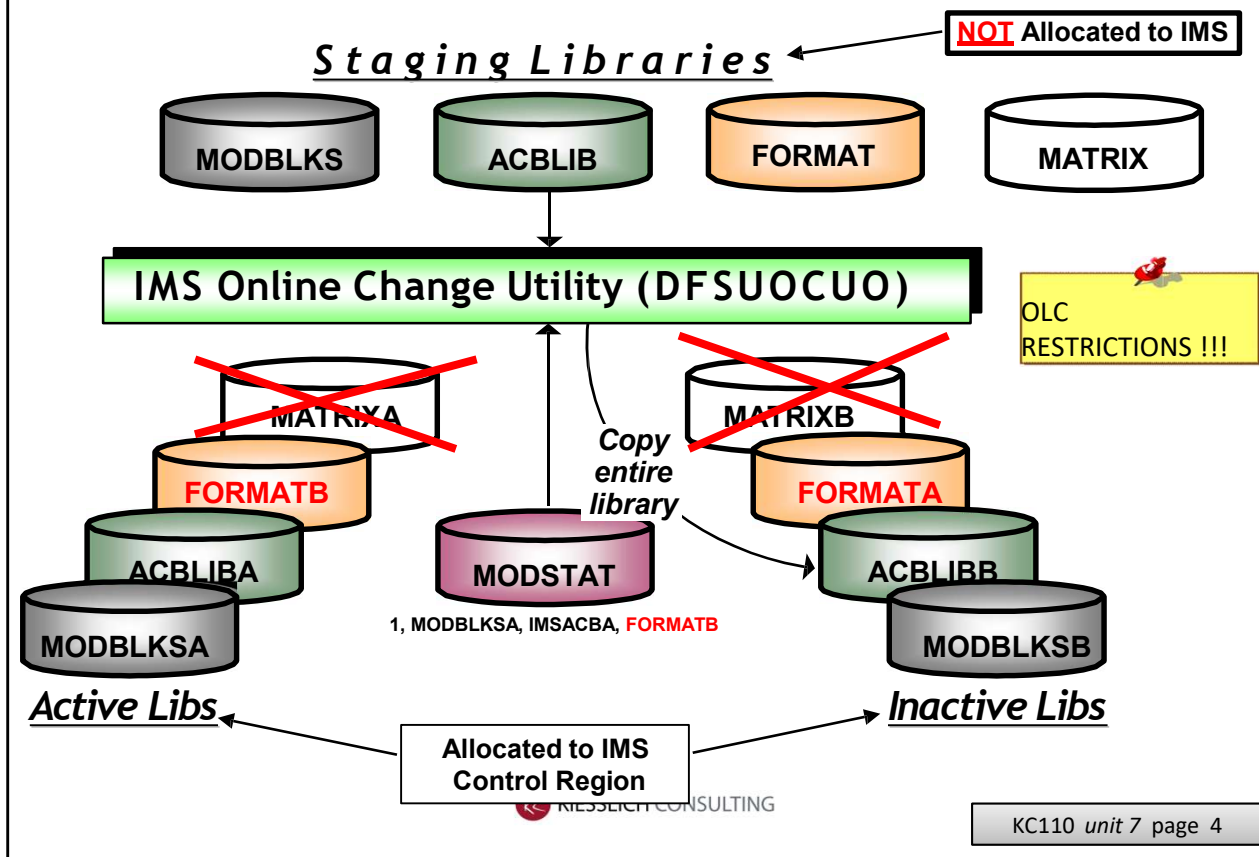


Notes:

There are a number of Staging libraries used by IMS:

- **MODBLKS data set:** Updated by IMS Gen contains information on resources managed by IMS.
- **ACBLIB data set:** Updated by the ACBGEN utility contains execution time information loaded into pools at schedule time on PSBs and DMBs. We will discuss ACBGEN additionally later.
- **FORMAT data set:** Updated by MFSGEN utility contains MFS format blocks (MIDS, MODS, DIFS and DOFS) used by MFS Edit on input and output message segments for MFS Messages.
- **MATRIX data set:** Updated by SMU (Security Maintenance Utility) and contains IMS Security and Password information. This data set was required and the utility was optional in IMS V9 and earlier. Neither the utility or data set are available or used in IMS 10 or later.

IMS Online Change Utility



Notes:

Online Change (OLC) provides the ability, **without an IMS shutdown**, to ADD, CHANGE AND DELETE:

- DATABASEs (requires /DBR [and probably an unload/reload], if changing or deleting)
- APPLCTNs
- TRANSACTs
- FP Routing Codes
- ACBs
- FORMATs
- Terminal, Command and Password SECURITY (MATRIX) (IMS V9 and earlier only)

The IMS Online Change Utility uses the contents of the MODSTAT data set to determine which data sets are inactive. The contents of a Staging Library is then copied to the corresponding INACTIVE data set (for example, the contents of FORMAT is copied to either FORMATA or FORMATB depending on which is INACTIVE).

For the new abilities (ACB MOLC) the STAGING library needs to get (dynamically) ALLOCATED to the IMS online ! See later foils

IMS Online Change Utility



OLC RESTRICTIONS:

- No Online Change for Terminals
 - ETO (or other OEM tool) instead
- No Additional Code can be added:
 - For example, OLC will not add FP to current system

RESIDENT DMBs/PSBs that are added or changed by OLC are NON-RESIDENT until NEXT IMS restart

Online Change Utility JCL

```
2 //OLCUTL      EXEC OLCUTL, TYPE=ACB, IN=S, OUT=U, SOUT='*'
3 XX          PROC  TYPE=, IN=, OUT=, SOUT=A, SYS=, SYS2=, HLI=IMS
4 XXS        EXEC  PGM=DFSUOCU0, PARM=( &TYPE, &IN, &OUT)
5 XXSTEPLIB DD   DSN=&HLI..&SYS2.SDFSRESL, DISP=SHR
IEF653I SUBSTITUTION JCL - DSN=IMS.ACBLIB, DISP=SHR
IEF653I SUBSTITUTION JCL - DSN=IMS.ACBLIB, DISP=SHR
IEF653I SUBSTITUTION JCL - DSN=IMS.ACBLIB, DISP=SHR
IEF653I SUBSTITUTION JCL - DSN=IMS.MODSTAT, DISP=SHR

IEF653I SUBSTITUTION JCL - DSN=IMS.MODBLKS, DISP=SHR
IEF653I SUBSTITUTION JCL - DSN=IMS.MODBLKSA, DISP=SHR
IEF653I SUBSTITUTION JCL - DSN=IMS.MODBLKSB, DISP=SHR

IEF236I ALLOC. FOR AZISI66 S OLCUTL
IEF237I 22A ALLOCATED TO STEPLIB

VPW//PRT004 DD SYSOUT=*, DCB=(RECFM=VBA)

IEBCOPY MESSAGES AND CONTROL STATEMENTS
COPY OUTDD=IMSACBB, INDD=IMSACB
1IEB167I FOLLOWING MEMBER(S) COPIED FROM INPUT DATA SET REFERENCED BY IMSACB
IEB154I ADFAAUDT HAS BEEN SUCCESSFULLY COPIED
IEB154I ADFABCTL HAS BEEN SUCCESSFULLY COPIED
IEB154I ADFABCTP HAS BEEN SUCCESSFULLY COPIED
IEB154I ADFABDDR HAS BEEN SUCCESSFULLY COPIED
```

Notes:

The Online Change Utility determines input and Output Libraries based on the *TYPE=*, *IN=* and *OUT=* parameters furnished and the contents of the MODSTAT data set, then invokes IEBCOPY to copy the members.

Online Change command ACBLIB

example

EXAMPLE 1: ACB CHANGE

Step 1: Perform ACBGEN to the staging ACBLIB.

Step 2: Execute the Online Change Utility (copy Staging into Inactive ACBLIB).

Step 3: Issue the command to prepare the system for an ACBLIB change:

```
/MODIFY PREPARE ACBLIB
```

To list resources that still have work pending:

```
/DIS MODIFY ALL/DBS/PDS/RCS/TRS
```

To discontinue Online Change:

```
/MODIFY ABORT
```

Step 4: Switch Active and Inactive ACB Libraries; for example, activate ACBLIBB (if ACBLIBA was initially the Active Library) as the Active library and deactivate ACBLIBA:

```
/MODIFY COMMIT
```



Notes:

There are several variations of Online Change available in IMS. In this topic, we discuss *Local Online Change* that impacts a single IMS system. There is also the *Global Online Change* function, performed using different commands, that results in changes to multiple IMS systems simultaneously.

MOLC !!!! (see slide 9 and subfol.)

Member Online Change is another (new) way, due to the fact that sometimes the Global OLC against all ACBs doesn't get thru
(heavy workload keeps PSBs and DMBs in working storage pool)

Online Change command APPLCTN

example

EXAMPLE 2: APPLCTN (program) CHANGE

Step 1: Perform MODBLKS-GEN (or other MSGEN type) into staging MODBLKS.

Step 2A: Execute the Online Change Utility (copy Staging into Inactive MODBLKS-Library).
Online Change Utility (copy Staging into Inactive MODBLKS-Library).

Step 2B -I: (Only if still using IMS V9 with SMU Security) Execute SMU IMS Security is used) into staging MATRIX library.

Step 2B -II: (Only if still using IMS V9 with SMU Security) Execute Online Change Utility (copy Staging into Inactive MATRIX-library).

Step 3: (All users) Issue the command to prepare the system for a MODBLKS (and optionally with V9 MATRIX) change:

`/MODIFY PREPARE MODBLKS`

Step 4: Switch Active and Inactive MODBLKS (and optionally with V9 MATRIX) libraries; for example, activate MODBLKSB (if MODBLKSA was initially the Active library) as the Active library and deactivate MODBLKSA:

`/MODIFY COMMIT`



Notes:

If we wanted to perform changes to both ACBLIB *AND* MODBLKS, we can issue command:

`/MODIFY PREPARE ACBLIB MODBLKS`

And then issued the /MOD COMMIT which would result in multiple changed libraries.

However, only 1 Online Change can be active at any given time; another /MOD PREPARE is rejected if one is already in progress for which a /MOD COMMIT (or /MOD ABORT) has not been issued.

