

## ***IMS Repository – Part 2 Commands, Usage and Management***

Information Management software

Now that we understand the repository infrastructure and required setup, we now move to part 2 of the IMS repository session, which focuses on how to use the repository in your business operations.

## ***Topics***

- **IMS repository commands**
  - IMS and RM IMSplex commands issued from SPOC
  - Batch ADMIN 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**
- **Appendix**

# IMS Repository Commands

## IMS and RM IMSplex Commands Issued from SPOC

This section describes the new and enhanced commands in IMS 12 that apply to repository DRD. These commands are “IMSplex” commands, which are also known as type-2 commands, which can be issued to one or more IMS systems in an IMSplex through an Operations Manager (OM) interface. We will cover when it is appropriate to issue each command, describe how they are useful in different scenarios and show syntax for reference.

## New RM Commands

### ▪ UPDATE RM

- Dynamically enables/disables repository usage for the RM address space
  - Make CSLRIxxx changes first if enabling usage
- Dynamically changes the audit level setting and overrides the AUDIT\_DEFAULT originally set in FRPCFG member
  - Optionally audit security failures or member update, read and system read attempts

### ▪ QUERY RM

- Determines whether or not RM is enabled for repository usage
- Shows RM status of connection to repository
- Shows information about repositories being managed by RM such as audit access level

155

Use the new UPDATE RM command to dynamically enable the RM address space to use the repository, or change the audit access level that was originally specified in the FRPCFG member.

Notice that the command must be issued with TYPE(REPO) and REPOTYPE(IMSRSC) to indicate that a repository resource type and an “IMSRSC” type of repository is being updated. These are the only valid values for these parameters.

To dynamically enable RM for repository usage, specify SET(REPO(Y)) and command processing will reread and re-process the REPOSITORY section within the CSLRIxxx member. This is why it is important to add repository definitions to the CSLRIxxx member before issuing this command. Also during command processing, RM registers to the Repository Server (RS) if RM is not already registered. RM connects to the repository name specified in the REPOSITORY section of the CSLRIxxx member. If the command is successful at the command master RM, the command master RM communicates the changes to other active RMs in the IMSplex. All RMs in the IMSplex will have the same repository settings. If RM is defined to use the resource structure, the command master RM will update the resource structure with the repository name and repository type that it is connected to. Subsequent RMs that are restarted after the change will ensure that they are connected to the same repository name and repository type as read from the resource structure.

On the other hand, if RM repository usage is already enabled, you can dynamically disable it by specifying SET(REPO(N)). The CSLRIxxx member is not reread/re-processed in this case like it is when SET(REPO(Y)) is specified. Therefore, as part of the repository disabling process, you can remove the repository definitions from CSLRIxxx either before or after UPDATE RM is issued with SET(REPO(N)). **Note that if the repository definitions specified on the REPOSITORY= statement are still present in CSLRIxxx, any RMs that start after the UPDATE RM ... SET(REPO(N)) command is issued will re-connect to the repository during the RM startup.**

The AUDITACCESS() parameter allows you to dynamically change the audit level settings and override what was originally specified on the AUDIT\_DEFAULT parameter in the FRPCFG member.

## **UPDATE IMS Command Enhancements**

- Dynamically enables IMS to use the repository
  - RM must be repository-enabled
  - DFSDFxxx must contain valid repository definitions
- Dynamically disables automatic export
  - Take this step after migration to repository, otherwise system RDDS will continue to be updated at system checkpoint

Dynamically enable an IMS system to use the repository by issuing the UPDATE IMS command, or use it to disable the automatic export capability (covered on next slide). **Note that before you issue this command to dynamically enable IMS for repository usage, you must add repository definitions to the DFSDFxxx member and RM must already be repository-enabled.**

## **QUERY IMS Command Enhancements**

- Determines whether IMS is enabled for
  - Repository usage
  - Automatic export
- Displays repository attributes

To determine whether an IMS system is enabled for repository usage, issue the QUERY IMS command. You can also use this command to determine whether automatic export is enabled for an IMS system. These enhancements were added to existing QUERY IMS functionality – essentially, to display the information about the possible actions that were added with the UPDATE IMS command (dynamically enabling repository usage and/or automatic export).

## ***How Do I Know If Repository is Currently Enabled?***

- Issue QUERY RM command to determine if RM is enabled for repository usage
- Issue QUERY IMS command to determine if **IMS** is enabled for repository usage
  - Ad hoc or after an UPDATE RM or UPDATE IMS command is issued to enable repository usage

The RM address space and IMS need to separately be enabled to use the repository, which we will discuss later in the “Migration to Repository” section. However, since we just introduced the commands it is worth mentioning here that the QUERY RM and QUERY IMS commands can each be issued to determine whether RM and IMS are repository-enabled (respectively). For example, after an UPDATE RM or UPDATE IMS command is issued to enable repository usage – you can confirm that the repository is now enabled with the QUERY commands. But of course, if RM or IMS was not able to be enabled for repository usage for some reason, you would see this in a non-zero UPDATE command completion code.

Information besides whether repository usage is enabled is also displayed in the QUERY command output, as we just discussed.



## ***IMS Resource Lists***

- Each IMS can have its own unique set of resource definitions
  - Resources that the IMS is capable of processing are contained in an IMS resource list, which has all resource names and types defined in it
- Repository can contain one or more IMS resource lists (one for each IMS that is using the repository)
- Resources in an IMS resource list can have unique attribute values among different IMS systems
- Several commands can work with resource definitions that are specific to a particular IMS system
  - QUERY
  - EXPORT
  - DELETE DEFN
  - IMPORT

In understanding the IMS repository, an important concept to grasp is that of the IMS resource list. For each repository-enabled IMS, a IMS resource list exists for it within the repository. This list contains the resources that the IMS is capable of processing, specifically, the names and resource types. The resources that exist within each of these lists can have attribute values that differ from one another per IMS system. There are several IMS type-2 commands that work with the stored definitions associated with these resources, which we will now discuss...

## ***QUERY Command Enhancements***

- **Now displays:**
  - Stored IMS resource definitions in the repository
  - Runtime IMS-specific resource definitions
  - The IMS systems that have the specified resources defined

The QUERY command has been enhanced to display IMS resource/descriptor definitions (names and attribute values) in the repository and also at each IMS system. In addition, QUERY can now display specific IMSIDs that have the resource names included in the command. Note that this slide only shows the enhancement made to the QUERY command, not the full possible syntax for the command.

## ***EXPORT Command Enhancements***

- Hardens runtime resource adds/changes to repository for one or more IMS systems (active or inactive)
  - Can export resources changed within a specified timeframe, or only those have changed since last EXPORT, or resources that you specify explicitly
- Populates an empty repository for the first time, discussed later

161

Issue the EXPORT command when resources and/or descriptors have been either created or updated, and they need to be hardened to the repository. If hardening changes to offline stored definitions is part of your change management process, use the EXPORT command when definitional changes occur or at regular intervals during operations.

The EXPORT command is processed by a single command master IMS (the benefits of this are discussed in a few slides) and will write valid specified resources/descriptors to the repository.

The EXPORT command can also use data included in QUERY command output. If QUERY is issued with the OPTION(TIMESTAMP) parameter included, you can determine the exact time that a resource was created or updated (see the TimeCreate and TimeUpdate column header output). You are then able to use those timestamp values for the EXPORT STARTTIME() and/or ENDTIME() parameters. The STARTTIME() and ENDTIME() parameters can be as specific as tenths and hundredths of a second, and matches the timestamp granularity displayed in QUERY SHOW(TIMESTAMP) command output, so copying the exact values is facilitated in this way. Note that the STARTTIME() and ENDTIME() parameters are optional.

## ***EXPORT Command Considerations***

- All resource definitions are exported as a single unit of work
  - Export will fail if one resource is in error and no other resources will be exported
- Processed by one command master IMS
  - Writing to one shared repository is beneficial because EXPORT will succeed for all specified IMSs or none
  - Different than RDDS DRD, where EXPORT can be processed by multiple IMS systems, possibly succeeding on some and failing on others
- Certain resources/descriptors cannot be exported
  - System-defined descriptors
  - HALDB partitions

162

The EXPORT command writes all of the definitions to the repository as a single unit of work, as is the case with RDDS DRD. So in either case, the export will fail if one resource is in error, and no other resources will be exported. The difference between these two DRD types is that when exporting to the repository, only one command master IMS processes the command. This command master IMS creates/updates the definitions contained in each IMS resource list specified on SET(IMSID()). With RDDS DRD, each IMS that receives the command will process it. In this case, the command could succeed on some systems and fail on others.

In some cases, resources/descriptors cannot be exported -- such as when they are IMS system-defined descriptors, HALDB partitions, or invalid. Validation is done by RM before export occurs.

## **EXPORT Command Considerations**

- In a cloned environment, use EXPORT with SET(IMSID(\*)) so all IMS resources lists will be updated
  - IMS resource list(s) must exist in repository when using SET(IMSID(\*)) -- if first time exporting to repository, must specify individual IMSID(s)
- In a non-cloned environment, issue EXPORT with no SET(IMSID()) parameter at each IMS, so default IMSID of the IMS command master is taken
  - Route command to individual IMS systems without changing syntax
- For best performance, only export updated resources to minimize unnecessary resource access within repository
  - If updated resource/descriptor names are unknown, use:
    - OPTION(CHANGESONLY) to export only the resources/descriptors that have been created or modified since the last export
    - STARTTIME() to only export resources that have changed or created since a specific time

163

The EXPORT command should be used in different ways depending on whether you have a cloned or non-cloned IMS environment. Specifically, **in a cloned environment it is recommended to always specify the SET(IMSID())** parameter on the EXPORT command **with an “\*”** to ensure that all IMS resource lists are exported to. This will keep the definitions contained in each of the cloned IMS’s resource lists consistent with one another. Keep in mind that if the repository is empty, you must specify one or more individual IMSIDs on the EXPORT command to first create the IMS resource list(s). These IMS resource list(s) can later be updated with EXPORT ... SET(IMSID(\*)).

On the other hand, **in a non-cloned environment** the IMS systems may have differing attribute values for their resources and therefore **it is recommended to omit the SET(IMSID()) parameter** so that each IMS resource list can be updated one at a time, separately and the IMS-specific attribute values can be maintained. When omitting the SET(IMSID()) parameter, remember to route the command to the correct IMS. Since there is no IMS-specific information in the command syntax, you can easily re-issue this command without modifying the command itself.

For best performance, export only the resources and descriptors that have been changed since they were last hardened to the repository with an EXPORT command. You can more easily pinpoint the changed resource/descriptors by including OPTION(CHANGESONLY). Another method of minimizing the total number of resources/descriptors to be exporting is including the STARTTIME() parameter to target only those that have been created or updated after the specified time.

## ***New DELETE DEFN Command***

- Deletes stored resource definitions from the repository
- Use this command to harden runtime definition deletes to the repository
  - Runtime definitions are deleted with the DELETE command
  - Stored definitions are deleted with the DELETE DEFN command

Issue the DELETE DEFN command to delete stored resource definitions from one or more IMS resource lists contained in the repository. If runtime resource definitions have been deleted from an online IMS system with the DELETE command, these deletes can be hardened to the repository with the DELETE DEFN command. In this situation, take care to specify the same resources or descriptors in this command whose runtime definitions were deleted from the online system. Note that only one resource or descriptor type can be specified and therefore, the command may need to be issued multiple times if multiple resource/descriptor types were deleted from the online system.

## ***IMPORT Command Enhancements***

- Reads resource/descriptor stored resource definitions from the repository into IMS, where they become runtime resource definitions
- Use this command to percolate definitional changes made to the offline repository to 1+ running IMS systems, for example:
  - Coldstart an IMS with no resources defined, issue IMPORT to read in its definitions
  - Make changes to repository then roll them out to 1+ running IMS systems
- Can now replace an existing runtime resource definition with a stored resource definition that exists in the repository (or RDDS)
  - Work in progress cannot exist for the resource that IMPORT is attempting to update

165

The IMPORT command reads stored definitions that exist in the repository into running IMS systems. This command can be used if an IMS is coldstarted with no resources defined to populate the control region with runtime resource definitions. Or if changes were made to the repository offline and you'd like to roll the changes to the systems in IMSplex, the IMPORT command can be used to accomplish this. An example of when this scenario is when the "RDDS to Repository", or CSLURP10 utility (introduced in part one of this session) is used to populate a repository with definitions, which haven't been read into any IMS system yet.

Make sure that SOURCE(REPO) is specified, so the repository is the data set that is read, and indicate which resources should be imported using the other parameters shown on this slide.

### ***New Type-2 Command RACF (or SAF equivalent) Definitions for OM***

<b>Command</b>	<b>Resource Keyword</b>	<b>RACF Access Auth</b>	<b>Resource Name</b>
DELETE	DEFN	UPDATE	IMS.CSLplxname.DEL.DEFN
QUERY	RM	READ	IMS.CSLplxname.QRY.RM
UPDATE	RM	UPDATE	IMS.CSLplxname.UPD.RM

- Note: the IMSplex name must begin with the characters CSL
- Define in RACF OPERCMDS class

If you want to restrict access to the new type-2 Repository DRD commands, use the resource name format shown here when defining the resource profiles to RACF (or another SAF interface). If you use RACF, the OPERCMDS class is used to contain resource profiles representing the commands. Note the resource names associated with the new DELETE DEFN, QUERY RM, and UPDATE RM commands that would need to be added to the RACF OPERCMDS class to prevent unauthorized access. The required RACF permissions are also shown, and note that the IMSplex name must begin with the characters "CSL".



## Batch ADMIN Commands

Batch ADMIN commands are available to manage user repositories. The user repository data sets must be defined before the repository is started. The commands provide ability to add a repository, list repository details like status, data set names, start and stop repository for operations, rename or delete a repository, update repository-specific attributes or change the disposition of a repository's data sets. As we will discuss in this section, user repositories are defined to the Repository Server catalog repository (hereafter referred to as "RS catalog repository") using the batch ADMIN ADD command, and are started with the batch ADMIN START command.

Batch ADMIN commands focus on managing the individual user repository, whereas in the next section we will discuss z/OS modify interface commands that have similar function, but that are geared toward managing the Repository Server.

## ***Repository Server Batch ADMIN Utility***

- Invoked via JCL statements with FRPBATCH
- Enables repository administration
- Commands issued from utility

<b>Command</b>	<b>Function</b>
ADD	Add a user repository to the Repository Server catalog
UPDATE	Update user repository definition in the RS catalog repository
RENAME	Rename an existing user repository in the RS catalog repository
DELETE	Remove a user repository from the RS catalog repository
DSCHANGE	Change data set disposition
LIST	List status information for all user repositories or detailed information for a single user repository
START	Request the Repository Server to start a user repository
STOP	Request the Repository Server to stop a user repository

168

Batch ADMIN commands are available by invoking the FRPBATCH utility with JCL statements. This slide summarizes the batch ADMIN commands that are available for managing a user repository.

## ***ADD Command***

- Defines a newly allocated repository data set to the RS catalog repository
  - Use during initial repository setup after allocating repository data sets
  - Specifies primary, secondary, and (optional) spare repository data set names
- Determines whether or not the repository data sets will be opened automatically when the repository is started
- Defines security class to be used to restrict access to the repository (optional)

To define a user repository to the **RS** catalog repository, use the batch **ADMIN ADD** command using the syntax shown on this slide. Here, you must specify the user repository name as well as the names of the user repository primary/secondary index and member data sets. Note that the user repository name will be converted to upper-case if it is specified with anything else.

## ***UPDATE Command***

- Updates attributes of either the primary, secondary, or spare repository data set:
  - Auto-open feature (whether the repository data sets are opened when repository is started)
  - Security class being used to restrict repository access
- Repository must be stopped before it can be updated

Use the Batch ADMIN *UPDATE* command to modify a user repository definition within the RS catalog repository **datasets** (specifically, to change the data sets, auto-open option or security class associated with a specific repository). The only required parameter for this command is the REPOSITORY parameter. The parameters associated with this command have the same meaning as they do when issued with the batch ADMIN *ADD* command.

Note that a user repository must be stopped before it can be updated. We will cover how to stop a repository later in the session.

## ***RENAME Command***

- Changes the name of a repository

Use the batch `ADMIN RENAME` command to rename a user repository name defined within the RS catalog repository.

## ***DELETE Command***

- Removes a repository from the RS catalog repository
  - Physical data sets are not deleted (use the z/OS Access Method Services (IDCAMS) utility or a similar method)

Use the batch ADMIN DELETE command to remove a user repository from the RS catalog repository. Note that once you have deleted the user repository, you must delete its associated physical data sets in a separate step using the IDCAMS utility or similar method.