Pls also refer to the TYPE 2 commands: INIT GOLC TYPE(...) ... (slides 23 ff)

ACBLIB Member Online Change (MOLC)

- New capability since IMS 10 to add or change one or more members of the ACBLIB without the need to perform an online change on the entire library
 - **Does not support deletion of ACBLIB members**
- Only the resources that are affected by the member online change are quiesced, allowing for more concurrent activity during the online change process than the current full library switch online change
- Coexists with existing full library switch online change capability
- Goal is to improve usability and availability of online change over previous IMS versions

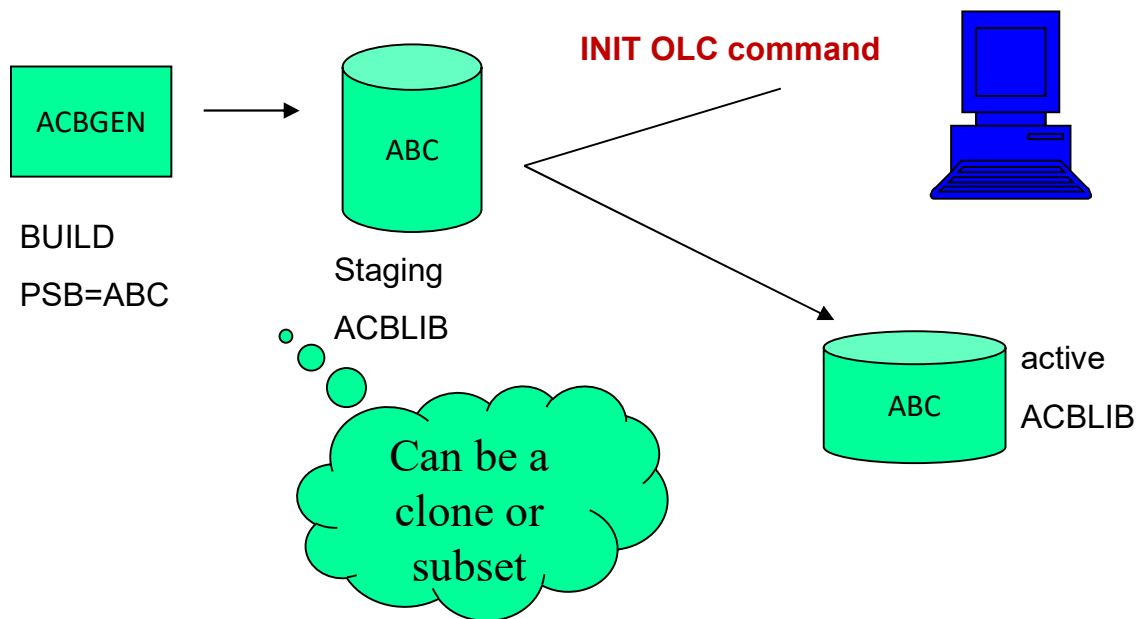
ACBLIB MOLC Requirements ...

- Uses existing IMS libraries (PSBLIB, DBDLIB, ACBLIB) and existing IMS control block generation processes (PSBGEN, DBDGEN, ACBGEN)
- Uses **IMS type-2** commands only
 - INIT OLC PHASE(PREPARE) TYPE(ACBMBR) ...
- Uses staging ACBLIB as the source ACBLIB
 - Full library switch OLC uses inactive ACBLIB (copied from staging ACBLIB)
- Must use OLCSTAT data set (not MODSTAT)
 - IMSplex must be using global online change
 - GOLC needs OLC=GLOBAL in DFSCGxxx or DFSDFxxx
 - Single IMS system cannot use MODSTAT
- CSL with RM required for multiple IMS systems
 - Resource structure recommended but not required
- CSL with SCI and OM required for single IMS system
 - Specify RMENV=N in DFSCGxxx or DFSDFxxx
- Coexists with existing online change ACBLIB full library switch capability
- No coexistence with previous IMS versions
 - All members in an IMSplex need to be at least at IMS 10

TYPE2 CMD : needs CSL ! (SCI + OM)

Staging ACB ? – ok, then DFSMDA into IMS for Dynalloc !

Overview of ACBLIB MOLC Processing



This is an overview diagram of the ACBLIB member online change process. It uses the same ACBGEN process, the staging ACBLIB, and enhanced type-2 INIT OLC commands. Changes go directly into the active ACBLIB.

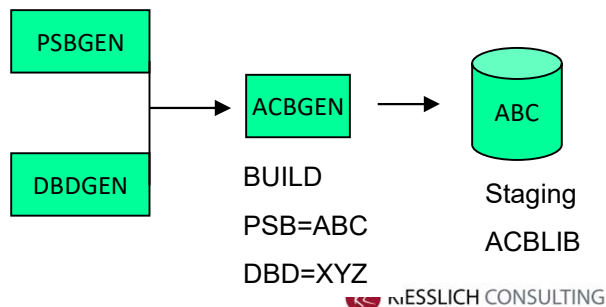
Steps to perform an ACBLIB MOLC (1)

Step 1: Build updated member(s)

- Update PSB source and/or DBD source
 - Perform PSBGEN/DBDGEN for affected members
 - Perform ACBGEN into the staging ACBLIB library for affected members
 - Must use new BLDPSB=YES option on BUILD DBD= statement to get all PSBs rebuilt that are affected by a DBD change
 - BUILD DBD=(dbdname,...), BLDPSB=YES|NO
- BLDPSB=YES is the default

BUILD DBD=(CUSTOMER,ORDER), BLDPSB=YES

Staging ACBLIB can be a clone or a subset of the active ACBLIB
ACBGEN enhancement will include IMS version in the utility output
Allows user to be able to check the IMS library level used by ACBGEN



KC110 unit 7 page 12

The BLDPSB=YES option is new with IMS 10 and ACBLIB member online change. It indicates that ACBGEN rebuilds all PSBs that reference the changed DBD on the BUILD DBD=dbdname statement into the staging ACBLIB. Member online change requires that all PSBs be rebuilt (flag set in PDIR) or it will fail. For performance reasons, these relationships are found during the execution of ACBGEN in batch versus having to discover these relationships online when ACBLIB member online change is invoked. However, this means there may be additional processing done at ACBGEN time and the ACBGEN utility may run longer. BLDPSB=YES is the default.

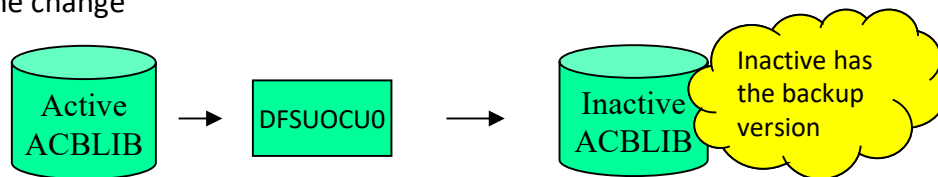
In previous IMS versions, ACBGEN worked as if BLDPSB=NO was specified. BLDPSB=NO indicates that ACBGEN does not rebuild PSBs that reference the changed DBD if the changed DBD does not change the physical structure of the DBD. The DBD is rebuilt but PSBs are not necessarily rebuilt (ex. exit/randomizer change). You can still specify BLDPSB=NO if you are not planning to use ACBLIB member online change. BLDPSB=NO is not the default, so this operates differently than in previous versions. It must be specified if you want this capability.

The ACBGEN utility does not clear out the staging ACBLIB. The ACBGEN utility only updates the members in the staging ACBLIB that are specified in the BUILD DBD and BUILD PSB statements.

Steps to perform an ACBLIB MOLC (2)

Step 2: Create a Backup Copy (optional)

- Copy active library to the inactive library using the OLC copy utility (DFSUOCU0)
 - New TYPE=ACTVACB to create backup copy of the active ACBLIB
 - Optional to use DFSUOCU0, then user copy procedures can do copy
- Backup copy recommended (though optional) because ACBLIB member online change updates member(s) directly into the active ACBLIB dataset
- If a problem happens with ACBLIB member online change, then a full library switch online change could be used for recovery using this backup copy in the inactive library
- In an IMSplex where IMS subsystems share ACBLIBs (active/inactive), DFSUOCU0 needs to be executed on one IMS in the IMSplex
- In an IMSplex where IMS subsystems do not share ACBLIBs (active/inactive), DFSUOCU0 needs to be executed on every IMS in the IMSplex
- DFSUOCU0 TYPE=ACTVACB can also be used to synchronize the active ACBLIB to the inactive ACBLIB before doing a full library switch online change to ensure that members added to the active via MOLC will be included in the full library switch online change



There is a new option on the OLC copy utility (DFSUOCU0) called TYPE=ACTVACB. This copies the active ACBLIB to the inactive ACBLIB to create a backup that can be used in the event of a problem with ACBLIB member online change. This 'copied' inactive ACBLIB can be used with a full library switch online change to recover from the ACBLIB member online change failure.

This new TYPE=ACTVACB OLC copy utility can also be used to synchronize the active to the inactive library if you have been doing many member online changes and then want to switch to doing a full library switch online change. This will ensure that all changes from any member online changes will be included in the full library switch online change.

Sample JCL for DFSUOCU0 for ACBLIB Member Online Change

```
//DFSUOCU0 JOB          MSGLEVEL=1,MSGCLASS=A,CLASS=K  
//STEP1    EXEC        OLCUTL, TYPE=ACTVACB,OUT=G,SOUT=*,  
//          OLCLOCL='DUMMY',,OLCGLBL=,SYS=  
//
```

The ACTIVE ACBLIB is copied to the INACTIVE ACBLIB (OUT=G)

```
//DFSUOCU0 JOB          MSGLEVEL=1,MSGCLASS=A,CLASS=K  
//STEP1    EXEC        OLCUTL,TYPE=ACTVACB,OUT=O,SOUT=*,  
//          OLCLOCL='DUMMY',,OLCGLBL=,SYS=  
//
```

The ACTIVE ACBLIB is copied to the ACBLIB specified on the
IMSACBO DD card in the OLCUTL procedure (OUT=O).

Here is a sample of the JCL that can be used to run DFSUOCU0 to create a backup copy for the ACBLIB member online change process.

Steps to perform an ACBLIB MOLC (3)

Step 3: Bring members online via OLC INIT command

- First time setup (discussed later)
 - Allocation of staging library
 - Specify usage of ACBLIB – shared or dedicated
- Use INITIATE OLC PHASE(PREPARE) TYPE(ACBMBR) NAME(xxx,yyy) command to specify ACBs that ACBLIB member online change will process
 - Each member specified will be read from the staging ACBLIB, validation will be performed, and a prepare list will be created
- Use INITIATE OLC PHASE(COMMIT) to activate the changes
 - Updated members will be written to the active ACBLIB after verification that all members can be written successfully to the active ACBLIB
 - Only those resources affected by the change are quiesced
 - Updated members are copied directly into the active ACBLIB
 - New members go into the first active ACBLIB concatenation
 - Changed members go into the correct active ACBLIB concatenation
 - DOPT PSBs are not supported
- Process is similar to existing logic to determine what has changed

TYPE(ACBMBR) cannot be specified with TYPE(ACBLIB,FMTLIB,MODBLKS) in the same INIT OLC PHASE(PREPARE) command. TYPE(ACBLIB,FMTLIB,MODBLKS) and TYPE(ALL) are for full library switch online change and the source ACBLIB is the active ACBLIB. TYPE(ACBMBR) is for ACBLIB member online change and the source ACBLIB is the staging ACBLIB.

You need only specify the ACB member(s) with changes in the INIT OLC PHASE(PREPARE) command; IMS will find all the necessary members that are associated with the specified member(s). For example, there is an update to DBDX and an ACBGEN BUILD DBD=DBDX,BLDPSB=Y is done. DBDX is used by members PSBA, PSBB, and PSBC, so these PSBs are rebuilt by ACBGEN. The following command can be entered to bring member DBDX online, along with members PSBA, PSBB, and PSBC:

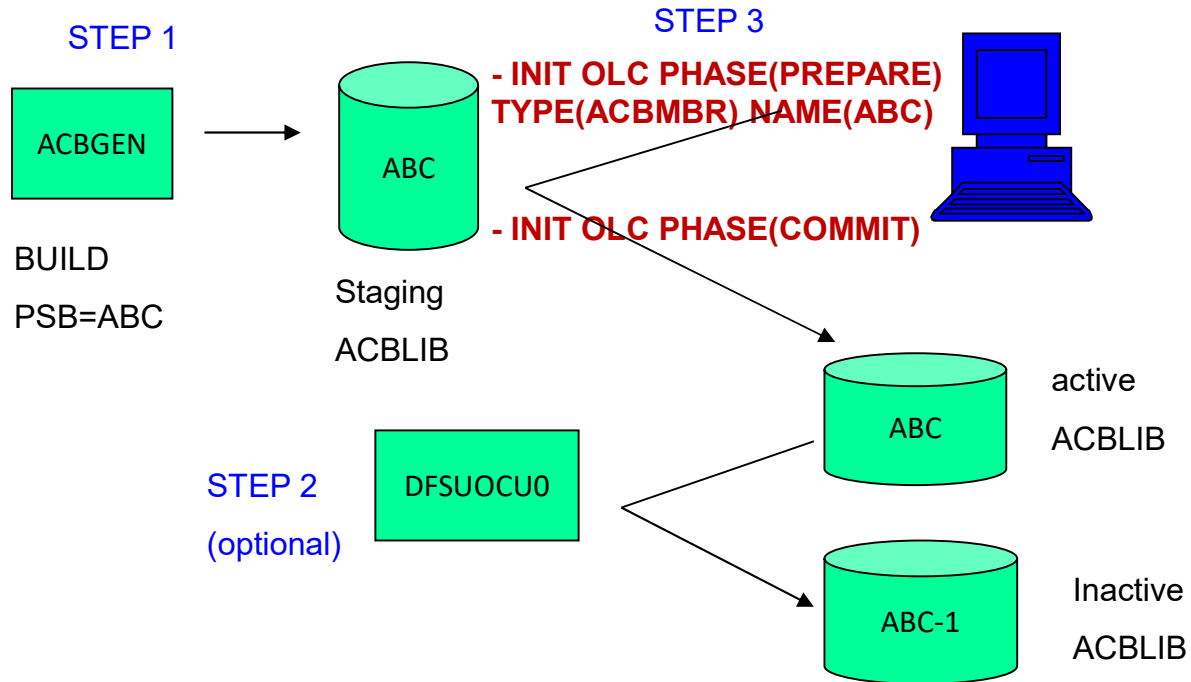
```
INIT OLC PHASE(PREPARE) TYPE (ACBMBR) NAME(DBDX)
```

Dynamic Option PSBs (DOPT) are not supported by ACBLIB member OLC. Therefore when ACBLIB member OLC processes a changed DBD that references a DOPT PSB, it does not copy the DOPT PSB from the staging ACBLIB to the active ACBLIB, though the changed DBD referenced by this DOPT PSB will be copied to the active ACBLIB. The INITIATE OLC PREPARE command output shows for the DOPT PSB entry a completion code indicating the DOPT PSB will not be copied to the active ACBLIB; COMMIT processing will still be successful if this occurs.

If a GPSB is specified in the INIT OLC PHASE(PREPARE) command, it will be copied to the active ACBLIB and its TTR address will not be refreshed. However, IMS only uses its own GPSB skeletal PSB and will not use the one in the ACBLIB.

If the staging ACBLIB is a subset of the active ACBLIB, the member OLC process reads only the directory entries for those ACB members in the staging ACBLIB. If the staging ACBLIB is a clone of the active ACBLIB, the member OLC process reads all the directory entries in the staging ACBLIB.

Total Process for ACBLIB MOLC



This is a diagram of the steps needed to perform an ACBLIB member online change.

ACB MOLC Prereqs : Specify Setup Parameters

- Ensure that the staging ACBLIB can be allocated by your IMS control region since it is now used by ACBLIB member online change

- Create DFSMDA member for ACBLIB staging library

```
DFSMDA TYPE=INITIAL
```

```
DFSMDA TYPE=IMSACB,DSNAME=STAGING.LIBRARY DFSMDA  
TYPE=FINAL
```

- Or, add an IMSACB DD statement to your IMS procedure
 - If used, takes precedence over DFSMDA member

```
//IMSACB DD DSN=STAGING.LIBRARY, DISP=SHR
```

Here is a sample DFSMDA member for dynamic allocation of the ACBLIB staging library for ACBLIB member online change. This is needed because the staging library was not used in online change in previous IMS versions; therefore, it would not be in the IMS Control Region JCL. This new DFSMDA member of TYPE=IMSACB is created for the staging ACBLIB. With TYPE=IMSACB, users only need to specify the data set name for the staging ACBLIB. A dynamic allocation member with the name of IMSACB will be created for the staging ACBLIB. Because of this, no database MDA members should be created with a DDNAME=IMSACB. IMS dynamically allocates the staging ACBLIB with DISP=SHR.

If you don't set up a DFSMDA member for the staging library, you will need to include a DD statement for the staging library in your IMS control region procedure. If the DD statement exists, it will override any DFSMDA member.

If dynamic allocation fails, the OLC PREPARE process will fail and return a return/reason code and completion code indicating the error. The user will need to issue a TERMINATE OLC command to abort the online change in progress, and then reissue the INIT OLC PHASE(PREPARE TYPE(ACBMBR) command.

ACB MOLC Prereqs : Specify Setup Parameters ...

- Specify whether ACBLIB is shared or dedicated in the IMSplex
 - ACBSHR=Y/N in DFSCGxxx PROCLIB member or DFSDFxxx PROCLIB member
 - Y – indicates that all of the IMSs in the OLCSTAT are using the same active and inactive ACBLIB
 - the default
 - N – indicates that each IMS in the OLCSTAT is using its own dedicated active and inactive ACBLIB
 - ACBSHR does not apply to the staging ACBLIB because it is not updated by ACBLIB member online change
 - The staging ACBLIB can be shared or dedicated
 - Multiple copies must be identical
 - All IMSs in the IMSplex sharing the OLCSTAT data set must specify the same value of ACBSHR=

If ACBSHR= is coded in both the DFSCGxxx and the DFSDFxxx PROCLIB members, IMS uses the ACBSHR= value defined in the DFSCGxxx PROCLIB member.

During the INIT OLC PHASE(PREPARE) TYPE(ACBMBR) processing, the ACBSHR= value of the command master is sent to all IMSs participating in the member OLC. If any IMS ACBSHR= value is different than that of the command master, the INIT OLC PHASE(PREPARE) TYPE(ACBMBR) command will fail.

The ACBSHR= value of the IMSs participating in the member online change will be displayed in the output of the INIT OLC PHASE(PREPARE) command.

If you have multiple copies of the staging ACBLIB in your IMSplex, they must be identical. All the IMSs read from the staging library to determine the changed members affected. That's why they have to use the same dataset or an exact copy of it.

ACB MOLC IMSPLEX Implications

- The ACBLIB member online change process will be coordinated among all IMSs sharing the OLCSTAT data set
 - OM chooses an IMS system to be the command master IMS
 - The command master IMS will coordinate the member OLC process with other IMSs sharing the OLCSTAT data set using RM
 - ACBSHR=Y/N in DFSCGxxx PROCLIB member or DFSDFxxx PROCLIB member specifies whether or not the active and inactive ACBLIB is shared among the IMSs in the OLCSTAT data set
 - All IMSs in the IMSplex sharing the OLCSTAT data set must specify the same value of ACBSHR=
 - Member OLC will update all the active ACBLIBs, whether shared or non-shared

ACB MOLC Prepare Phase

■ Build updated members

- INIT OLC PHASE(PREPARE) TYPE(ACBMBR) NAME(acb1,acb2,...)
- Determine what changed
 - A DBD change will affect all the PSBs that refer to that DBD
 - Need to use BLDPSB=YES for new/changed DBDs to rebuild all related PSBs
 - Will only process members in the list and associated resources
 - All members in the list must be processed successfully or PREPARE fails
- Create an add/change list for OLC process
- Quiesce only affected resources
- If PREPARE phase fails, the TERMINATE OLC command needs to be issued

ACB MOLC Commit Phase 1 + 2

- Update active ACBLIB
 - INIT OLC PHASE(COMMIT)
 - Updated member(s) are written to the active ACBLIB with encrypted name(s)
 - Member name is translated to hex characters and written as new member to the active ACBLIB
 - Existing members are not updated
 - Ensures all new/updated members fit in the active ACBLIB
 - Prevents x37 abends from creating inconsistent members in the active ACBLIB
 - If Commit Phase 1 fails, the TERMINATE OLC command needs to be issued
- Commits all members into the active ACBLIB
 - Delete old member(s)
 - Rename encrypted member(s) to the actual name(s)
 - Refresh TTRs for changed members
 - IMS now processes with new and/or changed members

Type2 Commands Used with ACBLIB MOLC

- INITIATE OLC PHASE(PREPARE) TYPE(ACBMBR) NAME(mbrname)
- QUERY OLC SHOW(RSCLIST)
- QUERY MEMBER TYPE(IMS)
- INITIATE OLC PHASE(COMMIT)
- TERMINATE OLC

There are five type-2 commands that can be used with ACBLIB member online change.