## ***DSCHANGE Command***

- Changes a repository data set's disposition to one of two possibilities:
  - DISCARD
  - SPARE
- Use this command after a primary or secondary repository data set fails as part of recovery process
  - If spare defined:
    - RS will drive recovery and replace the failed data set with the spare, leaving user with two RDSs: the primary and secondary (one of which the spare just replaced)
    - User must then allocate a new spare RDS and assign the 'SPARE' disposition to it
  - If no spare defined:
    - Failed RDS is automatically stopped and user must manually assign a 'DISCARD' disposition to it
    - User must then allocate a new RDS pair, then start the repository

173

There are certain times when it is appropriate to change the disposition of a repository data set (RDS) to either SPARE or DISCARD, which can be done using the batch ADMIN *DSCHANGE* command.

If an error occurs on the primary or secondary repository data set (RDS), recovery is driven by the Repository Server automatically if a spare RDS is present. Once this occurs, the user must then allocate and define a new RDS to replace the one that had the failure. This new RDS should be designated as the spare RDS, which can be done using the batch ADMIN *DSCHANGE* command with *ACTION(SPARE)* specified. This will change the disposition of a repository data set pair (RDS) to SPARE. **More detail about recovery in the event of an RDS error will be covered later in the session.**

If you want to replace an existing RDS with a different RDS, you must first stop the repository and change the disposition/status of the RDS to DISCARD. This can be done by issuing the batch ADMIN *DSCHANGE* command with *ACTION(DISCARD)* specified. Once an RDS has a disposition of DISCARD, it can be replaced with a newly defined data set.

## ***LIST Command***

- Display details associated with a single repository or all repositories that are defined to the RS catalog repository:
  - Repository name
  - Repository status
  - Date of last update
  - USERID that last updated user repository

Use the batch ADMIN *LIST* command to display the details of a single repository, including its status, or display all user repository names. The information that will be shown if a user repository is specified for the REPOSITORY parameter is listed on this slide. Note that you can also issue the command as just *LIST STATUS* (without specifying the REPOSITORY parameter) to see only a list of user repository names defined to the RS catalog repository.



## ***START Command***

- Use this command to start a repository after defining it to the RS catalog repository with *ADD* command

Use the batch *ADMIN START* command to start a specific user repository, for example after it has been defined to the RS catalog repository with the batch *ADMIN ADD* command. Note that with the optional *OPEN()* parameter, you can override the *AUTOOPEN=* parameter value that was originally specified when the repository was added to or last updated in the RS catalog repository (with batch *ADMIN ADD* or *UPDATE* commands, respectively). This parameter value indicates whether the user repository's RDSs will be open when it is started (with *OPEN(YES)*), or when a user first connects to it (with *OPEN(NO)*). **Note that you can only override the *AUTOOPEN=* parameter if it was originally specified as *AUTOOPEN=NO*.**

You can optionally include the *MAXWAIT* parameter to indicate how many seconds should elapse before a particular action that you also specify is taken. You can specify a wait time of up to 9999 seconds and opt to have the command continue processing with a return code of either 0 or 4, or opt to have it terminate with a return code of 8. By default, if 5 seconds has elapsed once this command has been issued, the command will continue processing and will give a return code of 4. The specific parameter values for the *MAXWAIT()* syntax are shown on this slide.

## ***STOP Command***

- Use this command in preparation for updating user repository definitions within a RS catalog repository with *UPDATE* command
  - A stopped repository rejects connection attempts and is deallocated/closed by the Repository Server

Use the batch *ADMIN STOP* command to stop a specific user repository that is defined to the RS catalog repository. A stopped repository will reject user connection attempts. The command also results in the repository being closed and deallocated by the Repository Server. Note that this command has the same *MAXWAIT()* parameter value as the batch *ADMIN START* command.

Much like a */DBR* command that prevents programs and transactions from accessing a database, the batch *ADMIN STOP* command, when issued with *MAXWAIT(xx,IGNORE or CONTINUE)*, can continue processing after *xx* seconds have elapsed. At this point, the command continues processing (just like the */DBR* command) and a specific return code is received, determined by whether *IGNORE (rc=0)* or *CONTINUE (rc=4)* was specified. Of course, if *ABORT* was specified instead of *IGNORE* or *CONTINUE*, the command would terminate processing and a *rc=8* would be received when *xx* seconds has elapsed.

## z/OS Modify Interface Commands

Repository Server parameters are able to be displayed and dynamically updated via z/OS modify interface commands.

## Repository Server z/OS Modify Interface Commands

- Enables repository administration
- Commands issued from z/OS console
- See Appendix for command syntax and more detailed description

Command	Function
ADMIN	Administrative functions – change repository data set disposition, display repository data set attributes, start/stop repositories
AUDIT	Dynamically change audit level originally specified in FRPCFG member
SECURITY	Refresh RACF profile definitions
SHUTDOWN	Shutdown one or more repository server address spaces
STOP	Stop/shutdown specific repository server

This slide summarizes the different z/OS modify interface commands that available to administer the user repositories from a Repository Server perspective.

**Batch ADMIN and z/OS Modify Interface Commands**

Batch ADMIN	Repository Server z/OS Modify Interface
ADD	
LIST	ADMIN DISPLAY
START	ADMIN START
STOP	ADMIN STOP (repository, not Repository Server)
RENAME	
DELETE	
DSCHANGE	ADMIN DSCHANGE
UPDATE	
	AUDIT (change audit level)
	SECURITY (refresh in-storage profiles)
	SHUTDOWN
	STOP (stops Repository Server)

179

This slide displays all batch ADMIN commands and Repository Server commands and if applicable, their equivalents.

## Comparison of DRD Use with the RDDS Versus the Repository

Let's now compare RDDS DRD with repository DRD and examine their similarities and differences.

## Deleting Resources – RDDS Versus Repository DRD

### ▪ RDDS DRD deleting

- To delete a resource, issue a DELETE command at each IMS system that contains the runtime resource definition
  - Example: DELETE TRAN NAME(TRANA)
- Automatic export will occur at system checkpoint and remove the deleted resource from the system RDDS
- EXPORT with OPTION(OVERWRITE) can be issued at an IMS that deletes were performed on, to remove the deleted resource from the system/non-system RDDS
  - Example: EXPORT DEFN TARGET(RDDS) ... OPTION(OVERWRITE)

### ▪ Repository DRD deleting

- To delete a resource, issue a DELETE command at each IMS system that contains the runtime resource definition (same as RDDS DRD)
- Issue a DELETE DEFN command to remove the deleted resource from repository, specifying FOR(IMSID()) to indicate which IMS systems that the deletes were performed on
- EXPORT command cannot harden deleted resources to repository

181

To delete a resource with RDDS DRD, the runtime definition is deleted with a DELETE command. This deletion is hardened to the IMS's system RDDS at system checkpoint via automatic export (if it is enabled) or with an EXPORT command including OPTION(OVERWRITE). The resource is then removed from the system RDDS's stored resource definitions. With repository DRD, the runtime resource definition is also deleted with the DELETE command, just like with RDDS DRD. However, since automatic export is not supported with repository DRD and the EXPORT command cannot be used to harden deletions to the repository, a different step needs to be taken to accomplish this. The DELETE DEFN command must be used to remove stored resource definitions from the IMS resource list associated with the runtime resource deletions that occurred in the running system.

## ***Importing Resources – RDDS Versus Repository DRD***

- **RDDS DRD importing**
  - IMPORT DEFN SOURCE(RDDS) is processed at each IMS it is routed to, and each IMS reads in definitions from the specified RDDS data set
- **Repository DRD importing**
  - IMPORT DEFN SOURCE(REPO) is processed at each IMS it is routed to, and each system reads in stored definitions from its IMS resource list within the shared repository
  - SCOPE(ALL) new parameter for IMPORT only applies to repository DRD
    - Recommendation: if specifying routing -- route to all IMSs in IMSplex when including this parameter value (route will supersede SCOPE(ALL))
- **Both RDDS DRD and repository DRD can create new and update existing runtime resource definitions using an IMPORT command...**

182

When you issue an IMPORT command in an RDDS DRD environment, each IMS system that receives the command will read the stored resource definitions from the specified RDDS into the control region where they become runtime resource definitions.

With repository DRD, each IMS that receives the IMPORT command will read the stored resources definitions from its respective IMS resource list contained in the repository into the control region. Also with repository DRD you can use the SCOPE() parameter to specify whether the IMPORT command should be applied to only active IMS systems, or to both active and inactive IMS systems. This was covered earlier in the session when the IMPORT command enhancements were being discussed.



## ***Updating Resources with IMPORT Command***

- New enhancement in IMS 12
- Applies to both RDDS DRD and repository DRD
- New OPTION(UPDATE) parameter value
  - Allows existing runtime resource definitions to be updated via IMPORT command
  - If resource does not exist, it will be created as a runtime resource definition in the IMS system

As mentioned previously, the IMPORT command can now be used to refresh runtime resource definitions in an IMS system with stored resource definitions contained in an RDDS or repository. This action is taken when the IMPORT command is issued with OPTION(UPDATE) included. If the resource doesn't exist as a runtime definition, the stored resource definition will be created in the running system.

## **IMPORT Command with UPDATE Option**

- Imports resource and descriptor definitions from an RDDS or repository
- UPDATE applies to all definitions affected by IMPORT command

Existing Runtime Definition Exists?	OPTION(UPDATE) Specified?	IMPORT DEFN Result
No	No	Runtime definition created
No	Yes	Runtime definition created
Yes	No	IMPORT DEFN fails
Yes	Yes	Runtime definition updated

- Resource definitions can be created or updated with the IMPORT command using OPTION(UPDATE)
- Now for an example scenario...

184

The IMPORT command imports resource and descriptor definitions from an RDDS or user repository. If the UPDATE option is specified it applies to all of the resources/descriptors being imported.

The table shown on this slide summarizes the following scenarios:

If the imported definition is for a resource or descriptor that is unknown to IMS, IMS creates the runtime definition for the resource, whether or not OPTION(UPDATE) is specified.

If the imported definition is for a resource or descriptor for which IMS already has a runtime definition (the resource or descriptor already exists in IMS) and OPTION(UPDATE) is not specified, the definition is not imported and the command fails.

If the imported definition is for a resource or descriptor for which IMS already has a runtime definition and OPTION(UPDATE) is specified, the existing runtime definition is updated with the attributes from the imported definition.

## Updating Resources with IMPORT Command

- Example scenario
  - An IMS application program exists on a test IMS (IMST) and on a development IMS (IMSD) that are in the same IMSplex
  - Changes are made to this application program, requiring new/changed resource definitions on both IMS systems
  - Testing required on test IMS system before definitions are ported to development IMS
  - On test IMS system, IMST:
    - Dynamically add new resources with DRD CREATE
    - Dynamically update existing resources with DRD UPDATE
    - EXPORT these new/updated runtime resource definitions to update IMST's IMS resource list and stored resource definitions in repository
      - EXPORT DEFN TARGET(REPO) NAME(rsc-names) SET(IMSID(IMST)), route command to IMST
  - Complete successful testing on IMST
  - Port definitions to development IMS, IMSD:
    - EXPORT IMST's updated runtime resource definitions to update IMSD's IMS resource list and stored resource definitions in repository
      - EXPORT DEFN TARGET(REPO) NAME(rsc-names) SET(IMSID(IMSD)), route command to IMST
    - Update IMSD's runtime resource definitions
      - IMPORT DEFN SOURCE(REPO) NAME(rsc-names) OPTION(UPDATE), route command to IMSD
    - New runtime resource definitions are added to development IMS
    - Existing runtime resource definitions are updated in development IMS

185

Let's take a look at how the IMPORT command can be useful when issued with the UPDATE option specified. The example scenario outlined on this slide illustrates the flexibility of the command in that it can create or update runtime resource definitions depending on whether a runtime definition exists for a resource being imported or not. In the example, we assume that a repository is being used instead of an RDDS.

We begin by dynamically creating new and updating existing runtime resource definitions on IMST with CREATE and UPDATE commands. We then capture these definitions and harden them to the repository with an EXPORT command.

Next, all testing occurs on IMST and when successful, we are ready to port the definitions to IMSD.

We first update IMSD's repository information by issuing an EXPORT command and routing it to IMST. Remember, the resource changes were made on IMST so we must route the command to that IMS system to capture the changes, but have it reflected in IMSD's repository information. Specifically, IMSD's IMS resource list is updated to include the new definitions and its stored definitions are updated with the updated resource definitions, which all occurred in the runtime environment for IMST.

Now we are ready to import these new/changed resources to IMSD runtime environment, so we issue an IMPORT command with OPTION(UPDATE). The resources are read into IMSD where they become runtime resource definitions, and are created if they don't yet exist or are updated if they do.

As you can see, the EXPORT command hardens runtime resource definitional changes to the repository. This can be useful in ensuring that a user repository is not empty, so that when an IMS coldstarts, it will have definitions to read in via automatic import.

## **Considerations for *IMPORT* with *OPTION(UPDATE)***

- When updating an existing runtime resource definition, the resource cannot be in use or the *IMPORT* will fail
  - Example: access to database must be stopped before database definition is updated
    - UPD DB NAME(xxx) STOP(ACCESS)
    - /DBR DB xxx
    - This applies to changing the resident value and to changing the access type
- Recommendations
  - Review stored resource definitions to determine if *IMPORT* would result in updating (versus creating) runtime definition
    - RDDS Extraction Utility (DFSURDD0) can be used to display definitions stored in RDDS
    - *QUERY* with *SHOW(DEFN)* can be used to display definitions stored in repository
  - Ensure runtime resources to be updated are not in use
    - Use the *SHOW(WORK)* filter on the *QUERY* command to determine if the resources to be updated are in use

186

When using the *IMPORT* command with the *UPDATE* option, a runtime definition is created if one does not already exist and an existing runtime definition is updated with the new attributes.

When updating an existing runtime resource definition, the resource cannot be in use or the *IMPORT* will fail. If a database definition is to be updated by the *IMPORT* command, access to the database must be stopped before the import is done. This applies to changing the resident value and to changing the access type. When updating the resident value with the *UPDATE DB* command, you must stop access to the database before you issue the command. You do not, however, have to stop access to the database when using the *UPDATE DB* command to update the access type. With the *IMPORT* command you must always stop access to the database before a database definition is to be updated, even if the import is just updating the access type.

## **Considerations for *IMPORT* with *OPTION(UPDATE)***

- **Performance**
  - When the *IMPORT* command is specified with the *UPDATE* option, existing resources affected by the update are quiesced
    - e.g. Updating a tran quiesces tran and associated program
    - Resource cannot be scheduled
    - Resource cannot be updated or deleted
  - Certain latches are held which will prevent:
    - Resources from being scheduled
    - System checkpoint
- **Recommendation to minimize performance impact**
  - If a large number of resource definitions are to be updated with the *IMPORT* command, avoid issuing the command during peak processing periods

When the *IMPORT* command is specified with the *UPDATE* option, all existing resources affected by the update are quiesced. For example, if the import is updating a transaction definition, the transaction and the associated program are quiesced.

While quiesced, work cannot be run against the resource. The resource cannot be scheduled. The resource cannot be updated or deleted.

Certain latches are held during the import process that prevent work from being done. No resources can be scheduled while the resources to be updated are being quiesced. A system checkpoint is not allowed while an import is in progress.

## ***Exporting Resources – RDDS Versus Repository DRD***

### ▪ **RDDS DRD exporting**

- Handles resource additions, changes and deletions
- Automatic export can occur at system checkpoint, which overwrites entire contents of oldest system RDDS with the IMS's definitions
- EXPORT DEFN TARGET(RDDS) command can overwrite an entire system/non-system RDDS or append to it
- Each IMS has its own system RDDS that contains its entire set of definitions
  - When IMS clones exist together in an IMSplex, EXPORT command must be routed to each of the IMSs
    - Command could fail on some IMSs and succeed on others
    - Manual coordination is required to keep system RDDSs associated with different IMS clones consistent with same contents
- EXPORT only applies to active IMS systems
  - No way of applying DRD activity occurring in IMSplex to an inactive IMS

188

When exporting in an RDDS DRD environment, resources that have been newly created, changed, or deleted can all be hardened to the RDDS via automatic export if it is enabled. They can also be hardened to the RDDS by issuing an EXPORT command, which can overwrite the entire contents of an RDDS (EXPORT can also append added and changed resources/descriptors to the RDDS).

With RDDS DRD, each IMS system has its own dedicated pair of system RDDSs that contain the system's entire collection of MODBLKS definitions. When an EXPORT command is issued in a cloned environment, it should be routed to all of the IMS systems in the IMSplex so that their system RDDSs remain synchronized with the same set of definitions. In this case, each IMS system that receives the command will process it and perform export. There is potential here for one IMS system to fail the EXPORT command, whereas others could succeed. This would require the user to manually re-synchronize the system RDDSs by correcting the error and re-issuing the EXPORT on the IMS that failed. Manual coordination is not always straightforward and is not recommended since it requires more effort on the part of the user.