INIT OLC PHASE(PREPARE) Command

All IMSs in OLCSTAT must be V10 and up, otherwise command will fail

TYPE(ACBMBR) parameter

Specifies that a member online change is to be performed for ACBLIB members included in the NAME parameter

Mutually exclusive with any other TYPE parameter, including TYPE(ALL)

```
INITIATE OLC PHASE (PREPARE) TYPE(ACBMBR)  
NAME (acbmbr1,acbmbr2,..) OPTION(FRCNRML
```

INIT OLC PHASE(PREPARE) Command ...

- OPTION(FRCNRML) parameter
 - Only valid OPTION when TYPE(ACBMBR) specified
 - Allows a member online change to be processed if any IMS in OLCSTAT is shutdown normally
 - IMS that is down removed from OLCSTAT data set
 - When this IMS restarts and it has missed a member level online change, message DFS3433W ACBLIB MEMBER OLD ID MISMATCH MOLCID=yyyydddhhmmss is issued indicating a member level online change modify ID mismatch (MOLCID)
 - No OPTION(FRCABND) support – if an IMS is down due to an abend the INIT OLC PHASE(PREPARE) TYPE(ACBMBR) command will fail; the user must do a full library switch OLC

The FRCNRML (Force Normal) option on the INIT OLC PHASE(PREPARE) command allows the user to do an online change even if one of the IMSs in the OLCSTAT data set is down normally (not via an abend).

The FRCABND (Force Abend) option on the INIT OLC PHASE(PREPARE) command is not supported.

When using the FRCNRML option and ACBSHR=N (separate, non-shared ACBLIBs), this member online change will be missing from that IMSs active ACBLIB. The user must ensure that an ACBGEN utility is executed to put those missing member OLC changes into the active ACBLIB before that IMS is restarted.

When using the FRCNRML option and ACBSHR=Y (single, shared ACBLIB), since there is only one ACBLIB, it will contain all member online changes that were processed while any IMS was down normally. The user does not have any additional ACBLIB processing to do.

INIT OLC PHASE(PREPARE) Command

Example

Response for: INIT OLC PHASE(PREPARE) TYPE(ACBMBR)
NAME(OLCDB105,OLCDX111)

MbrName	Member	CC	ACBSHR	DBDName	PSBName	Add	Chng
---------	--------	----	--------	---------	---------	-----	------

IMS2	IMS1	0	N				
IMS2	IMS1	0		OLCDB105		Y	
IMS2	IMS1	0		OLCDX111			Y
IMS2	IMS1	0		OLCDB111			Y
IMS2	IMS1	0		OLCDI111			Y
IMS2	IMS1	0			OLCPB105	Y	
IMS2	IMS1	0			OLCPB111		Y
IMS2	IMS2	0	N				
IMS2	IMS2	0		OLCDB105		Y	
IMS2	IMS2	0		OLCDX111			Y
IMS2	IMS2	0		OLCDB111			Y
IMS2	IMS2	0		OLCDI111			Y
IMS2	IMS2	0			OLCPB105	Y	
IMS2	IMS2	0			OLCPB111		Y

Output from all participating IMSs is sent to the master IMS, which builds the output lines for all the resources.

QUERY OLC Command

SHOW(RSCLIST) parameter support added

- Valid only when a TYPE(ACBMBR) online change is in progress after an INIT OLC PHASE(PREPARE) has been completed
- Returns the ACBLIB members that will be added/copied to or changed in the active ACBLIB
- Mutually exclusive with SHOW(ALL)

SHOW(ALL) includes

SHOW(ACTVLIB,DSN,LASTOLC,MBRLIST,MODID)

Need not specify LIBRARY(OLCSTAT) as is the case with global OLC

QUERY OLC SHOW(RSCLIST)

QUERY OLC SHOW(RSCLIST) Command Example

Response for: QRY OLC SHOW(RSCLIST)

MbrName	CC	DBDName	PSBName	Add	Chng
IMS1	0	OLCDB105		Y	
IMS1	0	OLCDX111			Y
IMS1	0	OLCDB111			Y
IMS1	0	OLCDI111			Y
IMS1	0		OLCPB105	Y	
IMS1	0		OLCPB111		Y
IMS2	0	OLCDB105		Y	
IMS2	0	OLCDX111			Y
IMS2	0	OLCDB111			Y
IMS2	0	OLCDI111			Y
IMS2	0		OLCPB105	Y	
IMS2	0		OLCPB111		Y

The DBDs and PSBs to be added or changed by the ACBLIB member online change are listed here.

QUERY MEMBER TYPE(IMS) Command

Can be issued after an INIT OLC PHASE(PREPARE) command has been successfully completed

New OLCMACB status added to command response to indicate that a member online change is in progress

All other global online change status will be returned in command response as well

QUERY MEMBER TYPE(IMS) SHOW(ALL)

QUERY MEMBER TYPE(IMS) Command Example

Response for: QRY MEMBER TYPE(IMS) SHOW(ALL)

MrName	CC	Type	Status	LclAttr	LclStat	ModId
IMS1	0	IMS		GBLOLC	OLCPREPC,SECCMD,SECMSG	1
IMS2	0	IMS	OLCMACB,OLCPREPC			
IMS2	0	IMS		GBLOLC	OLCPREPC,SECCMD,SECMSG	1

OLCMACB is a new status indicating a ACBLIB member online change is in progress.

INIT OLC PHASE(COMMIT) Command

Enhanced to support TYPE(ACBMBR) online change
Commits all members that were specified on the INIT OLC PHASE(PREPARE)
command into the active ACBLIB

INIT OLC PHASE(COMMIT)

Response for: INIT OLC PHASE(COMMIT)

MbrName Member CC

IMS2	IMS1	0
------	------	---

IMS2	IMS2	0
------	------	---

TERMINATE OLC Command

- Used to abort a member online change that is in progress
- Can be issued after an INIT OLC PHASE(PREPARE) TYPE(ACBMBR) error or after an INIT OLC PHASE(COMMIT) error to terminate a member online change
- New versions of updated members will be deleted from the active ACBLIB, as long as they have not yet been committed with an INIT OLC PHASE(COMMIT) command
- Cannot be issued after the OLCSTAT data set has been updated after COMMIT complete
- Will write X'7010' log record indicating that the member online change has been terminated

TERMINATE OLC

TERMINATE OLC Command Example

Response for: TERMINATE OLC

MbrName	Member	CC
---------	--------	----

--

IMS2	IMS1	0
------	------	---

IMS2	IMS2	0
------	------	---

ACBLIB MOLC Migration

- Must be using OLCSTAT, not MODSTAT, for online change
 - Requires CSL to be set up
- OLCSTAT data set must be reformatted in IMS 10
 - OLCSTAT data set header increased from 80 bytes to 128 bytes
 - Coexistence APARS for IMS V8 (PK23401) and IMS V9 (PK23402) will allow IMS 10 to tolerate a V8/V9 OLCSTAT data set
 - Called Global Online Change Coexistence SPE
 - Global Online Change utility (DFSUOLC0) must be run
- IMSs in the IMSplex must be at IMS 10 to perform an ACBLIB member online change
- Supports XRF, FDBR, and DBCTL Warm Standby (no impact)
- Does not support RSR
- Does not support MSDBs

PK23401 and PK23402 have action holdcards in them with directions for applying the Global Online Change Coexistence SPE. Here is a portion of the planned APAR text:

Please follow the steps as outlined below when applying this maintenance:

1. Apply APAR maintenance to each V8/V9 IMS.
2. Note the current OLCSTAT dataset contents such as the library suffixes, IMS list information & modify ID.
3. Run Global Online Change Utility (DFSUOLC0) with FUNC=INI to initialize the OLCSTAT dataset,
to recreate it with the saved contents before APAR maintenance was applied.

Note that the Global Online Change Utility (DFSUOLC0) can be run while the IMSplex is up.

ACBLIB MOLC Summary

- Provides improved availability during online change process by limiting quiesced resources to only those being changed by specified ACBs
- ACBLIB member online change should be more efficient than full library switch global online change
- Complements DRD MODBLKS capability
 - Add new database definition or application definition via DRD
 - Activate associated DBDs/PSBs/ACBs via ACBLIB member online change
 - In any sequence

Dynamic allocation of ACBLIB data sets (1)

- IMS outage required prior to IMS 11 if you need to resize the ACBLIB data sets since they are allocated via JCL
- In IMS 11, DFSMDA can optionally be used instead of JCL allocation
 - Inactive ACBLIB dynamically allocated only when needed
 - Active ACBLIB will be allocated all the time
- Now the inactive ACBLIB can be resized when necessary
 - Perform an online change to switch the newly resized inactive ACBLIB to the active ACBLIB providing more space now
 - Resize the new inactive ACBLIB (previously too small active ACBLIB)
- Benefits
 - Manageability of ACBLIB is now improved
 - Users can increase the size of ACBLIB data sets without an outage
- At control region initialization, IMS checks for the presence of the IMSACBA / IMSACBB DD statements
 - If they exist, no DFSMDA members are used (same as pre-IMS 11)
 - If they do not exist, DFSMDA members are used

Prior to IMS 11 and this new capability, if you need to resize the ACBLIB data sets, you must take down IMS to do that because they are allocated via JCL, causing unavailability to end users. IMS 11 provides the optional capability to use DFSMDA dynamic allocation rather than JCL to allocate the ACBLIB data sets. Though dynamic allocation will be used for the active ACBLIB, it will be allocated all the time. However, the inactive ACBLIB will only be allocated when needed; therefore, since the inactive will be deallocated most of the time, it can be resized. Then a subsequent online change can be used to switch the resized inactive ACBLIB to be the active ACBLIB and the resizing will take effect. The now inactive ACBLIB (previously active ACBLIB) can then be resized to match the new active ACBLIB.

This capability improves the manageability of the ACBLIB data sets.

Are you aware of ???

Any Need for ??

Well ... only , if your ACBLB gets a lot changes or new ACBs.

.

Dynamic allocation of ACBLIB data sets (2)

Need to create a DFSMDA member for each of the IMSACBA and IMSACBB data set concatenations

```
DFSMDA TYPE=INITIAL
DFSMDA TYPE=IMSACBA
DFSMDA
TYPE=DATASET,DSNAME=IMS.IMSPLEX.V11.ACBLIBA.DOPT
DFSMDA
TYPE=DATASET,DSNAME=IMS.IMSPLEX.V11.ACBLIBA
DFSMDA TYPE=IMSACBB
DFSMDA
TYPE=DATASET,DSNAME=IMS.IMSPLEX.V11.ACBLIBA.DOPT
DFSMDA
TYPE=DATASET,DSNAME=IMS.IMSPLEX.V11.ACBLIBB
DFSMDA TYPE=FINAL
END
```



Dynamic allocation statements for IMSACBA & IMSACBB data sets can be combined in the same job. They **cannot** be combined with **other** statements to dynamically allocate any other IMS data set.

KC110 unit 7 page 36

Here are examples of using the DFSMDA macro to create members that will be used for dynamically allocating the ACBLIBA and ACBLIBB data sets. There are two new DFSMDA TYPEs, DFSMDA TYPE=IMSACBA and DFSMDA TYPE=IMSACBB, that will be used for dynamically allocating the two ACBLIB data sets. These examples show having two concatenations for each of the ACBLIB data sets.

The DFSMDA members are stored in one of the STEPLIB concatenations of the IMS procedure. IMSDALIB is also supported.

Dynamic allocation of ACBLIB data sets (3)

- Only the active ACBLIB datasets are allocated at control region initialization !
- The inactive ACBLIB data sets are dynamically allocated during an online change process !
- After an online change, the inactive ACBLIB data sets are deallocated
- The same DFSMDA member is used to allocate data sets in both CTL and DLISAS !
 - **No inconsistencies between CTL and DLISAS can exist**
- Dynamic allocation of ACBLIB data sets is supported in all online configurations (IMS/TM, DBCTL, DCCTL, SAS and non-SAS, XRF, FDBR)
 - Not supported in batch
- Can be used for correcting errors with an unusable inactive ACBLIB
 - Due to error in copying staging ACBLIB to inactive ACBLIB in preparation for an online change
- Additional data sets can be added or changes made to the current data sets in the inactive ACBLIB concatenation

KC110 unit 7 page 37

With DFSMDA, the active ACBLIB data set is dynamically allocated at IMS initialization and remains allocated for the duration of the IMS subsystem. With DFSMDA, the inactive ACBLIB is only allocated when needed to perform an online change. Once the online change is complete, the inactive ACBLIB data set will be deallocated.

Without dynamic allocation of ACBLIB data sets, both CTL and DLISAS need DD statements for the ACBLIBs so there can be inconsistencies between these two sets of DD statements. When DFSMDA is used, these inconsistencies cannot occur.

Dynamic allocation of ACBLIB data sets is not supported in batch. It is supported in all online configurations (IMS/TM, DBCTL, DCCTL, SAS and non-SAS, XRF, FDBR).

Dynamic allocation of ACBLIB data sets (4)

/DISPLAY MODIFY ALL command output now indicates new status for IMSACBA and IMSACBB

(A) Active, (I) Inactive, (U) Unallocated, () DFSMDA not used

LIBRARY	IMSACBA	(A)	IMSTESTG.DELTA1
		(A)	IMSTESTG.IMS10AC.ACBLIB1
		(A)	IMSTESTG.IMS10A.ACBLIB1
LIBRARY	FORMATA	(A)	IMSTESTG.MFS.FORMAT1
		(A)	IMSTESTG.MFS.FORMAT2
		(A)	IMSTESTG.FMT1
LIBRARY	MODBLKSA	(A)	IMSTESTG.MODBLKS1
LIBRARY	IMSACBB	(U)	IMSTESTG.DELTA2
		(U)	IMSTESTG.IMS10AC.ACBLIB2
		(U)	IMSTESTG.IMS10A.ACBLIB2
OR	LIBRARY	IMSACBB	() NO DFSMDA MEMBER
	LIBRARY	FORMATB	(I) IMSTESTG.MFS.FORMAT3
		(I)	IMSTESTG.MFS.FORMAT4
		(I)	IMSTESTG.FMT1
	LIBRARY	MODBLKSB	(I) IMSTESTG.MODBLKS2
	DISPLAY MODIFY COMPLETE *08230/110121* STAGE		

The /DISPLAY MODIFY ALL command output has been enhanced for the inactive ACBLIB to show a new status 'U' meaning unallocated when using DFSMDA or () if there is no DFSMDA member for the inactive ACBLIB data sets.

Dynamic Resource Definition (DRD)

What this unit is about

The IMSGEN (also called Sysgen) is a process that IMS Systems Programmers perform in order to define and tailor IMS to meet the needs of their business. As input, a series of macros are coded that define the attributes of IMS resources; output is executable IMS code, a series of non-executable table load modules that describe applications resources, and optionally, members that populate Macro and PROC libraries. The IMS *Online Change* function and utility permits many of the system changes introduced by an IMSGEN to be incorporated into a running IMS system without the need for an IMS restart. In IMS V10, the DRD (Dynamic Resource Definition) function was added that permits an alternate approach to IMSGEN for defining application resources such as databases and transactions.

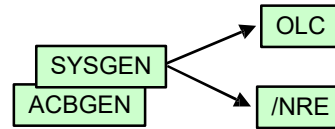
What you should be able to do

After completing this unit, you should be able to:

- List the functions of IMSGEN
- Understand the role of the different types of IMSGEN and when each of these different types would be appropriate
- Identify the purpose of the different IMSGEN macros
- Describe the difference between the IMSGEN *Stage 1* and *Stage 2*
- Understand how *Staging*, *Active* and *Inactive* libraries are update and modified by the IMS Online Change commands and utility

Modifying resource definitions without DRD

- To add, change, or delete these resources in a running IMS system requires:
 - MODBLKS Sysgen
 - ACBGEN (if DATABASE or APPLCTN change)
 - < and >
 - Online change process
 - At some point during MODBLKS OLC process, all activity is quiesced
 - One resource can prevent entire process from completing
 - < or >
 - IMS restart
 - Might not have available window to recycle IMS
- Process is more complex with multiple IMSs running in an IMSPLEX:
 - Coordinated online change
 - < or >
 - Multiple coordinated system restarts



Notes:

Dynamic Resource Definition (DRD) was introduced by IMS V10. Prior to DRD, to add, change, or delete a resource required an IMS sysgen and either an IMS restart or an online change. Both of these processes resulted in significant (or at least some) unavailability to the end user. In a multi-IMSPLEX, it was even more complex because of the need to coordinated the changes (and the unavailability) across multiple IMSs.

What are Online Change limitations?

- System definition process required:
 - Coding IMS Gen macros
 - Running the complete IMS Gen process
- Adding or changing DBDs and PSBs also requires:
 - DBDGEN, PSBGEN, ACBGEN, and ACBLIB OLC
- Online change process requires three MODBLKS data sets:
 - Staging, Active, and Inactive versions of MODBLKS
 - Online change could also be performed for ACBLIB, FMTLIB, and in V9 MATRIX
 - ACBLIB and FMTLIB changes are not reflected in an IMS Gen or DRD
- Online change process can be difficult:
 - Prepare process can take a long time
 - Resources being changed must be *quiesced*
 - Stop queuing and drain queues
 - Commit process:
 - Stops *all* scheduling, at least momentarily
 - Might be difficult to resolve work-in-progress (CICS, ODBA, BMPs)

Notes:

The problem, of course, is that the sysgen process is complex, even if you are just adding one transaction code to the system. AND, everything is quiesced during the commit phase of the online change process.

DRD is the solution

Dynamic Resource Definition

- Allow individual resource definitions to be:
 - Created
 - Deleted
 - Updated
 - without the need for MODBLKS OLC or IMS RESTART
 - Resources are:
 - Transactions
 - Routing codes
 - Programs
 - Databases
- Does not require resources NOT being changed to be quiesced

Notes:

The solution, simplified, is to allow the user to create, delete, or update resources in an online environment without the need for a sysgen and either an online change or an IMS restart, AND to quiesce only those resources being changed or being affected by the change.

Objective and requirements of DRD

- **OBJECTIVE:** *To improve the availability* of the IMS Online Environment by allowing the user to *dynamically* define and enable certain *resource definitions* without the requirement for an IMS Sysgen plus an IMS restart or an Online Change and the unavailability associated with either
- Resources include:

RESOURCE	SYSGEN MACRO	CONTROL BLOCK
Database	DATABASE (DB/TM, DBCTL)	DDIR
Application Program (PSB)	APPLCTN (DB/TM, DBCTL, DCCTL)	PDIR
Transaction	TRANSACT (DB/TM, DCCTL)	SMB
Routing Code	RTCODE (DB/TM, DCCTL)	RCTE

Notes:

The object, therefore, of Dynamic Resource Definition, is to improve the availability of IMS by allowing changes to the definitions of these resources without a sysgen or OLC. The table on this slide shows these resources and the control blocks representing them to the online system. DDIR is the Database Directory, PDIR is the Program Directory, SMB is the Scheduler Message Block, and RCTE is the Routing Code Table Entry (for Fast Path EMH routing codes).