In addition, RDDS DRD exporting only applies to active IMS systems and there is no way for an inactive IMS's stored resource definitions to be updated.

## ***Exporting Resources – RDDS Versus Repository DRD***

- **Repository DRD exporting**
  - Handles only resource additions and changes (no deletions)
  - Automatic export not supported
    - To achieve same effect within repository, issue:
      - EXPORT to repository with CHANGESONLY option or STARTTIME/ENDTIME, routing command to targeted IMS -- omitting SET(IMSID()) parameter
      - DELETE DEFN with NAME() specifying runtime resources that have been deleted with DELETE command and FOR(IMSID()) specifying which IMS resource lists should remove these resources
  - Each IMS has its own IMS resource list within one shared repository that contains its entire set of definitions

189

Repository DRD exporting only allows added and changed resources/descriptors to be written to the repository. As previously mentioned, automatic export is not supported with this type of DRD and in order to export to a specific IMS resource list within the repository, an EXPORT command is required. To export the IMS's runtime resources definitions that have been added or changed since the previous EXPORT command was issued, include the OPTION(CHANGESONLY) parameter omitting the SET(IMSID()) parameter so only the targeted IMS's resource list is updated with the changes. If any runtime resource definitions were deleted from the IMS, issue a DELETE DEFN command with the correct FOR(IMSID()) parameter to remove the associated stored resource definitions from the IMS's resource list in the repository.

With repository DRD, instead of each IMS having its own set of system RDDSs, each IMS has its own IMS resource list that contains its entire collection of MODBLKS definitions within the shared repository.

## Exporting Resources – RDDS Versus Repository DRD

- **Repository DRD exporting**
  - EXPORT DEFN TARGET(REPO) command updates IMS resource lists (additions) and stored definitions (changes) within the repository by specifying SET(IMSID()) parameter
    - Processed by one command master IMS as a single unit of work
    - Definitions will be written only if there were no errors updating the IMS resource list(s) (all or nothing written)
      - When clones exist together in an IMSplex, the possibility of the export succeeding on some IMSs and failing on others is eliminated, as coordination is automatic
  - Can export definitions to an IMS that is inactive – resource changes will be applied when the IMS restarts
    - Issue EXPORT with SET(IMSID()) specified, including the IMSID of the inactive IMS
    - The inactive IMS's stored resources definitions within the repository will be updated with the definitional changes (additions and changes only)
      - If there have been deletions that should be reflected in the inactive IMS system, issue DELETE DEFN specifying the name on FOR(IMSID())

190

When an EXPORT command is issued in a repository DRD environment, the SET(IMSID()) parameter determines which IMS resource list in the repository will be updated with new or changed definitions. Unlike RDDS DRD where each IMS that receives the EXPORT command will process it, a repository DRD EXPORT will only be processed by one command master IMS. Since the command is processed as a single unit of work, either all of the specified IMS resource lists are updated by the export or none of them are if an error occurs. The possibility of some IMS's stored resource definitions being different than the others is eliminated since only one IMS is processing the command as a single unit of work, writing to the shared repository. RDDS DRD export also processes the command as a single unit of work, but the difference is that multiple IMSs are each processing the command separately, updating different RDDSs, which as previously stated can succeed or fail at the different systems resulting in desynchronization.

Repository DRD export can update the stored definitions of an IMS that is inactive, unlike RDDS DRD export. These updated stored definitions can be applied when the IMS restarts, but remember, export does not handle resource deletions – only additions/changes. To deleted stored resource definitions from the repository, a DELETE DEFN command is required.



## Exporting Resources – RDDS Versus Repository DRD

- **Repository DRD exporting**
  - RM performs resource validation for transactions/routing codes being added to repository to ensure that an associated program exists (or that the program is also being added via EXPORT)
    - EXPORT fails if associated program is not present
  - RM performs resource attribute validation for programs, transactions, and routing codes being added to the repository to ensure they do not conflict with associated resources' attributes
    - Example: a transaction updated to FP(E) will cause RM to check that the associated program is defined as FP exclusive
    - If conflict, EXPORT fails
    - This and additional functionality provided by APAR PM32805 (see APAR for more detail)

191

When you export a newly created transaction or routing code to the repository using EXPORT, RM will validate that there is a program associated with the transaction or routing code in the repository. RM checks the EXPORT command to determine if the associated program is being exported as well. If the program is not present in the repository or is not included in the EXPORT command, EXPORT fails. This validation is only performed with repository DRD using RM. It is not needed with RDDS DRD, because IMS performs this validation when a transaction or routing code is CREATED. If the associated program does not exist at this time, the CREATE fails. Therefore, a system RDDS will always contain transactions/routing codes that have associated programs since any lack of a program is caught at CREATE time, and automatic export with RDDS hardens all definitions the system RDDS. When repository DRD is enabled, this same checking occurs when a transaction or routing code is created, but because you have the ability to export individual resources to the repository – this extra layer of checking is required to ensure that the program is present.

RM also checks the attribute values of resources being added to the repository via EXPORT, to ensure that they do not conflict with attributes associated with other resources. An example of conflicting attribute values is shown here. Note that this functionality was added to IMS 12 via APAR PM32805.

## ***Exporting Resources – RDDS Versus Repository DRD***

- New parameters only applicable to repository DRD
  - STARTTIME() and ENDTIME()
  - OPTION(CHANGESONLY)
- Parameters applicable to RDDS DRD
  - OPTION(APPEND)
  - OPTION(OVERWRITE)

This slide shows the EXPORT parameter values that are specific to RDDS DRD and those that are specific to repository DRD.

## Using DRD with the IMS Repository in an Online Environment

Let's now explore a few repository DRD usage scenarios that occur in an online environment.

## ***IMPORT Command Usage with Repository DRD***

- **IMPORT DEFN SOURCE(REPO)** reads stored resource definitions into active IMS systems in an IMSplex, where they become runtime resource definitions in the control region
- Changes can be made offline to the repository and then rolled out to active IMS systems, for example:
  - CSLURP10 can populate a repository with definitions from an RDDS
  - IMPORT command can read in these new definitions to active IMS systems
- **RDDS DRD** does not have ability to make changes to stored resource definitions offline and roll them out to active systems

In IMS 12, you have the ability to import stored resource definitions from the repository at active IMS systems in an IMSplex by issuing the IMPORT command. One distinct difference between repository DRD and RDDS DRD is that repository DRD allows a user to make offline changes to the stored definitions for an IMS, that can later be imported at an active IMS using the IMPORT command. RDDS DRD does not allow a user to make offline changes to the stored definitions within an RDDS.

## ***Indoubt Work in Progress After IMPORT/EXPORT***

- If IMS, RM or the Repository Server address space terminates while an IMPORT/EXPORT command is processing, work in progress can have “indoubt” status
- Point of failure will determine whether IMPORT/EXPORT must be re-issued
  - If termination occurred before UOW was completed, changes will be backed out
  - If termination occurred during UOW commit, issue QUERY with SHOW(DEFN, TIMESTAMP) to determine whether IMPORT/EXPORT completed
- Check the following in the QUERY command output to confirm whether work in progress was committed before failure occurred
  - Definitional attribute values
  - Timestamps shown in:
    - TimeCreate
    - TimeUpdate
    - TimeImport

195

It is possible that IMS, RM or the Repository Server can unexpectedly terminate during IMPORT or EXPORT command processing. In this case, it will be uncertain if the unit of work (UOW) command processing actually completed, thereby being “indoubt”.

The point of failure is important in determining whether the command needs to be re-issued. Use the QUERY SHOW(DEFN, TIMESTAMP) command to display the definitional attribute values of the resources/descriptors involved in the IMPORT or EXPORT, as well as when they were last created, updated or imported. Then compare the timestamps shown in either the TimeCreate, TimeUpdate or TimeImport columns to the point of failure. This should indicate whether the IMPORT or EXPORT command should be issued again.

## ***Indoubt Work in Progress After IMPORT/EXPORT***

### ▪ Example 1

- EXPORT DEFN TARGET(REPO) TYPE(TRAN) NAME(TRANA,TRANB)
- IMS terminates during command processing
- Work in progress of EXPORT is indoubt
- QUERY TRAN NAME(TRANA,TRANB) SHOW(DEFN,TIMESTAMP)
- Check stored resource definition attribute values and TimeCreate or TimeUpdate column in command response data (IMS command master displays repository stored resource definitions)

### ▪ Example 2

- IMPORT DEFN SOURCE (REPO) TYPE(TRAN) NAME(TRANA,TRANB)
- IMS terminates during command processing
- Work in progress of IMPORT is indoubt
- QUERY TRAN NAME(TRANA,TRANB) SHOW(DEFN,TIMESTAMP)
- Check runtime definition attribute values and TimeImport column in command response data associated with each IMS

- If QUERY command indicates that work in progress was not committed, reissue IMPORT/EXPORT command

196

This slide shows two example scenarios which involve EXPORT and IMPORT command processing that is “indoubt” and the steps that would be taken in each case to resolve it.

## ***Resources with Unique Attribute Values in Repository***

- Some resources require unique attribute values across IMS systems
  - SIDR and SIDL values for remote transactions in MSC environment
  - Transaction class
- To export resources that have unique values, issue EXPORT DEFN TARGET(REPO) command, doing one of the following:
  - Specify the target IMS system using the SET(IMSID()) parameter
  - Omit the SET(IMSID()) parameter to default to the command master as the selected IMSID (important: ROUTE command to appropriate IMS system via OM API)
- The stored resource definitions within the repository will be updated with these unique attribute values for the specified IMS's resource list

In order to modify the SIDR and SIDL values for remote transactions and transaction descriptors, the EXPORT command with the specific IMSID specified on SET(IMSID) or the default SET(IMSID()) must be issued routing the command to the IMS whose definitions are to be exported.

## ***Resources with Unique Attribute Values in Repository***

- Note: EXPORT with SET(IMSID(\*)) will not export SIDR/SIDL values to each IMS resource list
- All other attribute values will be exported and each IMS resource list will be updated with these other values
- To modify SIDR and SIDL values for remote transaction resources and descriptors
  - Issue EXPORT command with the single IMSID on SET(IMSID())
    - If omitting SET(IMSID()), route command to correct IMS
- How SIDR/SIDL values are stored in repository
  - Local transaction resource and descriptor SIDR/SIDL values are saved as 0 in repository generic section
    - Import will set SIDR/SIDL values to lowest local SID value of the IMS system where the runtime definition is imported
  - Remote transaction resources and descriptor SIDR/SIDL values of each IMS are saved as unique in the respective IMS's specific section

198

When EXPORT is issued with SET(IMSID(\*)), all other attributes besides SIDR/SIDL will be collapsed in the repository, meaning that no IMS resource list will have its own unique values for these attributes.

Once again, in order to modify the SIDR and SIDL values for remote transactions and transaction descriptors, issue the EXPORT command with the specific IMSID specified on SET(IMSID). If SET(IMSID()) is omitted, the command must be routed to the IMS whose SIDR and SIDL definitional values are to be exported.

For local transactions, SIDR and SIDL values are the same but for remote transactions they are different. This is reflected in the transactions' stored definitions in the repository in the following way. For local transactions and transaction descriptors, the SIDR and SIDL values are saved as 0 in the repository in the generic section. When the stored resource definition is imported from the repository either during AUTOIMPORT processing or during processing of the IMPORT command, the SIDR and SIDL values are set to the lowest local SID value of the IMS system where the runtime resource definition is imported. In the case of remote transactions and transaction descriptors, the SIDR and SIDL values are maintained for each individual system's IMS specific section.

Stored definitions within the repository's generic section apply to all IMS systems, whereas definitions in an IMS specific section in the repository only apply to that specific IMS system, and is different from the other IMS's definitions.



## ***Exporting Resources Created with DFSINSX0 to Repository***

- DFSINSX0 user exit used to dynamically create a transaction resource to process a message sent to IMS with an unknown destination
- In order to export these dynamically created resources to repository
  - Set TRNQ\_FC\_EXPORT=1 on exit input parameter list
  - Issue an EXPORT DEFN TARGET(REPO) command including either
    - NAME() parameter specifying names of dynamically created transactions
    - STARTTIME() and ENDTIME() parameters that encompass timeframe the exit dynamically created the transaction resources in
      - Use this if resource names are unknown

The Destination Creation (DFSINSX0) user exit can be used to dynamically create transaction resources that will process messages that come into IMS with an unknown destination. To export these dynamically created transactions to the repository, set the TRNQ\_FC\_EXPORT as shown in this slide and issue the EXPORT command with either the NAME() or STARTTIME()/ENDTIME() parameters.

Keep in mind that when using DRD with the repository, no automatic export will occur at system checkpoint, as is the case with RDDS DRD. So to harden transactions that are dynamically created with the DFSINSX0 exit to the repository, they must be exported with an EXPORT command.

## Managing the IMS Repository in an Offline Batch Environment

200

Let's now look at some scenarios in which the repository can be managed in an offline environment, namely, for security updates and recovery procedures.

## Offline Repository Management - Examples

- Updating security settings to protect repository from unauthorized access
  - Changing RACF security class name associated with a user repository in RS catalog repository with batch `ADMIN UPDATE` command
    - `UPDATE REPOSITORY(REPO1) SECURITYCLASS(XFACILIT)`
  - Changing RACF definitions
    - `RDEFINE XFACILIT FRPREP.REPO1 UACC(NONE)`
    - `PERMIT FRPREP.REPO1 CLASS(XFACILIT) ID(ANGIE) ACCESS(READ)`
  - Refreshing RACF in-storage profiles with z/OS modify interface `SECURITY` command
    - `F REPOSVR1,SECURITY REFRESH`

201

The above example illustrates the use of both batch admin and z/OS modify interface commands being used to specify security settings. The SAF class name the REPO1 repository will use is XFACILIT, which is then defined to RACF to prevent unauthorized user access. A userid named ANGIE is then permitted to read it within the RACF definitions. Finally, the in-storage RACF profiles are refreshed to reflect these updates. Note: the z/OS modify interface SECURITY command must be used any time the RACF definitions are updated.

## ***Offline Repository Management - Recovery***

- If a write error occurs on a primary or secondary data set pair within repository, its disposition status is changed to DISCARD
- Repository Server manages recovery process if SPARE data set present
- User's only task is to allocate/define a new spare data set and assign it to SPARE disposition with DSCHANGE command

A repository data set pair (hereafter referred to as the RDS) that is identified by the server as having lost integrity is discarded. If an RDS is discarded due to a write error, then the repository will be stopped at this time to enable recovery. In this event, the Repository Server will drive recovery automatically if a spare RDS is available. If no spare RDS is available, the user repository is stopped and administrator intervention is required to restart the user repository.

## Offline Repository Management - Recovery

- Example: a repository named REPO1 contains a primary (COPY1), secondary (COPY2) and spare (SPARE) data set
- Write error occurs on the primary and Repository Server drives recovery
  - Primary data set is automatically changed to disposition of DISCARD
  - Secondary data set's definitions are copied to spare
  - Repository Server changes spare to primary
  - Issue command to determine which RDS has been discarded (with batch ADMIN LIST or z/OS modify interface ADMIN,DISPLAY command)
  - Delete and define the discarded primary data sets to replace old spare (best practice: new spare should be larger than previous)
  - Change disposition of this new data set to SPARE
    - Batch ADMIN command: DSCHANGE REPOSITORY(REPO1) RDS((1) ACTION(SPARE))
    - z/OS modify interface command: F REPOSVR1,ADMIN DSCHANGE (REPO1,S,1)

203

This slide shows an example scenario in which we begin with three RDSs that each have different dispositions/statuses of COPY1 (primary RDS), COPY2 (secondary RDS), and SPARE (spare RDS). A write error then occurs on the primary RDS. The Repository Server will set change disposition of the primary RDS from COPY1 to DISCARD and copy the contents of the secondary RDS to the spare RDS automatically (if a spare is available). At this point, the existing spare in our example shown becomes the new primary data set that replaces the repository data set that failed.

The batch ADMIN LIST command or z/OS modify interface command F xx,ADMIN DISPLAY() command can be issued to show the dispositions, or statuses, of each repository data set. You must then delete the bad repository data set whose disposition is now DISCARD. Allocate and define a new repository data set (ideally, the size should be larger than the previous spare) and assign a disposition of SPARE to this new data set using either the batch ADMIN or z/OS modify interface commands shown here. Notice that in each command, a "1" is specified. In our example scenario, the primary data set failed and so a "1" representing the old primary data set is specified to set its disposition to SPARE.