Major components of DRD

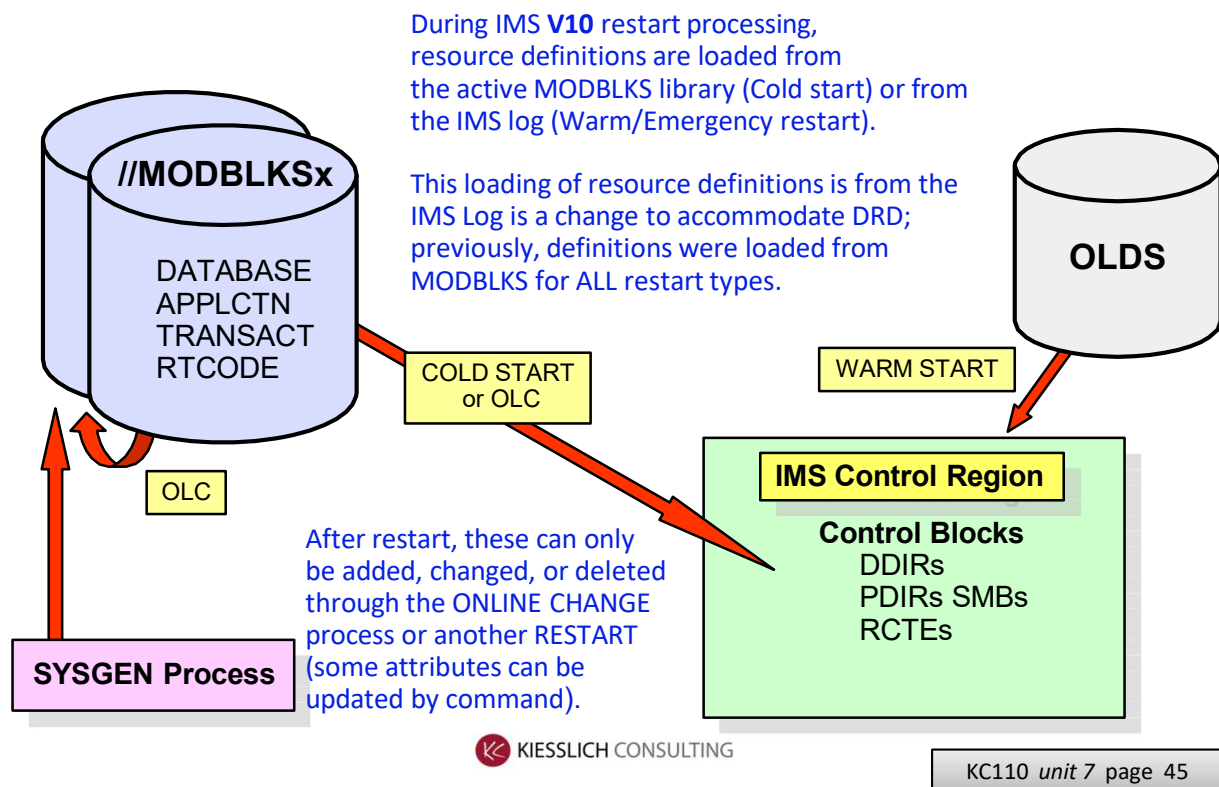
- New and enhanced Type-2 commands to dynamically CREATE, DELETE, and UPDATE IMS resource definitions traditionally found in MODBLKS library
 - [Databases, programs, transactions, routing codes](#)
- A set of *resource definition data sets* to contain both statically defined (Sysgen) and dynamically created (DRD) definitions
 - [Resource definitions](#)
 - [Model descriptors](#)
- New function to automatically IMPORT and EXPORT resource definitions from/to a resource definition data set
- An enhanced Type-2 command to QUERY the attributes and status of defined IMS resources and descriptors

Notes:

The major components of DRD include:

- One new (CREATE) and several commands (UPDATE, DELETE) which are not new but have new functionality.
- A set of BSAM data sets used to contain resource definitions from which IMS might import or to which IMS might export.
- Model descriptor definitions to make the task of assigning (setting) attribute values easier for the user.
- An automatic import function to enable IMS to determine the source of its definitions at Cold start.
- An automatic export function to enable IMS to save new and changed definitions to the RDDS so that the next Cold start can import them.
- Enhancements to the QRY command to show ALL resource and descriptor attributes and status.

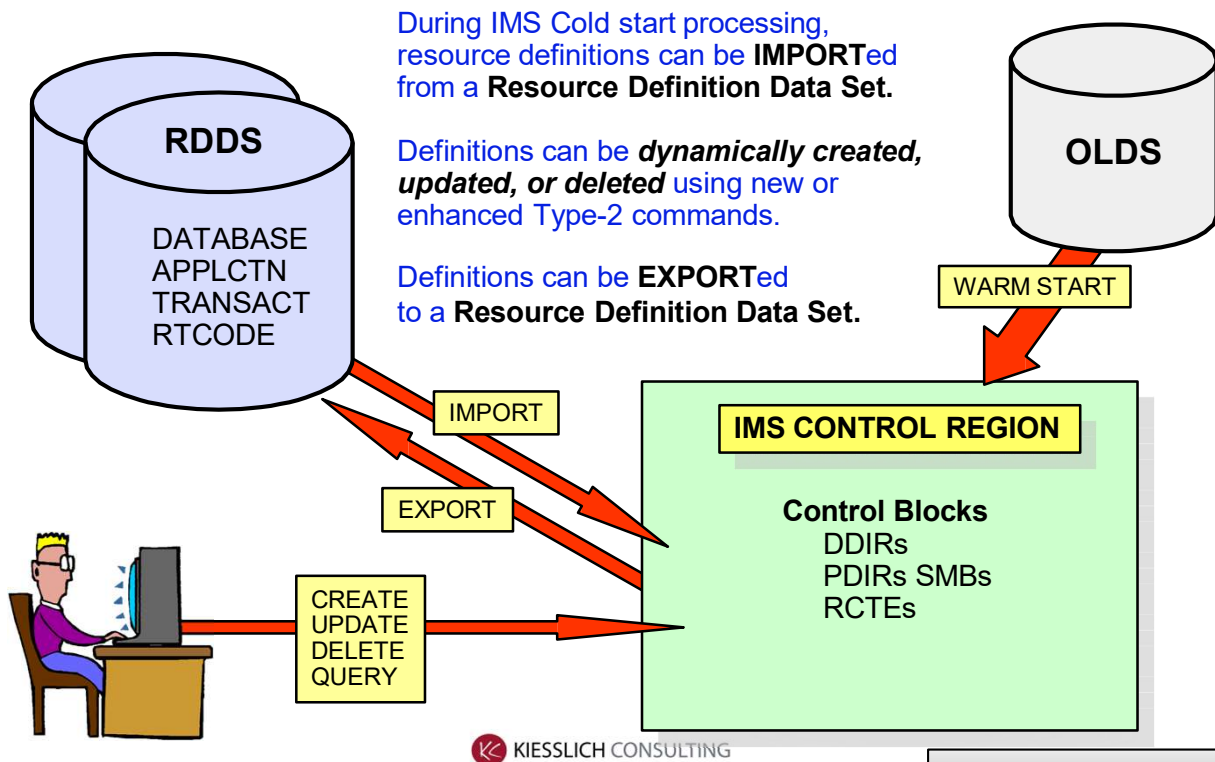
Modifying MODBLKS resources without DRD



Notes:

Without DRD, a MODBLKS online change, or a sysgen and a Cold start is required to update most IMS resources (some attributes could be updated using the /ASSIGN command).

Modifying MODBLKS resources with DRD



Notes:

This page shows the sysgen process being replaced by a user entering commands. The DRD commands processed are also logged to the IMS OLDS.

DRD is **supported ONLY with Type 2 commands**; as we saw earlier, the use of Type 2 commands require a CSL with at least an SCI and an OM address space. **CSL is REQUIRED** and there is **NO Type 1 Commands Support** for DRD changes.

IMS requires that multiple copies of the Resource Definition Data Set (RDDS). This should not surprise those familiar with IMS's design history of avoiding *single points of failure*.

The loading of resource definitions from the active MODBLKS at Cold start has been replaced by an (auto) IMPORT function from the Resource Definition Data Set (RDSS). The (auto) EXPORT function allows resources defined by the user to be dynamically saved in an RDDS for the next IMS Cold start.

PROCLIB members (1 of 2)

- DFSPBxxx:
 - CSLG=xxx
 - Suffix to DFSCGxxx – CSL PROCLIB member
 - DFSDF=xxx
 - Suffix to DFSDFxxx – IMS Definition PROCLIB member – new in V10
- DFSCGxxx:
 - CSL PROCLIB member
 - DRD requires a CSL environment with SCI and OM; RM not required
 - MODBLKS=DYN
 - This enables DRD
 - Alternate, default value (no DRD) is MODBLKS=OLC (Online Change)
 - This PROCLIB member can be replaced with CSL section of DFSDFxxx
 - But if both are coded, this one (DFSCGxxx) overrides conflicting values in DFSDFxxx

Notes:

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 (new). DFSCGxxx contains the new parameter MODBLKS=DYN (defaults to MODBLKS=OLC) which enables DRD. 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.

PROCLIB members (2 of 2)

- DFSDFxxx:
 - New *IMS Definition* PROCLIB member
 - DFSDF=xxx in DFSPBxxx
 - Divided into Sections:
 - <SECTION=COMMON_SERVICE_LAYER>
 - Must include MODBLKS=DYN for DRD
 - Might define CSL parameters (alternative to DFSCGxxx)
 - > If both exist, DFSCGxxx parameters override DFSDFxxx parameters
 - <SECTION=DYNAMIC_RESOURCES>
 - Define DRD parameters
 - Other sections for other parameters:
 - <SECTION=DIAGNOSTIC_STATISTICS>
 - <SECTION=SHARED_QUEUES>

Notes:

As mentioned earlier, DFSDFxxx is a new PROCLIB member required for DRD but which can also be used in place of DFSCGxxx and DFSSQxxx. The user can define four sections in this member. A complete description of DFSDFxxx is contained in System Enhancements.

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

- DRD is enabled by parameter in DFSCGxxx or CSL section of DFSDFxxx
 - MODBLKS=DYN
- Parameters in DFSDFxxx determine the automatic import and export functions of DRD

```
<SECTION=COMMON_SERVICES>
    MODBLKS=DYN

<SECTION=DYNAMIC_RESOURCES>
    RDDSDSN= (IMS10.RDDSDSN1,IMS10.RDDSDSN2,...)
    AUTOEXPORT=AUTO | RDDS | NO
    AUTOIMPORT=AUTO | RDDS | MODBLKS | NO
    IMPORTERR=ABORT | CONTINUE
    RDDSERR=ABORT | NOIMPORT
```

Notes:

There is a new PROCLIB member in V10 called the System Definition PROCLIB member. This PROCLIB member has several sections, including one that might replace the CSL PROCLIB member (DFSCGxxx) and one which is used to define the DRD environment.