## Migration to Repository

204

We'll now discuss the steps required to migrate to using repository DRD. The scenarios presented include migration when:

- IMS is active
- IMS is inactive
- RDDS DRD is currently enabled
- MODBLKS online change is currently enabled

We'll also discuss various ways of populating the repository for the first time, as well as fallback to RDDS DRD and MODBLKS online change.

## ***Migration/Coexistence***

- **IMS 12 RM is required for the repository**
  - IMS 10 and IMS 11 RM cannot support the repository
- **Apply the following maintenance to IMS 10 or IMS 11 RM before implementation of the repository**
  - IMS 10: PM19025
  - IMS 11: PM19026
  - With these APARs IMS 10 or IMS 11 RM will abend U0010-0300 if a repository is enabled when they are initialized
- **Refer to *IMS 12 Release Planning* manual, GC19-3019**

IMS 10 and IMS 11 RMs cannot support the repository. APARs PM19025 for IMS 10 and PM19026 for IMS 11 cause IMS 10 and IMS 11 RMs to check for an enabled repository when they are initialized. If the repository is enabled, the RM will abend with U0010 and subcode X'0300'. If the repository is not enabled, the IMS 10 or IMS 11 RM will initialize normally.

## Migration from RDDS DRD to Repository DRD

- Assumptions
  - Physical data sets that will store the repository contents have been created
  - Repository Server has been defined in the FRPCFG member (including the RS catalog repository primary/secondary data sets)
  - Repository Server address space has been initialized
- Issue a batch `ADMIN ADD` command to define a user repository to the RS catalog repository
- Issue a batch `ADMIN START` command to request the Repository Server to start the user repository and make it available for use
- Edit `CSLRlxxx` to define repository-specific information in `<SECTION=REPOSITORY>`
- Edit `DFSDfxxx` to define valid repository definitions in `<SECTION=REPOSITORY>`
- Next steps vary, depending on whether IMS being migrated is active or inactive...

206

The next few slides cover the steps needed to migrate to using repository DRD from RDDS DRD. **Note that each part of the setup covered here only needs to be done once, during initial migration.** Note the assumptions that list steps that should have already been completed before proceeding with the next migration steps shown on this slide.

Configure the Repository Server configuration parameter member, ensuring that the primary and secondary RS catalog repository index and member data sets are included. Define `PRIMARY_CATALOG_REPOSITORY_INDEX=` and `PRIMARY_CATALOG_REPOSITORY_MEMBER=` for the primary RS catalog repository data set and `SECONDARY_CATALOG_REPOSITORY_INDEX=` and `SECONDARY_CATALOG_REPOSITORY_MEMBER=` for the secondary RS catalog repository data set within the FRPCFG member.

Next, define the user repository to the RS catalog repository using the batch `ADMIN ADD` command and start the user repository with the batch `ADMIN START` command. Then add the `REPOSITORY` section to both the RM initialization (`CSLRlxxx`) and system definition (`DFSDfxxx`) members, specifying the appropriate repository definitions.

The next steps assume that the RM address space is already running.



## ***Migration from RDDS DRD to Repository DRD***

- **If IMS is active (and RM is active)**
  - Issue the following command to refresh RM address space and dynamically enable repository usage
    - UPDATE RM TYPE(REPO) REPOTYPE(IMSRSC) SET(REPO(Y))
  - Issue the following command to enable IMS to use the repository
    - UPDATE IMS SET(LCLPARM(REPO(Y) REPOTYPE(IMSRSC))
  - Populate the repository for the first time by issuing the following command:
    - EXPORT DEFN TARGET(REPO) TYPE(ALL) NAME(\*)
      - Route command to the IMS being migrated, since SET(IMSID()) will default to that IMS

207

If IMS is active, dynamically enable both the RM address space and IMS to use the repository with the commands shown. The user repository should currently be empty at this point since the previous steps just defined it to the RS catalog repository for the first time and started it. To capture all of the IMS's definitions and populate the user repository for the first time with them, issue an EXPORT command as shown, specifying that all resources should be exported to the repository. This ensures that the next time this IMS coldstarts, it will have definitions to read in from the repository, assuming that AUTOIMPORT=AUTO or AUTOIMPORT=REPO is set in the DFSDFXxx member. **Note the importance of routing the command to the specific IMS being migrated.** Either omit the SET(IMSID()) parameter as shown, and ROUTE it to the IMS using the OM API routing capability, or take care to specify the correct IMSID on the SET(IMSID()) parameter if it must be included in the command.

## Migration from RDDS DRD to Repository DRD

- If IMS is inactive (and RM is active)
  - Issue the following command to refresh RM address space and dynamically enable repository usage
    - UPDATE RM TYPE(REPO) REPOTYPE(IMSRSC) SET(REPO(Y))
  - Populate the repository for the first time by running the RDDS to Repository utility (CSLURP10), using the most current system RDDS
  - Restart the IMS system using either:
    - Coldstart (ensure that AUTOIMPORT=AUTO or AUTOIMPORT=REPO in DFSDFXxx so that definitions are read from repository)
      - Alternative: coldstart with no resources defined (AUTOIMPORT=NO) and IMPORT definitions after coldstart complete
    - Warmstart or emergency restart (must have already completed migration to IMS 12)

208

If IMS is inactive, dynamically enable the RM address space to use the repository with the command shown. The technique used in populating the repository for the initial time is different than what was shown on the previous slide when IMS is active. In this case, populate the repository offline using the CSLURP10 (RDDS to Repository) utility. **Note that it is recommended to use the IMS system's most recently updated system RDDS**, which can be determined by running the DRD "Extract RDDS Contents" utility against each of the IMS's system RDDSs and viewing the timestamp in the output that reflects when the RDDS was last updated.

Once the repository has been populated with the IMS's definitions by way of the CSLURP10 utility, coldstart the IMS. If automatic import has been enabled, IMS will read in the definitions contained in the repository. If automatic import is disabled, an IMPORT command can be issued once coldstart has completed, to read the definitions into the control region. A coldstart is not required however – a warmstart or emergency restart can be used as well as long as IMS V12 has already been migrated to.

## ***After Migration from RDDS DRD to Repository DRD***

- **Remove RDDS DRD elements**
  - Issue the following command to reduce unnecessary I/O associated with automatic export at system checkpoints
    - UPDATE IMS SET(LCLPARM(AUTOEXPORT(N)))
    - Otherwise, automatic export will continue to export to the system RDDS at system checkpoint despite repository being enabled
  - Remove RDDS definitions from DFSDFxxx to cleanup for next IMS coldstart
  - If AUTOIMPORT=RDDS, update to AUTOIMPORT=AUTO
    - AUTOIMPORT=REPO also valid
- **EXPORT/IMPORT command syntax that will be issued to reference REPO instead of RDDS (may need to update automation)**
- **Note: no coldstart required to migrate to repository DRD!**

209

Once RM and IMS have been enabled to use the repository and sufficient testing has been completed, the RDDS DRD components are no longer needed. If automatic export is enabled, disable it with the command shown. Otherwise, IMS will export definitions to the oldest system RDDS at system checkpoint if there have been definitional changes made since the previous checkpoint.

Modify the DFSDFxxx member to delete the RDDS-related definitions such as the AUTOEXPORT=, RDDSERR= and RDDSDSN=() parameters and change the AUTOIMPORT= value to the recommended setting AUTOIMPORT=AUTO (if not already done).

**Note that any automation may need to be updated with the new EXPORT DEFN TARGET(REPO) and IMPORT DEFN SOURCE(REPO) command syntax to replace EXPORT DEFN TARGET(RDDS) and IMPORT DEFN SOURCE(RDDS), respectively.**

**Also note that migration to repository DRD does not require a coldstart. All of the migration steps can be completed without having to take an outage.**

## ***Migration from MODBLKS OLC to Repository DRD***

- Follow same steps as migration from RDDS to repository
  - DFSDFxxx will require additional editing since enabling DRD for first time (see *IMS V12 System Definition* manual)
- Create a temporary RDDS (used only for migration purposes) for use with RDDS to Repository utility CSLURP10
  - Generate this RDDS with definitions using one of the following
    - Create RDDS from MODBLKS utility (DFSURCM0)
    - Create RDDS from Log Records utility (DFSURCL0)
    - Issue an EXPORT with TYPE(ALL) and NAME(\*) command to populate a non-system RDDS

210

If RDDS DRD was never enabled, migration to repository DRD is still possible, with some extra steps. For example, the Common Service Layer (CSL) must be enabled and the DFSDFxxx member will require some additional editing to specify that DRD will now be used to manage MODBLKS resources instead of online change. The documentation shown here contains instructions on both of these required steps. Migration from MODBLKS online change to DRD requires an IMS coldstart.

Once the CSL has been implemented and DRD has been specified to manage the MODBLKS resources instead of online change, the steps for migration to repository DRD (when an IMS is inactive) are the same as for RDDS DRD, with the exception of one item. A temporary RDDS for the sole purpose of populating the repository is required. DRD utilities are available to simplify this step, as shown on the slide. Once the temporary RDDS has been generated using the utility, the steps are the same as those specified for enabling an IMS for repository usage when it is inactive.

## **Merging Multiple RDDS Data Sets in Repository**

- **Consolidating multiple RDDS data sets' contents into the repository**
  - For each RDDS, run RM utility CSLURP10 using the desired RDDS name as input
  - One RDDS can be specified each time utility is run
  - Definitions contained in the input RDDS will be copied to the repository
  - IMS resource lists and stored definitions within repository are updated based on the IMSIDs specified on CSLURP10's SYSIN DD statement:  
IMSPLEX(NAME=plexname IMSID(imsid))
    - RM performs resource validation for transactions/routing codes being added to repository to ensure that an associated program exists (or that the program is also being added via CSLURP10)
    - RM performs resource attribute validation for programs, transactions, and routing codes being added to the repository to ensure they do not conflict with associated resources' attributes
      - Example: a transaction updated to FP(E) will cause RM to check that the associated program is defined as FP exclusive
      - If associated program is not present or if attribute values conflict, CSLURP10 fails with RC=8 and CSL2616E issued

211

The RDDS to Repository utility (hereafter referred to as CSLURP10) can be run against one or more IMS systems to copy RDDS contents to the repository, thereby merging multiple RDDS contents together, divided into respective IMS resource lists.

When the CSLURP10 utility tries to write a transaction or routing code to the repository, RM will validate that there is a program associated with the transaction or routing code in the repository. RM checks whether CSLURP10 is attempting to write the associated program is being exported as well. If the program is not present in the repository or is not included in the CSLURP10 utility, the utility fails. RM also checks the attribute values of resources being added to the repository via CSLURP10, to ensure that they do not conflict with attributes associated with other resources. An example of conflicting attribute values is shown here.

### ***Creating IMS Definitions in Repository from a Single IMS (Non-Cloned Environment)***

- **Offline:** select an RDDS that reflects the most recent snapshot of an IMS system's definitions and use it as input to the CSLURP10 utility
  - RDDS selection
    - Can use latest system RDDS, updated at last system checkpoint
    - Can EXPORT all definitions to an RDDS
  - Run CSLURP10 utility
    - Specify IMSPLEX(NAME=plexname IMSID(*imsid*)) to designate which specific IMS resource list will be updated in repository
- **Online:** export definitions to repository for a single IMS using EXPORT DEFN TARGET(REPO) NAME(\*) TYPE(ALL) command
  - ROUTE command to the IMS whose definitions should be exported so it is selected as command master
  - IMS resource list of the command master will be updated within repository
- Above steps can be repeated for each IMS that has its own unique set of definitions that needs to be written to the repository

212

To create definitions in a repository in a non-cloned environment offline, run the CSLURP10 utility, specifying an RDDS and a single IMSID on the IMSPLEX statement shown in this slide. This will copy the contents of the RDDS specified in the utility to the IMS's resource list repository.

An online alternative is issuing the EXPORT command with TYPE(ALL) and NAME(\*) specified, omitting the SET(IMSID()) parameter and routing the command to the IMS whose definitions should be exported to the repository.

Repeat these steps for each IMS whose stored definitions in the RDDS should be copied to the repository.

## ***Creating IMS Definitions in Repository from a Single IMS -- Cloned Environment***

- **Offline:** Run CSLURP10 utility and specify IMSPLEX(NAME=plexname IMSID(**imslist**)) to designate which IMS resource lists will be updated in repository
  - Select RDDS associated with a single IMS that reflects most recent snapshot of the system's definitions as input to CSLURP10
- **Online:** export definitions to repository for multiple IMSs using EXPORT DEFN TARGET(REPO) NAME(\*) TYPE(ALL) SET(IMSID(**imslist**))
  - Specify IMSIDs in SET(IMSID(**imslist**))
  - An IMS resource list containing the definitions is created in the repository for each IMS listed in the **imslist**

213

In a cloned environment, run the CSLURP10 utility, specifying the input RDDS and the appropriate IMSIDs on the IMSPLEX statement shown in this slide. This will copy the contents of the specified RDDS to each of the IMS's resource lists belonging to the respective IMSIDs in the repository.

An online alternative is issuing the EXPORT command with TYPE(ALL) and NAME(\*) specified, specifying all of the IMSIDs whose resource lists should be created in the repository using the SET(IMSID()) parameter.

## ***Fallback from Repository to RDDS DRD***

- Shutdown the IMS system that is falling back
- Generate an RDDS matching the contents of the repository by running “Repository to RDDS utility” (CSLURP20), using repository as input
  - If repository is not available:
    - Run the “Create RDDS from Log Records” utility
    - Run the “Create RDDS from MODBLKS” utility (if MODBLKS available)
    - EXPORT to RDDS to capture existing IMS definitions, which can be read in at next coldstart
- Edit DFSDFxxx to remove <SECTION=REPOSITORY> definitions
- Coldstart IMS, using automatic import to read stored resource definitions from RDDS into control region
  - AUTOIMPORT=AUTO or AUTOIMPORT=RDDS in DFSDFxxx <SECTION=DYNAMIC\_RESOURCES> portion
- Edit CSLRIxxx to remove <SECTION=REPOSITORY> definitions
- Issue UPDATE RM ... SET(REPO(N)) command to disable repository usage

214

In order to fall back to using RDDS DRD from repository DRD, follow the steps recommended on this slide. While you are able to dynamically enable IMS to use the repository with the UPDATE IMS command, you cannot dynamically disable repository usage with the command and instead must shutdown the IMS system as a first step.

The next step is re-generating a system RDDS that the IMS that is falling back will use for automatic import when it coldstarts next. In order to generate an RDDS, use the CSLURP20 utility to read the repository to copy the IMS's definitions from it to the RDDS. The JCL that executes the utility includes a statement that allows you to specify the IMSID of the system that is falling back (specifically, it is IMSPLEX(NAME=plexname IMSID(imsid)). The CSLURP20 utility will copy the resource definitions associated with the IMSID that you specify for this parameter from the repository to the RDDS.

If you do not have a repository available to use with the CSLURP20 utility, use the DRD utilities to create the RDDS from other data you have in your shop such as log records or MODBLKS contents. You could also allocate a non-system RDDS data set and EXPORT the IMS's definitions to it before shutting it down.

Once you've generated an RDDS, ensure that AUTOIMPORT=AUTO (AUTOIMPORT=RDDS can also be used) is defined in the DFSDFxxx member and that RDDSDSN() (also in DFSDFxxx) is set to the RDDS name that you generated in the previous steps. Note that other system RDDSs defined for the RDDSDSN() parameter should either be empty or have an earlier timestamp reflecting when they were last updated. This will ensure that the RDDS generated in the previous steps will be the one that is read by automatic import when the IMS coldstarts.

Assuming that only one IMS in the IMSplex has been enabled for repository usage and therefore will be the only one falling back, disable repository usage for the RM address space by removing the repository definitions from the CSLRIxxx member and issuing the UPDATE RM command with SET(REPO(N)) specified. **Note:** the UPDATE RM command processing is the mechanism that disables RM from repository usage. This is different than issuing UPDATE RM to enable repository usage, where the re-reading/re-processing of the REPOSITORY section in CSLRIxxx is the mechanism enabling repository usage (the command can not enable it on its own without reading the member). For the disabling scenario, you can remove the repository definitions in CSLRIxxx after the UPDATE RM command is issued. However, keep in mind that any other RM address spaces that start after the UPDATE RM command is issued with SET(REPO(N)) before you have had a chance to remove the repository definitions from CSLRIxxx will re-connect to the repository.



## ***Fallback from Repository to RDDS DRD***

- Once previous steps have been completed for all repository-enabled IMSs and RMs, **shutdown** Repository Server address spaces using either:
  - P *reposeservername* (single Repository Server)
  - F *reposeservername,SHUTDOWN ALL* (multiple Repository Servers)
- Delete repository data sets
- Delete RS catalog repository data sets

After all of the repository-related components have been removed and disabled, shut down all Repository Server address spaces and delete the user repository and RS catalog repository data sets.

## ***Fallback from Repository to MODBLKS Global OLC***

- **Follow same steps as fallback to RDDS DRD**
  - Disabling DRD will require additional steps (see *IMS V12 System Definition* manual)
- **Generate a MODBLKS data set**
  - Run CSLURP20 to copy repository contents to a temporary RDDS (or use steps on previous slides if repository not available)
  - Use this RDDS as input to DRD utility (DFSURDD0) to generate Stage 1 macros
  - Use Stage 1 macros to generate MODBLKS

To fall back from repository to global online change, follow the same steps just described on the previous slides that described fallback to RDDS DRD using a temporary RDDS. Since Dynamic Resource Definition is a prerequisite to enabling repository usage, it too has a fallback process which is documented in the *IMS V12 System Definition* manual in a section entitled “Disabling dynamic resource definition”.

In order to generate a MODBLKS data set, you can create a temporary RDDS that can be used as input to the DRD utilities to generate Stage 1 macro statements, which can then be used to generate a MODBLKS.

## **Uses for CSLURP10 and CSLURP20 RM Utilities**

- **CSLURP10**
  - Migration
    - Generate a repository with contents equivalent to a specified RDDS
- **CSLURP20**
  - Fallback
  - During migration
    - Maintain an RDDS containing equivalent repository definitions for backup purposes in case fallback to RDDS is required
  - Generate a non-system RDDS whose definitions can be IMPORTed by an IMS or which can be used as input into “Extract RDDS Contents” utility
- **Use utilities together**
  - Migrate definitions in one repository to another repository
    - Use CSLURP20 to generate an RDDS
    - Use this newly generated RDDS as input to CSLURP10 to populate the other repository

217

As we've seen, the CSLURP10 utility is useful during migration to the repository and the CSLURP20 utility is useful for fallback or for backup during migration. But the utilities can also be used in other ways – for example, together to copy resource definitions from one repository to another repository: the CSLURP20 can generate a non-system RDDS, which can then be used as input to the CSLURP10 utility to generate another repository, thereby copying the contents of the original repository to it. The definitions contained in the non-system RDDS generated by CSLURP20 can also be imported by an IMS system with an IMPORT command. Finally, this same non-system RDDS could be used as input to the Extract RDDS Contents utility (DFSURDD0), which could in turn generate equivalent Stage 1 macro statements, type-2 CREATE commands or a query report displaying resource information.

The CSLURP10 and CSLURP20 RM utilities provide offline, batch access to the repository without requiring IMS to be active, and therefore they can be useful in operations and resource management.

# Security Considerations

We'll now discuss how to secure access to Repository Server resources.

## Repository Access

- Access to a user repository can be gained through RM by either of the following types of RM callers
  - Authorized RM caller
    - **IMS** via commands such as:
      - EXPORT TARGET(REPO)
      - IMPORT SOURCE(REPO)
      - DELETE DEFN
      - QUERY with SHOW(DEFN)
  - Non-authorized RM callers
    - **CSLURP10** (RDDS to Repository RM utility)
    - **CSLURP20** (Repository to RDDS RM utility)
- Access to a Repository Server can be gained directly by either of the following
  - **Batch ADMIN** utility
  - **z/OS modify** interface

219

A user repository can be accessed either through RM or directly. If going through RM, the caller will be considered either “authorized” or “non-authorized”. IMS is an authorized caller and therefore has access to all repository contents via commands, so long as RM is authorized. On the other hand, the RM utilities are non-authorized RM callers. In this case, the utilities will not automatically have authorization for repository access just because RM is authorized, as is the case with IMS. Therefore, the RM utilities require separate authorization to access the repository as we will see on the following slide.

As previously mentioned, the batch ADMIN utility runs as a JCL job. Therefore, the user ID specified in the JCL can be used for authorization checking to determine whether repository access is allowed. Security for commands entered through the z/OS modify interface can be implemented using standard console security.

## ***Types of Repository Security***

- **Connection security**
  - Used by both authorized and non-authorized RM callers when they attempt to connect to the repository
  - RM specifies its USERID on its startup JCL, to be used in RACF authorization
    - If RM is authorized to access the repository, so is IMS since it is an authorized RM caller
  - RM Utilities specify their USERIDs in JCL, to be used for SCI registration and for RACF authorization
- **Member-level security**
  - Only used with non-authorized RM callers that access individual members within the repository
    - CSLURP10
    - CSLURP20

220

Repository security can be at the connection or member level.

Connection security applies to both authorized and non-authorized RM callers, which we mentioned were IMS and the RM utilities (respectively) on the previous slide. In either case, the caller must specify a user ID in the JCL, which will be checked in RACF to determine whether access to the repository is allowed. We will cover how to restrict access to the user repository, as well as other Repository Resources, in the next few slides.

Security can also be implemented at the member level within a user repository, which only applies to the RM utilities. In this case, you can restrict access to individual members, and separately authorize the user ID associated with the RM utility (again, specified in the JCL) being used to access these members. We will also cover how to do this in the next few slides.

Use member-level security if you want to restrict the RM utilities to accessing only certain resources. Once you have protected the individual resources in a RACF class (covered next a few slides from here), permit the user ID specified in the utility's JCL to access these resources accordingly. For example, the CSLURP20 utility reads repository resources and copies them to an RDDS. If you want to prevent the utility from reading certain resources from the repository and then copying them to the RDDS, restrict access to them and only permit the utility to access (read/copy) the desired resources. Examples of restricting access to individual resources and permitting user IDs to access them are shown later.

## **Repository Security Implementation**

- Repository Server resources can be restricted from unauthorized access, including:
  - User repository
  - RS catalog repository
  - Members within a user repository
  - Audit levels associated with an individual repository
- Choose a class to protect Repository Server resources in
  - FACILITY or
  - User-defined class (recommended if using member-level security due to 39-character profile name length restriction of FACILITY class)
    - Add new class to RACF Class Descriptor Table (ICHRRCDE)
    - Update RACF Router Table (ICHRFR01) with new class
- Protect resources by defining general resource profiles
- Grant access to users using defined resource profiles

221

There are several resources associated with the Repository Server that can be protected in RACF or other SAF interface, shown here. The process of restricting access to these resources is three-fold:

1. Define the security class that will protect the resources. If using RACF, the FACILITY class will suffice unless individual resources within the repository need protection. In this case, a user-defined class is better since it allows for longer profile names, which can become lengthy due to the required format.
2. Restrict access to the resources.
3. Grant permissions to selected user IDs.

## ***Protecting Repository Server Resources***

- Define profiles to restrict access to Repository Server resources to authorized users
- User repository
  - Format for defining resource profile
    - **FRPREP.repositoryname**
  - Example
    - **RDEFINE XFACILIT FRPREP.REPO1 UACC(NONE)**
    - **RDEFINE XFACILIT FRPREP.\* UACC(NONE)**
- RS catalog repository
- Repository users can access/update with batch ADMIN commands
  - Format for defining resource profile
    - **FRPREP.CATALOG**
  - Example
    - **RDEFINE XFACILIT FRPREP.CATALOG UACC(NONE)**

In this and the following examples, assume that XFACILIT is a user-defined RACF class. This slide shows the specific formats that must be used when restricting both a user repository and a Repository Server catalog repository from unauthorized access.



## Protecting Repository Server Resources

- Members within a repository (for non-authorized RM callers only)

– FRPMEM.repositoryname.DFS.RSC.membername

plexname + rsctype + rscname

plexname

8-byte CSL plexname where repository is defined (MUST start with characters "CSL")

rsctype

8-byte resource name to be secured

rscname

8-byte resource type:  
DB,DBDESC,PGM,PGMDESC,  
TRAN,TRANDESC,RTC,RTCDESC

– RDEFINE XFACILIT ← Example

FRPMEM.REPO1.DFS.RSC.CSLPLEX1.TRAN.PART UACC(NONE)

223

When defining profiles for individual resources, there is a particular format that must be used which is shown here. Note that the part of the profile definition containing the resource name actually consists of 3 items: the IMSplex name, the resource type and the resource name. All of these are 8-bytes long.

## ***Protecting Repository Server Resources***

- Audit levels associated with an individual repository
  - Format for defining resource profile
  - `FRPAUD.repositoryname.DFS.RSC.TYPE`
  - Example
    - `RDEFINE XFACILIT FRPAUD.REPO1.DFS.RSC.TYPE UACC(NONE)`


As previously discussed, there can be different audit levels associated with the repository that track various types of resource/member access. Only certain users should be able to change the repository's audit level. Access can be restricted by defining a resource profile for the audit level using the format shown on this slide. Appropriate users would then be granted access to change the audit level, which we will discuss shortly.

## Granting User Access to Repository Server Resources


- After defining resource profiles, grant access to appropriate users

- User repository

- RDEFINE XFACILIT FRPREP.REPO1 UACC(NONE)




- PERMIT FRPREP.REPO1 CLASS(XFACILIT) ID(VIEWER1)  
ACCESS(READ)



- PERMIT FRPREP.REPO1 CLASS(XFACILIT) ID(ADMIN1)  
ACCESS(ALTER)

- RS catalog repository

- RDEFINE XFACILIT FRPREP.CATALOG UACC(NONE)



- PERMIT FRPREP.CATALOG CLASS(XFACILIT) ID(ADMIN1)  
ACCESS(ALTER)

Once all of the resource profiles have been defined for the RS resources, the next step is granting access to specific users. This slide shows examples of granting access to a user repository as well as a Repository Server catalog repository using RACF.

## Granting User Access to Repository Server Resources

- Members with a repository

```

- RDEFINE XFACILIT
  FRPMEM.REPO1.DFS.RSC.CSLPLEX1.TRAN.PART UACC(NONE)
- PERMIT FRPREP.REPO1 CLASS(XFACILIT) ID(USRUTL10)
  ACCESS(UPDATE)
- PERMIT FRPMEM.REPO1.DFS.RSC.CSLPLEX1.TRAN.PART
  CLASS(XFACILIT) ID(USRUTL10) ACCESS(UPDATE)
- PERMIT FRPMEM.*.*.*.*.*.*
  CLASS(XFACILIT) ID(USRUTL20) ACCESS(READ)

```

- Repository audit levels

```

- RDEFINE XFACILIT FRPAUD.REPO1.DFS.RSC.TYPE UACC(NONE)
- PERMIT FRPAUD.REPO1.DFS.RSC.TYPE CLASS(XFACILIT)
  ID(USRZOSMI) ACCESS(UPDATE)

```

226

Here, we show examples of granting access to individual members within a repository. Note that if a user ID needs RACF UPDATE access for individual members that have been restricted from unauthorized access, the user ID also needs RACF UPDATE access for the repository that these members are contained in. Therefore a separate RACF PERMIT statement would be required to ensure this access. The example accounts for this, as you can see.

This slide also shows an example of granting a user ID permission to change the audit access level associated with a user repository.

## RACF Groups

- Able to group several user IDs together for higher efficiency when defining resource profiles and granting access to them
  - PERMITs will reference RACF group rather than each individual user ID
- Example

```
RDEFINE XFACILIT FRPREP.REPO1 UACC(NONE)
ADDGROUP FRPVIEW
ADDGROUP FRPEDIT
PERMIT FRPREP.REPO1 CLASS(XFACILIT) ID(FRPVIEW) ACCESS(READ)
PERMIT FRPREP.REPO1 CLASS(XFACILIT) ID(FRPEDIT) ACCESS(UPDATE)
CONNECT <VIEWER1> GROUP(FRPVIEW)
CONNECT <VIEWER2> GROUP(FRPVIEW)
CONNECT <VIEWER3> GROUP(FRPVIEW)
CONNECT <UPDATER4> GROUP(FRPEDIT)
CONNECT <UPDATER5> GROUP(FRPEDIT)
```

227

When using RACF and working with a large number of user IDs that have similar permissions to access protected resources, it is more efficient to group them together. This way, when granting permissions, group names can be referenced instead of individual user IDs. Using this method results in less overhead when dynamically managing permission changes. This slide shows an example of aggregately granting permission to several user IDs contained in the same group.

## DRD User Interface Enhancements

228

All new and enhanced DRD commands in IMS 12 were incorporated into the DRD User Interface (UI). The DRD UI is a ISPF panel driven interface which assists you in entering various DRD commands. **It is contained in the IMS application called Manage Resources, which can be invoked from the IMS application menu.**

## ***DRD UI Enhancements (Manage Resources Application)***

- **New panels**
  - EXPORT DEFN initial TARGET panel
  - EXPORT DEFN (REPO)
  - IMPORT DEFN initial SOURCE panel
  - IMPORT DEFN (REPO)
  - DELETE DEFN
- **Enhanced panels**
  - IMPORT DEFN (RDDS)
  - QUERY DB
  - QUERY TRAN
  - QUERY TRAN DESC
  - QUERY DBDESC
  - QUERY PGM
  - QUERY PGMDESC
  - QUERY RTC
  - QUERY RTCDESC

New panels were added to the DRD UI and existing panels were enhanced, shown on this slide. The next series of slides show example screenshots for various commands in “list view” unless otherwise noted. The alternative view is “syntax view” for more experienced IMS users that are familiar with command format. As you will see in the next series of slides, we’ve only included screenshots for new panels and panels that were enhanced with additional parameter information. Panels that were not changed, but that now apply to repository DRD (in addition to RDDS DRD), are not shown.

## ***DRD UI Enhancements (Manage Resources Application)***

- New DRD utilities that create a repository using either of the following as input:
  - MODBLKS data set
  - IMS log
- Available via service stream with APAR PM41281



## Enhanced EXPORT DEFN TARGET() Panel

```
File Action Manage resources SPOC View Options Help
-----
PLEX1          IMS Export Resources
Command ==> _____

----- Plex. . _____ Route. . _____ Wait. . _____
Select a target. Press Enter to continue.

* TARGET . . . . . : _ 1. RDDS
                   2. REPO
NAME Resource name . . . . . _____

F1=Help      F12=Cancel
```

## New EXPORT DEFN TARGET(REPO) Panel

```
File Action Manage resources SPOC View Options Help
-----
PLEX1                      IMS Export Resources
Command ==> _____
----- Plex. . _____ Route. . _____ Wait. . _____
Press Enter to continue.
Target . . . . . REPO                                     More: +
Resource name . . . . . _____
SET
  IMSID . . . . . _____
Start time . . . . . _____ YYYY.DDD HH:MM:SS:TH
End time . . . . . _____ YYYY.DDD HH:MM:SS:TH

F1=Help   F7=Up   F8=Down   F12=Cancel
```

## New EXPORT DEFN TARGET(REPO) Panel

```

File Action Manage resources SPOC View Options Help
-----
PLEX1                      IMS Export Resources
Command ==> _____

----- Plex. . _____ Route. . _____ Wait. . _____
Press Enter to continue.
More: - +

Resource type
Enter "/" to select types
- ALL      All RSCs & DESCs
- ALLRSC   All resources
- DB       Database resource
- PGM      Program resource
- RTC      Routing code resource
- TRAN     Transaction resource
- ALLDESC  All descriptors
- DBDESC   Database descriptor
- PGMDESC  Program descriptor
- RTCDESC  Routing code descriptor
- TRANDESC Transaction descriptor

OPTION
Enter "/" to select options
- ALLRSP   Show all responses
- CHANGEONLY Export to REPO

F1=Help    F7=Up    F8=Down    F12=Cancel

```

## New EXPORT DEFN TARGET(REPO) Panel – Syntax View

```
File Action Manage resources SPOC View Options Help
-----
PLEX1                               IMS Export Resources
Command ==> _____

----- Plex. . _____ Route. . _____ Wait. . _____
Press Enter to continue.
EXPORT DEFN TARGET (REPO)
  NAME ( _____ )
  SET ( IMSID ( _____ ) )
  STARTTIME ( _____ )
  ENDTIME ( _____ )
  TYPE ( _____ )
  OPTION ( _____ )

F1=Help  F12=Cancel
```

## Enhanced IMPORT DEFN SOURCE() Panel

```
File Action Manage resources SPOC View Options Help
-----
PLEX1          IMS Import Resources
Command ==> _____

----- Plex. . . . . Route. . . . . Wait. . . . .
Select a source. Press Enter to continue.

* SOURCE . . . . . : _ 1. RDDS
                  2. REPO
NAME Resource name . . . . . _____

F1=Help      F12=Cancel
```

## Enhanced IMPORT DEFN SOURCE() Panel

```
File Action Manage resources SPOC View Options Help
-----
PLEX1                               IMS Import Resources
Command ==> _____

----- Plex. . _____ Route. . _____ Wait. _____
Press Enter to continue.

Source . . . . . REPO

Resource name . . . . . _____

Resource type
Enter "/" to select types
- ALL      All RSCs & DESCs
- ALLRSC   All resources
- DB       Database resource
- ALLDESC  All descriptors
- DBDESC   Database descriptor

F1=Help  F12=Cancel
```