The member itself is discussed in detail in the Systems Management section of the class. For now, we will talk about that section of DFSDFxxx which is used to define the import and export options, and the default for DCLWA when not otherwise specified.

- **CSL Section:** MODBLKS=DYN in DFSCGxxx or the CSL section of DFSDFxxx is used to enable DRD. To run without DRD, you would specify MODBLKS=OLC, indicating that changes to MODBLKS resources can only be made by online change.
- **DRD Section:** If DRD is enabled, then the DRD section defines how it is to work. It defines primarily the parameters related to automatic import and export.
 - 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 might 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 any read or allocation error.

Meaning of the parameters will be discussed in the next several slides.

DRD resources

- Resources have names, attributes, and status
 - Name
 - 1-8 characters – with all the usual restrictions
 - Attribute
 - A property of the resource specified during resource definition or update
 - Status
 - The availability or usability of the resource
- During Cold start:
 - When DRD not enabled, resource control blocks loaded from MODBLKS
 - When DRD enabled, control blocks loaded from RDDS, MODBLKS, or not at all
- During Warm or Emergency restart:
 - Control blocks always loaded from restart log
- During execution
 - Resource control blocks can be dynamically created, updated, or deleted

Notes:

Resources themselves have names, attributes, and status. There is no change to the naming conventions or requirements with DRD. An attribute is a property of a resource. For example, a transaction has an attribute (MAXRGN) defining how many regions that transaction can be scheduled into. Another attribute is whether that transaction is conversational. Resources also have statuses. There are several different statuses for each resource, but generally they indicate whether or not the resource is available for use. When DRD is not enabled, the control blocks are loaded from MODBLKS during IMS Cold start. When DRD is enabled, they might be loaded from an RDDS or from MODBLKS (or not at all).

During Warm or Emergency restart, control blocks are loaded from the IMS logs regardless of whether DRD is enabled or not. This was changed in IMS V10. Previously, all restarts loaded resource definitions from the active MODBLKS.

During execution, resources can be created, deleted, or updated online. These changes are logged in x'22' log records. At system checkpoint, including initialization and shut down checkpoints, all resources are logged and optionally written to the RDDS (depending on the options set for AUTOEXPORT in DFSDFxxx).

Resource attributes

- A property of a resource defined during Sysgen process or using DRD CREATE or UPDATE process:
 - Example from Sysgen
DATABASE RESIDENT,ACCESS=UP,DBD=ACCTMSTR
 - Example from DRD
**CREATE DB NAME(ACCTMSTR)
SET(RESIDENT(Y),ACCTYPE(UPD))**
 - RESIDENT and UPDATE access are attributes of the ACCTMSTR database
- Default attributes:
 - Value used when not explicitly specified in Sysgen or DRD process
 - For example, DATABASE default for ACCESS
 - EX (exclusive) during Sysgen
 - Found in **default descriptor** for resource type being defined
- Default descriptors:
 - Identify default attributes for DRD Create process
 - Can be user-defined (CREATE DBDESC) or IMS-supplied (DFSDSDB1)

Notes:

This slide talks about attributes or *properties* of a resources. An example is the ACCESS attribute for a DATABASE, which in DRD is referred to as ACCTYPE (access type).

RESIDENT is another database attribute.

The resource definition process also has default attributes which are used when the user does not completely define all attributes in the sysgen macro or the CREATE command. For example, if you do not include ACCESS in the DATABASE macro, it defaults to EX (exclusive). If you do not SET the ACCTYPE attribute in the CREATE command, it will default to whatever is in the system *default descriptor*. Note that the user can set the default attributes to any valid value.

Default descriptors are descriptors supplied by IMS or defined by the user which identify what the default attributes should be. There is one default descriptor for each of the four resource types. The IMS supplied descriptors have names with the pattern DFSDSxx1, where xx is DB, PG, TR, or RT. The user can define descriptors with any valid attributes.

DRD descriptors

- A model (template) for defining (creating) a resource or another descriptor
 - Establishes defaults for attributes not SET in CREATE command
- IMS-defined descriptors
 - Provided with the IMS product
- User-defined descriptors
 - Defined by the user
- Current system default descriptor
 - Each resource type will have one current default descriptor
 - IMS-defined or user-defined

Notes:

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 can 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.

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
 - Externalize resource and descriptor definitions
- Source for *importing* definitions
 - Load resource and descriptor definitions

IMS *direction* is to have a *Repository* to hold definitions which can be shared across all IMSs in an IMSPLEX..

Notes:

The Resource Definition Data Sets (RDDS) are a set of BSAM data sets used by IMS to export resource and descriptor definitions to, or import these definitions from. A minimum of two is required, but more might be defined. They are used in a round-robin (or flip flop) fashion with the most recent one never being the target of an EXPORT (we do not want to write over our last good set of definitions in case there is a write error).

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.

It is IMSs direction to develop a *Repository* for these definitions in some later release. One goal of the repository will be to be able to shared definitions among IMS clones.

Recoverability

- Resource and descriptor definitions ...
 - Exist for the life of IMS or until deleted
 - Are recovered across Warm and Emergency restart:
 - Definitions are logged when created, updated, or deleted, and at system checkpoint time
 - Definitions are restored from logs during restart processing
 - Are lost across Cold start unless:
 - Previously EXPORTed to RDDS < and then >
 - IMPORTed during Cold start
< or >
 - Sysgen-ed into MODBLKS library and IMPORTed during Cold start

Notes:

Resource and descriptor definitions are logged when they are created, updated, or deleted (x'22'), and logged at system checkpoint. IMS Warm start restores the checkpointed definitions. IMS Emergency restart uses the checkpointed definitions from the restart checkpoint, plus any updates to the end of the log.

If IMS is Cold-started, then, of course, it cannot use these logs. IMS must import the definitions either from an RDDS (most probable) or you might do a sysgen, then Cold start IMS and import from MODBLKS. To use the RDDS, it is important that all definitions be previously exported using the AUTOEXPORT option.

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	Create resource or descriptor definition
DELETE	DEL	Delete resource or descriptor definition
UPDATE	UPD	Update attributes of resource or descriptor definition Update status of resource
QUERY	QRY	Query attributes of resource or descriptor definition Query status of resource
**IMPORT	IMP	Import resource and descriptor definitions from RDDS
**EXPORT	EXP	Export resource and descriptor definitions from RDDS

****** Neither the *IMPORT* nor the *EXPORT* command are supported before V11 .
V12 is providing improvements here .

KC110 unit 7 page 55

Notes:

There are six commands used with DRD:

- **CREATE** - This command is used to create new resources or descriptors.
- **DELETE** - This command has been enhanced to support deleting or resources or descriptors.
- **UPDATE** - This command has been enhanced to support descriptors and additional attributes for resources.
- **QUERY**- This command has been enhanced to show all attributes for resources and descriptors, plus some additional *statuses* that are important to DRD. For example, a resource cannot be deleted if it's in use
- **IMPORT**- This command is used to import definitions from a Resource Definition Data Set (RDDS).
- **EXPORT**- This command is used to export definitions to a Resource Definition Data Set (RDDS).

IMPORT and *EXPORT* commands exist since IMS V11. It was the *direction* of IMS to be able to support a common repository across multiple IMS systems incorporated into later releases...

See REPOSITORY slide 58 and subfol.

UPDATE command

```
UPDATE      rsc-type | desc-type  
            NAME (names)  
            SET (attr1 (val1) , attr2 (val2) , ...)  
            DEFAULT (Y)  << descriptors only
```

- NAME:
 - Can specify multiple names
 - Can use wild cards (* and/or %)
 - Update will apply to all names
- SET
 - All attributes of all resources or descriptors can be SET
- DEFAULT(Y) – For descriptors only:
 - “Y” makes descriptor the current system default descriptor
 - Cannot specify DEFAULT(N)
 - Must specify “Y” on another CRE or UPD

Notes:

The UPDATE (UPD) command has been enhanced significantly in V10, both to support DRD and to support systems management in general. All of the attributes of resources and descriptors can be *shown* using the QRY command (prior to the DRD enhancements of IMS V10, only a limited number of attributes could be shown, and only for DBs and TRANS).

When an update command is entered with multiple names, each named resource or descriptor is processed individually. Some might succeed and some might fail. The IMS response will identify the reason for any failures with a condition code describing the reason it failed. If entered from the SPOC, the SPOC will identify the condition code (a number) and a description of what the CC means.

For the IMS-defined descriptors, the only valid update is to make the descriptor the system default. This would only be needed if you first made a user-defined descriptor the default and then later wanted to restore DFSDSxx1 and the default.

RDDS Extraction Utility

- RDDS Extraction Utility:
 - Creates Stage 1 macros or CREATE commands from definitions in RDDS:
 - Can be used to copy definitions to another system
 - Can be used to fall back to non-DRD use
 - Can 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, and so on)
 - CREATE commands
 - CREATE commands for DB, DBDESC, PGM, PGMDESC, TRAN, TRANDESC, RTC, and RTCDESC are written

Notes:

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 might be used fall back from the use of DRD to the use of MODBLKS. Finally, the creation of Stage 1 macros might 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 JCL

- 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
//DFSRRDDS 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

Notes:

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 DFSRRDDS 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 no default.

IMS REPOSITORY

- IMS Repository commands
 - IMS and RM IMSPLEX commands issued from SPOC
 - Batch interface commands
 - Repository Server commands issued through z/OS modify interface
- Comparison of DRD use with RDDS versus repository
- Using DRD with the IMS Repository in an online environment
- Managing the IMS Repository in an offline batch environment
- Migration to repository
- Security considerations
- DRD user interface enhancements
- IVP enhancements for repository
- Summary

The IMS Repository topic covers commands, usage, and repository management. There are several types of commands that will be discussed, some comparisons of using DRD with RDDS vs. the Repository, some repository usage considerations, followed by migration and security considerations. Some topics are related to the DRD user interface and the IVP ...

IMS Repository Function

- A 'repository' is a generalized data storage facility that can be used to store various types of information
- The IMS Repository function is a centralized method for storing and retrieving resource definitions in an IMSPLEX
 - Enables multiple IMS systems in a multiple-IMS IMSPLEX to manage, store, share, and retrieve resource definitions
 - Enables a single IMS system in a single-IMS IMSPLEX to manage, store, share, and retrieve resource definitions
- Focus is on improving the systems management and resource management aspects of handling IMS resource definitions
 - Across multiple IMSs or for a single standalone IMS
 - For test systems, for production systems

Any repository implementation provides facilities for storing and retrieving various types of information.

In IMS 12, the IMS Repository function was introduced to provides facilities for storing and retrieving IMS resource definitions that are used in an IMSPLEX. The IMSPLEX can contain multiple IMS systems or a single IMS system. The IMS Repository function will manage IMS resource definitions for both of these types of environments.

The 'IMS Repository function' provides the architecture for a common method for storing and retrieving IMS definitions for both test and production environments.

IMS Repository Function Usage

- The various components of the IMS Repository function provide a centralized storage and retrieval solution for resource definitions
- In IMS 12, the resource and descriptor definitions for Dynamic Resource Definition (DRD) can be stored in an IMS Repository
 - Contains resource definitions for programs/transactions/databases/FP routing codes & descriptors
 - Called the IMSRSC, the IMS resource definition repository
 - Provides an alternative to using RDDSs (resource definition data sets) for DRD
 - But both the Repository and RDDS can coexist
 - Can replace one or more sets of RDDSs in an IMSPLEX with a single repository
 - Eliminates the need to manually coordinate and manage separate RDDSs per IMS across a multiple-IMS IMSPLEX
 - Provides an alternative to using MODBLKs with SYSGEN and online change
 - Considered a strategic alternative to the RDDS
- IMS 12 can retrieve the stored resource definitions from the IMSRSC Repository to dynamically generate runtime resources for DRD

The types of IMS resource definitions that can be managed by the IMS Repository function are those definitions used by the DRD (Dynamic Resource Definition) function that provides for dynamic creation/maintenance of resource definitions for programs, transaction, databases and FP routing codes.

This specific implementation of the Repository function is called the IMSRSC, the IMS resource definition repository.

Previous to IMS 12, DRD implementation required the usage of RDDSs (resource definition data sets). Each IMS system has to have its own set of RDDSs, and coordination of these data sets in a multiple-IMS IMSPLEX environment had to be done manually.

The IMSRSC Repository since IMS 12 provides another alternative for storing DRD definitions, the IMSRSC Repository. It is a single common repository that can be accessed by all IMSs in a multiple-IMS IMSPLEX configuration; manual coordination is no longer required since this 'coordination' is now provided by the IMS Repository function itself.

An IMS user can now have their 'stored resource definitions' for DRD in either RDDSs or the Repository. IMS will then use these stored definitions to generate 'runtime resource definitions' for programs, transactions, databases, and FP routing codes.

Resource Definitions

- **DRD Resources in IMS's memory, i.e. those being used by the online IMS system, are referred to as -**
 - **Runtime Resources or Runtime Resource Definitions**
- **DRD Resources saved in a Repository or RDDS are referred to as -**
 - **Stored Resources or Stored Resource Definitions**
- **EXPORT command copies Runtime Resources to Stored Resources**
 - Can EXPORT to RDDS or Repository
 - Executed on one IMS, it can specify that resource definitions exported to the Repository are to apply to multiple named IMS systems
 - But doesn't change any runtime resource definitions
- **IMPORT command copies Stored Resources to Runtime Resources**
 - Executed on one IMS, it can specify that stored resource definitions from the Repository to become runtime resource definitions in multiple IMS systems

IMS Repository Function Components

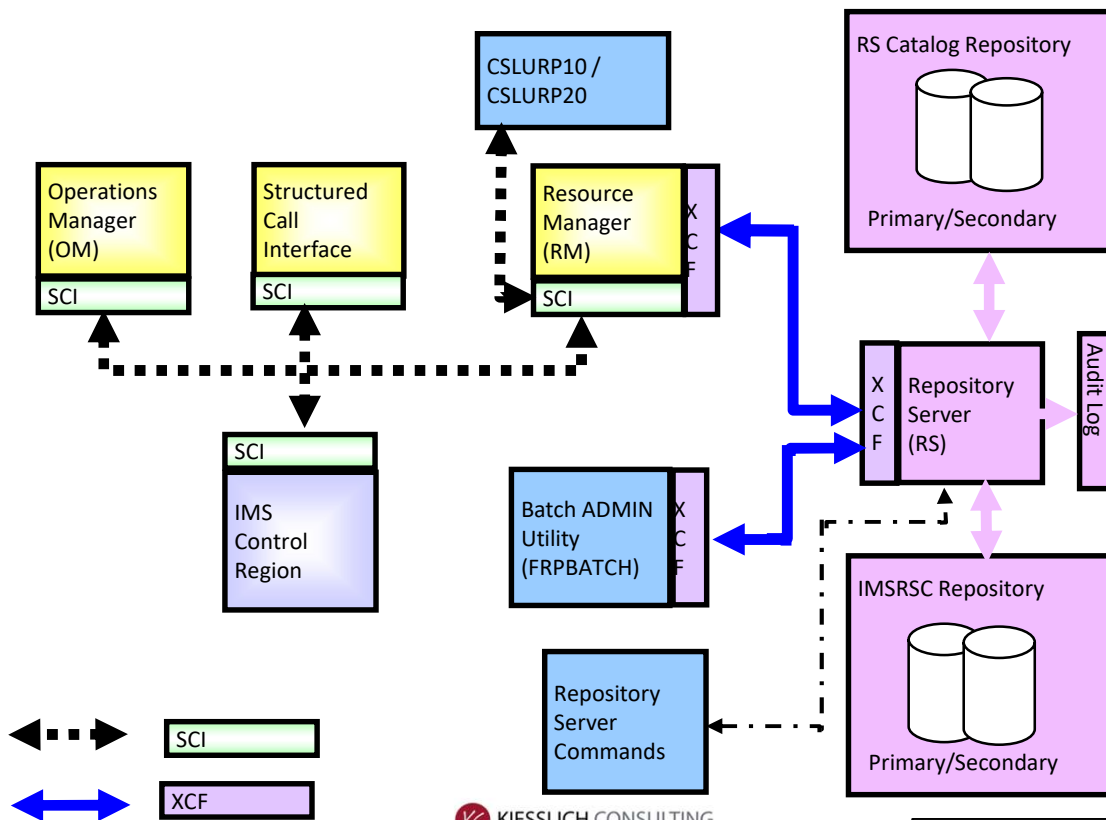
- A Repository Server (RS)
 - A new BPE-based address space managed by the Resource Manager (RM) CSL address space
- Repositories (data sets)
 - Catalog Repository
 - A catalog of the resource repositories
 - Used by the Repository Server
 - IMSRSC Repository(s)
 - Contains DRD stored resource definitions
- A Common Service Layer (CSL) IMSPLEX configuration consisting of
 - Operations Manager (OM)
 - Resource Manager (RM)
 - Structured Call Interface (SCI)
 - SPOC for entering type-2 commands
 - Optional Resource Structure with CQS address space
- Batch utilities
 - Batch ADMIN utility to manage the Catalog Repository
 - RDDS to / from Repository utilities

See graphics later

These are the 4 major components of the IMS Repository function.

- New: Repository Server address space referred to as the 'RS' address space.
- Repositories / DS on DASD
- (Common Service Layer) IMSPLEX configuration is needed now.
- Batch utilities to perform maintenance and migration/fallback functions.

IMS Repository Function Architecture



This diagram depicts the components of the IMS Repository Function.

The Repository Server, its repositories, and its audit log are in pink. CSL managers are in yellow. Batch and command interfaces are in medium blue.

SCI usage is shown with black arrows. XCF usage is shown with blue arrows.

IMS Repository Function Benefits

- Consolidation of resource definitions in a single place, the Repository
- DRD definitions are the initial implementation of the IMS Repository function (to optionally replace RDDSs)
- Full support for populating, managing, storing, sharing, and retrieving a consistent set of DRD stored resource definitions for multiple-IMS IMSPLEXes and single-IMS IMSPLEXes
 - Repository can be implemented without an outage
- Manual coordination of multiple RDDSs in a multiple-IMS IMSPLEX eliminated, replaced by basic functioning of the IMS Repository
- Improvements in IMSPLEX systems and resource management with the Repository
 - Via commands (3 types)
 - Via batch utilities (3 functions)
- A strategic direction for IMS architecture

The main benefit of the IMS Repository is that it consolidates IMS resource definitions in a single place that is IMS system-managed.

Environments using multiple-IMS IMSPLEXes gain significant benefits in the plex-wide coordination provided by the IMS Repository. Single-IMS IMSPLEX environments gain benefits from having a comprehensive technique that provides improved management capabilities.

DEMO ...

Help

PLEXA

IMS Manage Resources

Command ==> _____

Select an action and press Enter.

* Action _

1. Create new resources

2. Delete resources

3. Query resources

4. Update resources

5. Export resources

6. Import resources

7. Manage RDDS

F1=Help F12=Cancel

IMS Repository Function Components ...

Batch Utilities

Batch ADMIN utility (FRPBATCH)

Commands for managing content of the Catalog Repository , Functions such as

- ✓ ADD a new IMSRSC Repository,
- ✓ LIST the characteristics of an IMSRSC Repository,
- ✓ START or STOP an IMSRSC Repository

RDDS to / from repository utilities (Batch RM utilities)

- ✓ RDDS to Repository Utility (CSLURP10)
 - For migration
- ✓ Repository to RDDS Utility (CSLURP20)
 - For fallback

The last major component of the Repository function is the batch utilities that are provided.

A batch ADMIN utility (FRPBATCH) is provided to help manage the IMSRSC repositories that are part of a Repository Server. Some usage examples are to ADD a new IMSRSC Repository to a Repository Server, to LIST the characteristics of an IMSRSC Repository, and to START or STOP an IMSRSC Repository.

Two batch utilities (provided by the RM component) assist in migration to the Repository Server from RDDSs and fallback from the Repository Server to RDDSs. CSLURP10 provides facilities to move definitions from an RDDS to the Repository. CSLURP20 provides facilities to move definitions from the Repository to an RDDS.