## Enhanced IMPORT DEFN SOURCE() Panel

```

File Action Manage resources SPOC View Options Help
-----
PLEX1                      IMS Import Resources
Command ==> _____

----- Plex. . _____ Route. . _____ Wait. _____
Press Enter to continue.

- PGM Program resource          - PGMDESC Program descriptor
- RTC Routing code resource     - RTCDESC Routing code descriptor
- TRAN Transaction resource     - TRANDESC Transaction descriptor

OPTION
Enter "/" to select options
- ABORT Abort import if error   - ALLRSP Show all responses
- UPDATE Update runtime defs

SCOPE
Where to apply update. . . . . - 1. All
                               - 2. Active

F1=Help  F12=Cancel

```

## Enhanced IMPORT DEFN SOURCE() Panel – Syntax View

```
File Action Manage resources SPOC View Options Help
-----
PLEX1                      IMS Import Resources
Command ==> _____

----- Plex. . _____ Route. . _____ Wait. _____
Press Enter to continue.

IMPORT DEFN SOURCE(REPO)
NAME ( _____ )
TYPE ( _____ )
OPTION( _____ )
SCOPE( _____ )

F1=Help  F12=Cancel
```



## Enhanced IMPORT DEFN SOURCE(RDDS) Panel

```
File Action Manage resources SPOC View Options Help
-----
PLEX1                      IMS Import Resources
Command ==> _____
-----
Plex. . . . . Route. . . . . Wait. . . . .
Press Enter to continue.

Source . . . . . RDDS

Source RDDS data set . . . _____

Resource name . . . . . _____

< Resource Type options not shown >

OPTION
Enter "/" to select options
_ ABORT  Abort import if error  _ ALLRSP  Show all responses
_ UPDATE Update runtime defs

F1=Help  F12=Cancel
```

## Enhanced DELETE Panel

```
File Action Manage resources SPOC View Options Help
-----
PLEX1          IMS Delete
Command ==> _____

----- Plex. . _____ Route. . _____ Wait. . _____
Select and press Enter to continue.

* DELETE . . . . . : _ 1. RESOURCES
                   2. DEFN

F1=Help      F12=Cancel
```

**New DELETE DEFN Panel**

```

File Action Manage resources SPOC View Options Help
-----
PLEX1                      IMS Delete DEFN
Command ==> _____

----- Plex. . _____ Route. . _____ Wait. . _____
Press Enter to continue.
More: - +

Resource type
Enter "/" to select types

- DB      Database resource      - DBDESC  Database descriptor
- PGM     Program resource       - PGMDESC Program descriptor
- RTC     Routing code resource  - RTCDESC Routing code descriptor
- TRAN    Transaction resource   - TRANDESC Transaction descriptor

OPTION
Enter "/" to select options
- ALLRSP  Show all responses

F1=Help   F7=Up    F8=Down  F12=Cancel

```

## New DELETE DEFN Panel – Syntax View

```
File Action Manage resources SPOC View Options Help
-----
PLEX1                                IMS Delete Repository
Command ==> _____

----- Plex. . _____ Route. . _____ Wait. . _____
Press Enter to continue.
DELETE DEFN TARGET(REPO)
NAME ( _____ )
FOR(IMSID( _____ ))
TYPE ( _____ )
OPTION ( _____ )

F1=Help  F12=Cancel
```

## Enhanced QUERY Panel

```

File Action Manage resources SPOC View Options Help
-----
PLEX1                IMS Query Databases by Attribute
Command ==> _____

----- Plex. . _____ Route. . _____ Wait. _____
Press Enter to continue.

NAME Database name . . . . _____
      Resource type . . . . 1  1. Resource
                                2. Descriptor

< some options not shown >

SHOW Show these columns in the report
Enter "/" to select option
- ALL      All information          IMSID  Resources defined from repo
- ACCTYPE  Type of access          - LOCAL LOCAL value
- DEFAULT  Descriptor only         - MODEL Model information
- DEFN     Resource definitions     - RESIDENT Resident option
- DEFNTYPE Definition type         - STATUS Local database status
- GLOBAL   Global value            - TIMESTAMP Various timestamps

F1=Help    F3=Help    F4=Showlog  F6=Expand  F9=Retrieve  F12=Cancel

```

## Enhanced QUERY Panel – Syntax View

```

File Action Manage resources SPOC View Options Help
-----
PLEX1                      IMS Query Databases by Attribute
Command ==> _____

----- Plex. . _____ Route. . _____ Wait. _____
Press Enter to continue.

QUERY DB NAME( _____ )
SHOW( ALL, ACCTYPE, DEFN, DEFNTYPE, GLOBAL, IMSID, LOCAL, MODEL
      RESIDENT, STATUS, TIMESTAMP )
STATUS( ALLOCF, ALLOCS, BACKOUT, EEQE, LOCK, NOTINIT, NOTOPEN
        OFR, OLR, OPEN, RECALL, RECOV, RNL, STOACC, STOSCHD
        STOUPDS )
TYPE( DEDB, DLI, MSDB, PART, PHDAM, PHIDAM, PSINDEX )

F1=Help    F3=Help    F4=Showlog  F6=Expand  F9=Retrieve F12=Cancel

```

## Installation Verification Program (IVP) Enhancements for Repository

The next section describes how the IMS IVP application was enhanced to support repository DRD.

## ***IVP Repository Enhancement***

- IVP has been enhanced to include sample JCL to create the RS catalog repository data sets and a user repository
  - Repository Server configuration file
  - Repository Server startup procedure
  - JCL to execute the following:
    - Start a Repository Server
    - Add a user repository to the RS catalog repository
    - List user repository status information
    - Populate a user repository
    - Rename a user repository in the RS catalog repository
    - List detailed information for a single user repository
    - Modify and update user repository definitions
    - Delete a user repository in the RS catalog repository
    - Delete actual RS catalog repository and user repository data sets

Here is a list of new jobs/tasks that were added to the IVP for repository DRD.



## ***IVP Repository Enhancement***

- Enhancement contained within *O series* of steps within IVP
- No change to IVP system definition process (*C series*)
- Updated jobs/tasks
  - IV\_D201T and IV\_D202T: updated to APF-authorize and place the FPQCSSI2/FPQCXCF2 modules into LPA (if required)
  - Job IV\_E302J: updated to add the new user repository server procedure and the server configuration member into PROCLIB
  - Job IV\_O101J: updated to create RS catalog repository data sets and user repository data sets
- New jobs/tasks
  - IV\_O200J: starts the Repository Server
  - IV\_O223J: adds a user repository to the RS catalog repository
  - IV\_O224J: lists user repository status information
  - IV\_O225J: populates the user repository

## ***IVP Repository Enhancement***

- **New jobs/tasks**
  - IV\_O226J: renames a user repository in the RS catalog repository
  - IVPO227J: lists detailed information for a single user repository
  - IVPO228J: modifies and updates definitions for a user repository
  - IVPO229J: deletes a user repository in the RS catalog repository
  - IVPO230J: requests the Repository Server to start a user repository already deleted
  - IVPO252T: stops Repository Server
  - IV3O401J: deletes RS catalog repository and user repository data sets

# Summary

## ***Topics Covered***

- **Repository overview, functions and setup**
  - IMS Repository Function Infrastructure
  - IMS Repository set-up and access
- **Repository use and management**
  - IMS repository commands
  - 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
  - DRD user interface enhancements
  - IVP enhancements for repository

## ***DRD Configurations for IMS 12 Users***

- No DRD (MODBLKS OLC)
- RDDS DRD with system/non-system RDDSs
- Repository and RDDS DRD together
- Repository DRD

## ***Value of the IMS 12 repository for DRD***

- 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 in a single place
- Provides improved availability
  - Repository can be enabled/disabled without an IMS outage via command
  - Duplexing of data plus spare capability improves data availability
- Provides single source consistency for DRD stored resource definitions
  - No need for multiple sets of RDDSs in a multiple-IMS IMSplex
  - No need for coordinating multiple sets of RDDSs in a multiple-IMS IMSplex
  - Repository architecture controls consistency and integrity of data

## ***Value of the IMS 12 Repository for DRD***

- Provides improved functionality and flexibility for managing resources across an IMSplex
  - Generic resource definition plus IMS-specific resource definitions
  - EXPORT process is a single unit of work for entire IMSplex, all succeeds or all fails
  - EXPORT process controlled by user (no AUTOEXPORT)
    - Can select CHANGESONLY or by time periods
  - DELETE of stored resource definitions controlled by user
  - Can UPDATE an existing runtime definition via IMPORT
  - EXPORT reflected in all IMSs in an IMSplex, whether up or down at the time
  - QUERY will display stored resource definitions from repository
  - DFSINSX0 (Destination Creation Exit) supports export to repository

## ***Value of the IMS 12 Repository for DRD***

- Provides support for both test and production environments
  - Repository Server can include data from different IMSplexes though one per IMSplex recommended
  - Multiple IMSRSC repositories can exist within one Repository Server though one per Repository Server recommended
  - Migration and fallback utilities available based on RDDSs
    - Previously available DRD RDDS utilities can be used in backup/recovery scenarios
  - IVP available to assist with installation of repository
  - Supported by TSO SPOC Manage Resources application



## ***Value of the IMS 12 Repository for DRD***

- **Provides security capabilities for auditing and compliance**
  - Full support for RACF (SAF) interfaces
  - Repository audit log (optional)
    - Includes both online and batch access
  - OM type-2 repository commands optionally found in OM Audit Trail
- **Provides comprehensive set of repository administration tools**
  - Includes batch utilities and command interfaces for repository management
    - Can be performed when IMS is down
- **A strategic IMS architectural direction**
  - Based upon BPE, CSL, IMSplex architecture

# Appendix

## Example Use Case Scenarios

Let's now take a look at some example scenarios that involve using the repository with DRD.

## Use Case 1: Cloned Environment...

- Handling runtime resource definition updates in a cloned 3-way IMSplex, where IMSA and IMSB are active and IMSC is inactive
  - CREATE PGM NAME(PGMCAR) routed to IMSA and IMSB
  - EXPORT DEFN TARGET(REPO) TYPE(PGM) NAME(PGMCAR) SET(IMSID(\*)), take default routing to one command master IMS
    - Newly created program PGMCAR is added to IMSA's, IMSB's and IMSC's IMS resource lists and the stored definition is written to the repository
  - DELETE PGM NAME(PGMBUS) routed to IMSA and IMSB
  - DELETE DEFN TARGET(REPO) TYPE(PGM) NAME(PGMBUS) FOR(IMSID(\*)), take default routing to one command master IMS
    - Program PGMBUS deleted from IMSA's, IMSB's and IMSC's IMS resource lists and the stored definition is also deleted from the repository

258

In this first use case, we've got 3 IMS systems that are clones of one another. Two of the IMS systems are active while the other is inactive. There are 2 assumptions for this use case:

⑩ Assumption 1: IMSA, IMSB and IMSC have been defined to the repository and each have entries within the repository from either an EXPORT or from the CSLURP10 utility having been run.

⑩ Assumption 2: The PGMBUS program resource is defined as a runtime resource definition on all IMS systems and also exists as a stored resource definition for all IMS systems within the repository.

This scenario reviews the commands that would be issued when a user would create a new program resource and delete another program resource. It also shows the commands that would be issued in order to write these resource changes to the repository for all 3 IMS systems, even though one is inactive. Note: **the first CREATE command creates the program resource PGMCAR at IMSA and IMSB. It is not created at IMSC, since the command is not routed to it since IMSC is not active to process the command.**

## ***Use Case 1: Cloned Environment***

- Continued...
  - How will IMSC apply all of these changes?
    - Coldstart will automatically import the updated resource definitions
    - Warmstart will necessitate user issuing the following commands routed to IMSC:
      - `IMPORT DEFN SOURCE(REPO) TYPE(PGM) NAME(PGMCAR)`
      - `DELETE PGM NAME(PGMBUS)`

The scenario continues with a consideration for the inactive IMS, and how it would apply the resource changes that have occurred while it was down. Note that depending on the type of restart for the inactive system, certain additional actions may need to be taken, and are shown on this slide.

## **Use Case 2: Non-Cloned Environment (Unique Systems)**

- Handling runtime resource definition updates in a 2-way IMSplex, where IMSA and IMSB are active and have different resource definitions
- Route commands to individual IMS systems, for example:
  - Route the following commands to IMSA only
    - UPDATE PGM NAME(PGMAAA) SET(TRANSTAT(Y))
    - EXPORT DEFN TARGET(REPO) TYPE(PGM) NAME(PGMAAA)
    - DELETE PGM NAME(PGMBBB)
    - DELETE DEFN TARGET(REPO) TYPE(PGM) NAME(PGMBBB) FOR(IMSID(IMSA))
  - Route the following commands to IMSB only
    - UPDATE PGM NAME(PGMCCC) SET(TRANSTAT(Y))
    - EXPORT DEFN TARGET(REPO) TYPE(PGM) NAME(PGMCCC)
    - DELETE PGM NAME(PGMDDD)
    - DELETE DEFN TARGET(REPO) TYPE(PGM) NAME(PGMDDD) FOR(IMSID(IMSB))
- Exporting to and deleting from the repository ensures that these changes are hardened and the updated stored resource definitions are available for the next coldstart

260

In the next scenario, we've got 2 active IMS systems and this time they are not clones of one another. Therefore each IMS system has different resource definitions associated with it. This use case illustrates how to handle resource updates given these circumstances -- specifically, where we are updating certain resource attributes as well as deleting resources from each system.

The EXPORT and DELETE DEFN commands update the stored resource definitions in the repository for each IMS system and makes them available for the next coldstart. During a warmstart or emergency restart, IMS builds the runtime resource definitions from the IMS log. The IMS repository is not accessed during a warmstart or emergency restart.

### **Use Case 3: Mixed (Cloned + Unique Systems)...**

- **Handling runtime resource definition updates in a 5-way IMSplex**
  - IMSA/IMSB/IMSC are clones (IMSA/IMSB active, IMSC inactive)
  - IMSD/IMSE are not cloned and are both active
- **For cloned systems (IMSA, IMSB, IMSC):**
  - CREATE DB NAME(DB123) routed to IMSA and IMSB
  - EXPORT DEFN TARGET(REPO) TYPE(DB) NAME(DB123)  
SET(IMSID(IMSA IMSB IMSC)) routed to either IMSA or IMSB
    - New DB123 database added to IMSA's, IMSB's, and IMSC's resource lists and stored resource definitions
      - IMSC will import DB123 at next coldstart, or if warmstarted user can issue IMPORT DEFN SOURCE(REPO) TYPE(DB) NAME(DB123)
  - DELETE DB NAME(DB456) routed to IMSA and IMSB
  - DELETE DEFN TYPE(DB) NAME(DB456) FOR(IMSID(IMSA IMSB IMSC))
    - DB456 database deleted from IMSA's, IMSB's, and IMSC's resource lists and stored resource definitions
    - DB456 database will not be imported at these systems at next coldstart due to deletion (if IMSC warmstarts, user issues DELETE for runtime def)

261

In our next use case, we have 5 IMS systems with the following scenario:

- ⑩ 4 IMS systems are active
- ⑩ 1 IMS system is inactive
- ⑩ The IMS systems are a mix of clones and non-clones

Here, we review the steps that would be required in order to both create new resources and delete existing resources from all of these systems while saving these changes in the repository. This particular slide covers the case where each IMS system is a clone.

### ***Use Case 3: Mixed (Cloned + Unique Systems)***

- For unique systems (IMSD, IMSE), route commands to the individual IMS systems, for example:
  - Route all commands to IMSD only
    - CREATE DB NAME(DB345)
    - EXPORT DEFN TARGET(REPO) TYPE(DB) NAME(DB345)
    - DELETE DB NAME(DB456A)
    - DELETE DEFN TARGET(REPO) TYPE(DB) NAME(DB456A) FOR(IMSID(IMSD))
  - Route all commands to IMSE only
    - CREATE DB NAME(DB789)
    - EXPORT DEFN TARGET(REPO) TYPE(DB) NAME(DB789)
    - DELETE DB NAME(DB1011)
    - DELETE DEFN TARGET(REPO) TYPE(DB) NAME(DB1011) FOR(IMSID(IMSE))
- Exporting to and deleting from the repository ensures that these changes are hardened and the updated stored resource definitions are available for the next coldstart

262

The scenario continues where we again review the steps that would be required in order to both create new resources and delete existing resources from all of these systems while saving these changes in the repository. This particular slide covers the case where each IMS system is a non-clone and has differing, unique resources from the other.

The EXPORT and DELETE DEFN commands update the stored resource definitions in the repository for each IMS system and makes them available for the next coldstart. During a warmstart or emergency restart, IMS builds the runtime resource definitions from the IMS log. The IMS repository is not accessed during a warmstart or emergency restart.



## Use Case 4: Porting Resource Definitions (Online Method)

- Copying resource definitions from a development system (IMSA) to a test system (IMSB)
  - Assumptions:
    - Each IMS exists in a different IMSplex and is using a different repository
      - IMSA in PLEXA uses single point of control SPOCA
      - IMSB in PLEXB uses single point of control SPOCB
    - Each IMS is active
  - Export IMSA's runtime definitions to a non-system RDDS
    - EXPORT DEFN TARGET(RDDS) RDDSDSN(NONSYS.RDDS1) TYPE(DB) NAME(DB11 DB22) routed to IMSA using SPOCA
  - Run CSLURP10 on PLEXB using NONSYS.RDDS1 non-system RDDS as input and IMSB's repository as output
  - Import stored definitions to IMSB from its repository
    - IMPORT DEFN SOURCE(REPO) TYPE(DB) NAME(DB11 DB12) routed to IMSB using SPOCB
  - But what if these IMS systems were inactive...?

263

Next, we have a use case in which porting resource definitions from one IMS system to another is illustrated. In the scenario, there are 2 active IMS systems that exist in different IMSplexes. Because the IMS systems exist in separate IMSplexes, they each have different repositories and also have separate Single Points of Control (or SPOCs) from which type-2 commands can be entered.

The scenario shows example steps that a user would take to port resource definitions from one IMS to another, using a combination of the type-2 EXPORT and IMPORT commands with a non-system RDDS as a common midpoint. Once again, these 2 IMS systems included in the scenario are active. Next, we discuss how the steps would be different if the IMS systems were inactive.

## **Use Case 5: Porting Resource Definitions (Offline Method)**

- Copying resource definitions from a development system (IMSA) to a test system (IMSB) with same assumptions as Use Case 4
  - Run CSLURP20 on PLEXA using IMSA's repository as input and a non-system RDDS as output
  - Run CSLURP10 on PLEXB using the non-system RDDS as input and IMSB's repository as output
  - When IMSB is active, import stored definitions from its repository
    - `IMPORT DEFN SOURCE(REPO) TYPE(DB) NAME(DB11,DB12)`

You are able to port resource definitions from the repository that one IMS is using to a completely different repository that another IMS is using. Simply run the CSLURP20 and CSLURP10 utilities using a non-system RDDS as a common midpoint to move the stored resource definitions from one repository to the other. Once the stored definitions have been successfully ported to the targeted IMS system's repository, they can then be imported for use in the active system as runtime resource definitions using the type-2 IMPORT command.

## Use Case 6: Displaying and Printing Stored Definitions in Repository

### ▪ When IMS systems are active

- Issue QUERY command with SHOW(DEFN) specified to display list of resources (and their attribute values) within repository
  - Includes a display of which IMS systems have specific resources defined
  - Example: QUERY DB NAME(\*) SHOW(DEFN) to display all databases
- Tip: For the best command output results view, opt to group lines by either column or resource (not wrap) under SPOC preferences:

```
Format of listing . . . 2 1. Wrap individual lines.
                       2. Group lines by column.
                       3. Group lines by resource.
```

### ▪ When IMS systems are inactive

- Run CSLURP20 “Repository to RDDS” utility to generate non-system RDDS with repository contents
- Run DFSURDD0 “Extract RDDS contents” utility to generate a query report containing a list of resources with attribute values
- Follow this process for each IMS that is using the repository, one at a time

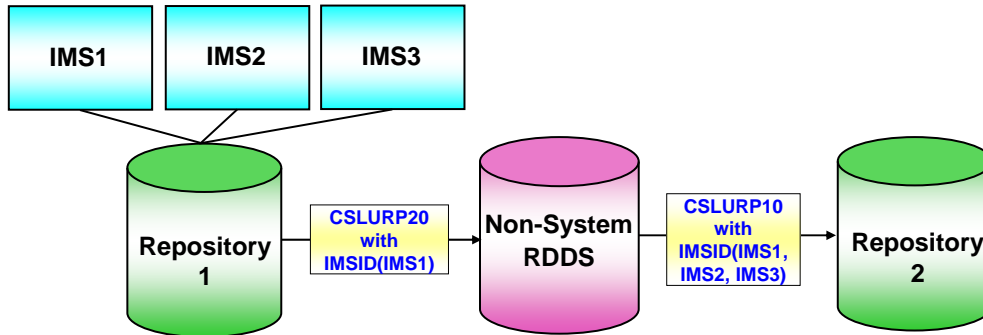
265

A common question that users ask is how to view repository contents. There are two ways, depending on whether or not any repository-enabled IMS systems are active. If at least one repository-enabled IMS is active, you can issue a type-2 QUERY command with specific resources names to determine whether they exist in the repository. Or to determine the names of all resources of a particular type that exist in the repository, an “\*” can be specified for the name as long as the resource type is included in the command. For example, to determine which databases are currently defined in the repository, as well as which IMS systems have these databases defined to them, issue a QUERY DB NAME(\*) SHOW(DEFN,IMSID) command. Note that the default listing format in the TSO SPOC application is “wrap individual lines”. A better choice for viewing command output would be grouping by either column or resource, so be sure to specify one of these settings in the SPOC preferences (accessed from the “SPOC” menu).

If no IMS systems are active, you can run utilities offline to determine repository contents. First, run the CSLURP20 utility using the desired repository as input and specify the name of a non-system RDDS for the output data set. Next, run the DFSURDD0 utility using the non-system RDDS used in the previous step as input, and designate that a query report should be generated. The query report will display a list of all resources along with their attribute values for the particular IMS system specified in the utility JCL. If there are multiple IMS systems that are using the repository, repeat this process for each IMS that you would like to view the stored resource definitions for.

## Use Case 7: Copying Repositories...

- To copy the contents of one repository to another in a cloned environment:
  - Capture all stored resource definitions in the initial repository by running CSLURP20 “Repository to RDDS” utility against it, specifying any one IMSID in the JCL, and using a non-system RDDS as output
  - Transfer these stored definitions to a different repository by running CSLURP10 “RDDS to Repository” utility specifying all IMSIDs in the JCL, using the non-system RDDS as input, and the new repository as output



266

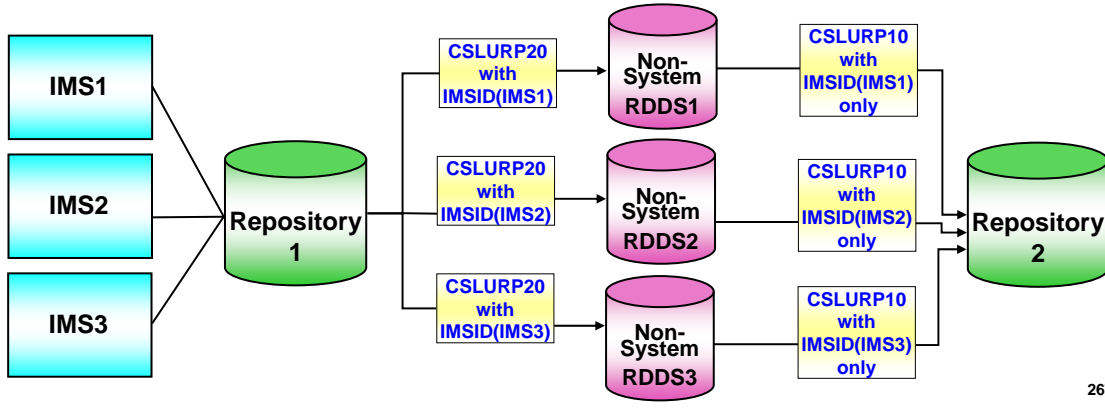
In order to copy the contents of one repository to another, the steps vary depending on whether the user has a cloned environment or not. This slide discusses a cloned environment, in which a user would run CSLURP20 for one IMS system, creating one non-system RDDS. Then they'd run CSLURP10 and specify multiple IMSIDs (for each of the cloned IMSs) on the utility's SYSIN DD, for example:

```
//SYSIN      DD      *
IMSPLEX(NAME=PLEX1 IMSID(IMS1,IMS2,IMS3))
```

Note: You can create a backup of Repository-1 and use the steps to restore the backup at Repository-2 after Repository-2 is started with empty datasets.

## Use Case 7: Copying Repositories

- To copy the contents of one repository to another in a non-cloned environment:
  - Capture all stored resource definitions in the initial repository by running CSLURP20 “Repository to RDDS” utility against it for each IMS, each time using a different non-system RDDS as output
  - Transfer these stored definitions to a different repository by running CSLURP10 “RDDS to Repository” utility once for each IMS, using the non-system RDDS as input and the new repository as output



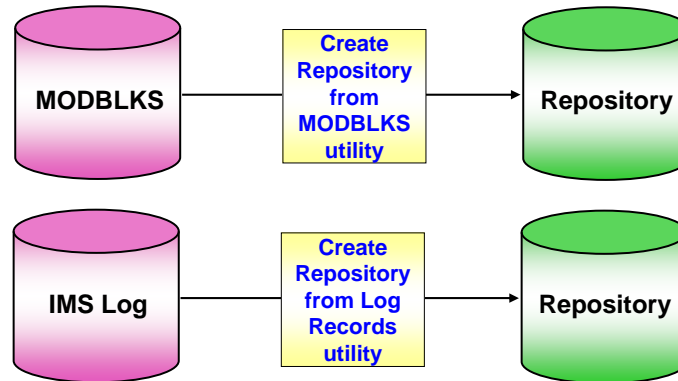
267

For a **non-cloned** environment: a user would run CSLURP20 for each IMS system, creating a different RDDS for each one. Then they'd run CSLURP10 for each RDDS that was created, and the repository would be preserved each time to allow for multiple updates.

Note: You can create a backup of Repository-1 and use the steps to restore the backup at Repository-2 after Repository-2 is started with empty datasets.

## Use Case 8: Creating Repositories from MODBLKS or IMS Log

- To create a repository from the contents of a MODBLKS data set, run the Create Repository from MODBLKS utility
- To create a repository from the contents of the IMS log, run the Create Repository from Log Records utility
- The new options for Manage Resources panels to be able to populate the repository will be available with APAR PM41281



268

You can use existing data in your shop such as the MODBLKS data set or the IMS log data sets to generate a repository with equivalent contents. APAR PM41281 is enhancing the Manage Resources panels to write to the repository using existing DRD and RM utilities. Until APAR PM41281 is available, you can perform a two step process to populate the repository from MODBLKS or the IMS log:

- Create a non-system RDDS from MODBLKS or the IMS log
  - Use DRD utility DFSURCM0 to create a non-system RDDS from MODBLKS
  - Use DRD utility DFSURCL0 to create a non-system RDDS from the IMS log
- Run CSLURP10 to populate the repository with the RDDS created in the above step

# Commands

## New UPDATE RM Command

```
UPDATE RM TYPE(REPO) REPOTYPE(IMSRSC)
SET(REPO(Y|N) AUDITACCESS())
```

- [Use command to dynamically enable/disable repository usage for the RM address space](#)
  - Make CSLRIxxx changes first if enabling usage
- **TYPE(REPO)**
- **REPOTYPE(IMSRSC)**
- **SET(REPO(Y))** dynamically enables RM repository usage and <SECTION=REPOSITORY> is reprocessed
- **SET(REPO(N))** dynamically disables RM repository usage
- **AUDITACCESS()** dynamically changes the audit level setting and overrides the AUDIT\_DEFAULT originally set in FRPCFG member

270

Use the new UPDATE RM command to dynamically enable the RM address space to use the repository, or change the audit access level that was originally specified in the FRPCFG member.

Notice that the command must be issued with TYPE(REPO) and REPOTYPE(IMSRSC) to indicate that a repository resource type and an "IMSRSC" type of repository is being updated. These are the only valid values for these parameters.

To dynamically enable RM for repository usage, specify SET(REPO(Y)) and command processing will reread and re-process the REPOSITORY section within the CSLRIxxx member. This is why it is important to add repository definitions to the CSLRIxxx member before issuing this command. Also during command processing, RM registers to the Repository Server (RS) if RM is not already registered. RM connects to the repository name specified in the REPOSITORY section of the CSLRIxxx member. If the command is successful at the command master RM, the command master RM communicates the changes to other active RMs in the IMSplex. All RMs in the IMSplex will have the same repository settings. If RM is defined to use the resource structure, the command master RM will update the resource structure with the repository name and repository type that it is connected to. Subsequent RMs that are restarted after the change will ensure that they are connected to the same repository name and repository type as read from the resource structure.

On the other hand, if RM repository usage is already enabled, you can dynamically disable it by specifying SET(REPO(N)). The CSLRIxxx member is not reread/re-processed in this case like it is when SET(REPO(Y)) is specified. Therefore, as part of the repository disabling process, you can remove the repository definitions from CSLRIxxx either before or after UPDATE RM is issued with SET(REPO(N)). **Note that if the repository definitions specified on the REPOSITORY= statement are still present in CSLRIxxx, any RMs that start after the UPDATE RM ... SET(REPO(N)) command is issued will re-connect to the repository during the RM startup.**

The AUDITACCESS() parameter allows you to dynamically change the audit level settings and override what was originally specified on the AUDIT\_DEFAULT parameter in the FRPCFG member.



## ***New UPDATE RM Command***

```
UPDATE RM TYPE(REPO) REPOTYPE(IMSRSC)
SET(REPO(Y|N) AUDITACCESS())
```

- Valid values for **AUDITACCESS()** are:
  - **NOAUDIT** for no auditing of member access
  - **SECURITY** for auditing security failures only
  - **UPDATE** for auditing member access with update intent
  - **READ** for auditing member access with read and update intent
  - **SYSTEMREAD** for auditing member access with system-level read, read, or update intent
    - A “system-level read” is a read that occurs as part of the process for updating a resource

Here are the valid values for the AUDITACCESS() parameter. Notice that they are in ascending order of the amount of auditing that will occur when different types of member access are attempted.

## New QUERY RM Command

```
QUERY RM TYPE(REPO) SHOW( )
```

- Use command to determine whether or not RM is enabled for repository usage and to see information about repositories being managed by RM
- SHOW(ATTRIB) returns repository audit access level
  - Indicates the type/level of member access auditing -- defined in FRPCFG member, in CSLRlxxx (RM initialization) member, or with UPDATE RM command
- SHOW(STATUS) returns RM status information:
  - CONNECTED = RM connected to repository
  - DISCONNECTED = RM disconnected from repository
  - SPARERECOV = repository spare recovery in progress
  - SPARERCVERR = repository spare recovery error
  - NOTAVAIL = repository unavailable
- SHOW(ALL) returns all RM attribute and status information

272

Issue the QUERY RM command to determine whether an RM address space is enabled for repository usage, or to view RM's audit access level and/or status. The RM can have one of the statuses shown on this slide.

## UPDATE IMS Command Enhancements

```
UPDATE IMS SET(LCLPARM( ))
```

- Use command to dynamically enable IMS to use the repository and disable automatic export
- LCLPARM(REPO(Y) REPOTYPE(IMSRSC))
  - Enables IMS to use the repository
  - RM must be enabled to use the repository which can be accomplished with the UPDATE RM command (otherwise, UPDATE IMS will fail)
  - Ensure that DFSDFxxx is updated with valid repository definitions before this command is issued
    - DFSDFxxx will be read during UPDATE IMS command processing (if any errors exist in member, UPDATE IMS will fail)
      - Only <SECTION=REPOSITORY> definitions

273

Dynamically enable an IMS system to use the repository by issuing the UPDATE IMS command, or use it to disable the automatic export capability (covered on next slide). **Note that before you issue this command to dynamically enable IMS for repository usage, you must add repository definitions to the DFSDFxxx member and RM must already be repository-enabled.**

## **UPDATE IMS Command Enhancements**

```
UPDATE IMS SET(LCLPARM( ))
```

- **LCLPARM(AUTOEXPORT(N))**
  - Disables automatic export to the system RDDS
  - Use after migration to repository to eliminate unnecessary processing overhead associated with automatic export
  - Reactivation of automatic export requires an IMS coldstart with AUTOEXPORT defined in DFSDFxxx

274

To dynamically disable automatic export for an IMS system, issue the UPDATE IMS command with LCLPARM(AUTOEXPORT(N)) included. This is a step that would be taken after migration to the repository has been completed to reduce I/O overhead that occurs with autoexport.

Note: once automatic export has been disabled with this command, a coldstart is required to re-enable it. This is due to the possibility that the RDDSs may not exist any longer after migration to the repository has been completed and therefore would not be available for autoexport if it was re-enabled dynamically.

## QUERY IMS Command Enhancements

```
QUERY IMS TYPE( ) SHOW( )
```

- Use this command to determine whether IMS is enabled for repository usage and/or automatic export
- TYPE(LCLPARM) indicates that local IMS information will be displayed
  - SHOW(GLOBAL) cannot be specified with this parameter
- SHOW(REPO) shows whether the IMS is enabled to use the repository + attributes of the repository
- SHOW(AUTOEXPORT) shows whether automatic export is enabled for the IMS
- SHOW(ALL) and SHOW(LOCAL) will display the same local parameter information and have been enhanced to include the new SHOW() parameters listed above

275

To determine whether an IMS system is enabled for repository usage, issue the QUERY IMS command. You can also use this command to determine whether automatic export is enabled for an IMS system. These enhancements were added to existing QUERY IMS functionality – essentially, to display the information about the possible actions that were added with the UPDATE IMS command (dynamically enabling repository usage and/or automatic export).

## QUERY Command Enhancements

```
QUERY rsc-type | desc-type NAME( ) SHOW( )
```

- New SHOW() parameters added to QUERY
- Use this command to display generic and IMS-specific definitions
  - SHOW(DEFN) returns both repository and local IMS resource definitions
  - SHOW(DEFN,GLOBAL) returns only repository IMS resource definitions
  - SHOW(DEFN,LOCAL) returns only local IMS resource definitions
  - SHOW(IMSID) returns all IMSIDs that have the specified resource defined
  - SHOW(DEFN,IMSID) returns all IMSIDs that have the specified resource defined + a list of the repository resource definitions + any IMS-specific definitions
- Command output will display an L prefix for the columns containing the local IMS information

276

The QUERY command has been enhanced to display IMS resource/descriptor definitions (names and attribute values) in the repository and also at each IMS system. In addition, QUERY can now display specific IMSIDs that have the resource names included in the command. Note that this slide only shows the enhancement made to the QUERY command, not the full possible syntax for the command.

## EXPORT Command Enhancements

```
EXPORT DEFN TARGET(REPO) TYPE() NAME()
STARTTIME() ENDTIME() SET(IMSID()) OPTION()
```

- [Use command to harden runtime resource adds/changes to repository](#)
- [Use command to populate an empty repository for the first time, discussed later](#)
- Writes an IMS system's [runtime](#) resources/descriptors definitions to the repository, where they will kept as [stored](#) resource definitions
- TYPE() defines the resource type
  - [ALL](#), ALLDESC, ALLRSC, DB, DBDESC, PGM, PGMDESC, RTC, RTCDESC, TRAN, TRANDESC
- NAME() defines the names of the resources to export
  - NAME(\*) is the default
- STARTTIME() indicates the time after which all created/modified resources will be exported
- ENDTIME() indicates the cut-off time for when created/modified resources be exported

Local time in yyyy.ddd  
hh:mm:ss:th format (only  
yyyy.ddd are required)

277

Issue the EXPORT command when resources and/or descriptors have been either created or updated, and they need to be hardened to the repository. If hardening changes to offline stored definitions is part of your change management process, use the EXPORT command when definitional changes occur or at regular intervals during operations.

The EXPORT command is processed by a single command master IMS (the benefits of this are discussed in a few slides) and will write valid specified resources/descriptors to the repository.

The EXPORT command can also use data included in QUERY command output. If QUERY is issued with the OPTION(TIMESTAMP) parameter included, you can determine the exact time that a resource was created or updated (see the TimeCreate and TimeUpdate column header output). You are then able to use those timestamp values for the EXPORT STARTTIME() and/or ENDTIME() parameters. The STARTTIME() and ENDTIME() parameters can be as specific as tenths and hundredths of a second, and matches the timestamp granularity displayed in QUERY SHOW(TIMESTAMP) command output, so copying the exact values is facilitated in this way. Note that the STARTTIME() and ENDTIME() parameters are optional.

## **EXPORT Command Enhancements**

```
EXPORT DEFN TARGET(REPO) TYPE( ) NAME( )  
STARTTIME( ) ENDTIME( ) SET(IMSID( )) OPTION( )
```

- **SET(IMSID())** specifies one or more IMSIDs whose resource lists will be updated in the repository as a result of EXPORT being issued
  - Wildcards \* and % supported
    - Examples: IMS\*, IMS%A
    - If there are no IMS resource lists in the repository, EXPORT will fail
  - The specified IMSID does not need to be active in the IMSplex
    - EXPORT will create an IMS resource list for this IMSID, to be read when IMS starts
  - If SET(IMSID()) is omitted, the default is the IMSID of the command master
    - Command master can be selected using the ROUTE parameter

278

Indicate which IMS resource lists in the repository should be exported to by listing them on the SET(IMSID()) parameter. This is an optional parameter and if omitted, the command master IMSID's runtime resource definitions will be exported to its IMS own resource list. You can control which IMS is selected as command master by using the ROUTE capability of the OM interface from which you are entering the command. Wildcards are supported and note that the EXPORT command will fail if there are no IMS resource lists in the repository.

A benefit of using repository DRD is the ability to update the IMS resource list of an inactive IMS (or create one for it if it does not exist). When the IMS restarts, it will read these stored definitions that were altered while it was inactive. This is true for all types of restart (coldstart, warmstart and emergency restart) and details about how this occurs are covered later in the session.



## ***EXPORT Command Enhancements***

```
EXPORT DEFN TARGET(REPO) TYPE( ) NAME( )  
STARTTIME( ) ENDTIME( ) SET( IMSID( ) ) OPTION( )
```

- **OPTION()** controls how much command output is displayed and what specific resources are exported
  - OPTION(ALLRSP) will return a line of output for each successfully exported resource/descriptor in the command response
  - OPTION(CHANGESONLY) specifies that only resources/descriptors created or updated since the last EXPORT will be exported to the repository
    - This option is a new capability that did not exist with RDDS DRD
    - OPTION(ALLRSP) automatically included

279

When issuing the EXPORT command, you can optionally display a line of output for each successfully exported resource, as well as optionally export only the resources/descriptors that had definitional changes since the last EXPORT command was issued. Automatic export is not possible when using DRD with the repository, but issuing the EXPORT command with OPTION(CHANGESONLY) at regular intervals allows you to ensure that definitional changes are captured and hardened to the repository.

## New **DELETE DEFN** Command

```
DELETE DEFN TARGET(REPO) TYPE(  
NAME() FOR(IMSID()) OPTION()
```

- Deletes stored resource definitions from the repository, and is processed by one command master IMS
- Use this command to harden runtime definition deletes to the repository
- TYPE() specifies a **single** resource/descriptor type
  - DB, DBDESC, PGM, PGMDESC, RTC, RTCDESC, TRAN, TRANDESC
- NAME() defines the names of the resources/descriptors to delete
  - Wildcard supported (not the default like it is with DELETE command)

280

Issue the DELETE DEFN command to delete stored resource definitions from one or more IMS resource lists contained in the repository. If runtime resource definitions have been deleted from an online IMS system with the DELETE command, these deletes can be hardened to the repository with the DELETE DEFN command. In this situation, take care to specify the same resources or descriptors in this command whose runtime definitions were deleted from the online system. Note that only one resource or descriptor type can be specified and therefore, the command may need to be issued multiple times if multiple resource/descriptor types were deleted from the online system.

## New **DELETE DEFN** Command

```
DELETE DEFN TARGET(REPO) TYPE( )  
NAME( ) FOR(IMSID( )) OPTION( )
```

- **FOR(IMSID())** indicates the IMS resource list(s) the resources or descriptors are deleted from within the repository
  - Can specify a single IMSID, or a list of multiple IMSIDs (the IMS must be defined to RM to use the repository and can be either active or inactive)
  - Usage example:

```
QUERY PGM NAME(PGM1) SHOW(WORK)
UPDATE PGM NAME(PGM1) STOP(SCHD) } Routed to IMS1 and IMS2
DELETE PGM NAME(PGM1)
DELETE DEFN TARGET(REPO) TYPE(PGM) NAME(PGM1)
FOR(IMSID(IMS1,IMS2))
```

281

To delete stored definitions from specific IMS resource lists within the repository, specify the IMSIDs associated with the IMS resource lists with the **FOR(IMSID())** parameter. As mentioned on the previous slide, it is appropriate to issue this command when you want to harden runtime definition deletes to the repository. Before deleting a runtime resource definition, it is a DRD best practice to first query the resource to determine whether work in progress exists. If there is not, the resource should then be stopped before attempting to delete it.

An example of this is shown here, when a program is first queried to determine whether any work in progress exists for it. Then, the scheduling is stopped for the program to prevent any new work in progress from occurring. The example continues to show that the program is deleted from two online systems with the **DELETE** command, then deleted from the IMS resource lists associated with these two systems in the offline repository with the **DELETE DEFN** command.

## ***New DELETE DEFN Command***

```
DELETE DEFN TARGET(REPO) TYPE()  
NAME() FOR(IMSID()) OPTION()
```

- FOR(IMSID()) (...cont'd)
  - Wildcards \* and % supported
- OPTION(ALLRSP) ensures that the command response displays a line of output for each resource/descriptor processed by command
  - Only valid with NAME(\*)

Wildcard support exists for the FOR(IMSID()) parameter. When issuing the DELETE DEFN command with NAME(\*), you can ensure that a line of output is displayed for each resource/descriptor that was processed.

## **IMPORT Command Enhancements**

```
IMPORT DEFN SOURCE ( )  
TYPE ( ) NAME ( ) OPTION ( ) SCOPE ( )
```

- Reads resource/descriptor stored resource definitions from the repository into the IMS system, where they become runtime resource definitions
- Use this command to percolate definitional changes made to the offline repository to 1+ running IMS systems, for example:
  - Coldstart an IMS with no resources defined, issue IMPORT to read in its definitions
  - Make changes to repository then roll them out to 1+ running IMS systems
- New SOURCE(REPO) keyword to read from repository
- TYPE() defines the resource/descriptor type
  - ALL, ALLDESC, ALLRSC ,DB, DBDESC, PGM, PGMDESC, RTC, RTCDESC, TRAN, TRANDESC
- NAME() defines the names of the resources to import
  - NAME(\*) is the default

283

The IMPORT command reads stored definitions that exist in the repository into running IMS systems. This command can be used if an IMS is coldstarted with no resources defined to populate the control region with runtime resource definitions. Or if changes were made to the repository offline and you'd like to roll the changes to the systems in IMSplex, the IMPORT command can be used to accomplish this. An example of when this scenario is when the "RDDS to Repository", or CSLURP10 utility (introduced in part one of this session) is used to populate a repository with definitions, which haven't been read into any IMS system yet.

Make sure that SOURCE(REPO) is specified, so the repository is the data set that is read, and indicate which resources should be imported using the other parameters shown on this slide.

## **IMPORT Command Enhancements**

```
IMPORT DEFN SOURCE( )  
TYPE( ) NAME( ) OPTION( ) SCOPE( )
```

- **OPTION()** controls how much command output is displayed and what specific resources are imported
  - **OPTION(ABORT)** will fail the **IMPORT** command if an error occurs while importing a resource/descriptor
  - **OPTION(ALLRSP)** will return a line of output for each successfully imported resource/descriptor in the command response (valid with **NAME(\*)**)
  - **OPTION(UPDATE)** will replace an existing resource with the one that exists in the RDDS or repository
    - Required if a resource already exists in a running IMS system, otherwise **IMPORT** will fail
    - Work in progress cannot exist for the resource that **IMPORT** is attempting to replace with the stored definition (recommendation: stop, then query the resource to determine whether it is currently in use)

284

You can control the output displayed by the **IMPORT** command by specifying the **OPTION()** parameter accordingly. Prior to IMS 12, the **OPTION(ABORT)** and **OPTION(ALLRSP)** were used with RDDS DRD import and are now also used with repository DRD import. **OPTION(ABORT)** will terminate command processing if an error occurs during import, and **OPTION(ALLRSP)** will return a line of output for each resource that was exported, and is valid if **NAME(\*)** was also specified in the command. If individual names are specified for the **NAME()** parameter, a line of output will be returned for each resource in this case as well.

A new **OPTION(UPDATE)** parameter has been added for **IMPORT**, which allows an existing runtime resource definition to be updated with a stored resource definition being imported from either the RDDS or the repository. We will elaborate more on this capability later in the session. **Note that this parameter is not the default and must be explicitly specified to update a resource in this way.**

## ***IMPORT Command Enhancements***

```
IMPORT DEFN SOURCE( )  
TYPE( ) NAME( ) OPTION( ) SCOPE( )
```

- **SCOPE()** is a new **IMPORT** parameter that indicates which **IMS** systems the **IMPORT** will apply to
  - This parameter is optional
  - **SCOPE(ALL)** will apply the **IMPORT** command to each active **IMS** in the **IMSplex**
    - Use care when specifying a **ROUTE** list since this will take precedence over **SCOPE(ALL)**, since **IMS** systems not specified on **ROUTE** will not receive the command
  - **SCOPE(ACTIVE)** means the same as **SCOPE(ALL)** in **IMS 12** but will have additional functionality in a future **IMS** release

285

When you issue an **IMPORT SOURCE(REPO)** command, you are able to indicate with the **SCOPE(ALL)** parameter that the import should be performed at each **IMS** system in the **IMSplex**.

You may be familiar with the **ROUTE** capability of the **OM API**, used to route commands to specific **IMS** systems. **ROUTE=ALL is recommended when SCOPE(ALL) is included.** If a **ROUTE** list is specified (other than **ROUTE=ALL**), the command is processed only by the **IMS** systems in the list that receive the command. Other **IMS** systems that have the resources defined but are not included in the **ROUTE** list will not receive the command and therefore will not be synchronized with the repository.

**SCOPE(ALL)** applies the import to the active **IMS** systems and is recommended to maintain synchronized definitions across the **IMSplex** that match the repository definitions. **SCOPE(ACTIVE)** means the same as **SCOPE(ALL)** in **IMS 12**.

## ADD Command

```
ADD REPOSITORY(repository-name)
REPDS1RID(primaryRID-name)
REPDS1RMD(primaryRMD-name)
REPDS2RID(secondaryRID-name)
REPDS2RMD(secondaryRMD-name)
REPDS3RID(NULL | spareRID-name)
REPDS3RMD(NULL | spareRMD-name)
AUTOOPEN(NO | YES)
SECURITYCLASS(NULL | securityclassname)
```

- REPOSITORY = name of the repository to be added
  - Can be up to 44 characters long
  - Mixed-case not supported, converted to upper-case
- REPDS1RID = primary repository index data set
- REPDS1RMD = primary repository member data set
- REPDS2RID = secondary repository index data set
- REPDS2RMD = secondary repository member data set

} required  
parameters

286

To define a user repository to the RS catalog repository, use the batch ADMIN ADD command using the syntax shown on this slide. Here, you must specify the user repository name as well as the names of the user repository primary/secondary index and member data sets. Note that the user repository name will be converted to upper-case if it is specified with anything else.



## ADD Command

```

ADD REPOSITORY(repository-name)
REPDS1RID(ds1_rid_dsname)
REPDS1RMD(ds1_rmd_dsname)
REPDS2RID(ds2_rid_dsname)
REPDS2RMD(ds2_rmd_dsname)
REPDS3RID(NULL | ds3_rid_dsname)
REPDS3RMD(NULL | ds3_rmd_dsname)
AUTOOPEN(NO | YES)
SECURITYCLASS(NULL | securityclassname)

```

- REPDS3RID = spare repository index data set
  - REPDS3RMD = spare repository member data set
  - AUTOOPEN = whether repository data sets are opened when the repository is started (YES, default) or when a user first connects (NO)
  - SECURITYCLASS = name of security class to be used for secured repository access
    - Must be 8 bytes long, left-aligned and padded with blanks if necessary
- } optional parameters

287

This slide shows the optional parameters for the batch ADMIN ADD command. Notice that you can optionally specify the spare repository index and member data set names here. If nothing is specified, there will be no spare as the default is null.

You can also control whether the repository data sets you are specifying with this command are opened when the repository is started (AUTOOPEN YES, which is the default) or when a user first connects to it (AUTOOPEN NO).

If you are going to be restricting access to the user repository, specify the name of the 8-byte security class that will be used to restrict access here. This will override the SAF\_CLASS= parameter value in the FRPCFG configuration member, if one was specified. Alternatively, if you wish to deactivate repository security – you can specify SECURITYCLASS(NULL) on the batch ADMIN ADD command to accomplish this. **More detail regarding security setup will be discussed later in the session, in the “Security Considerations” section.**

## UPDATE Command

```
UPDATE REPOSITORY(repository-name)
REPDS1RID(ds1_rid_dsname | NULL)
REPDS1RMD(ds1_rmd_dsname | NULL)
REPDS2RID(ds2_rid_dsname | NULL)
REPDS2RMD(ds2_rmd_dsname | NULL)
REPDS3RID(ds3_rid_dsname | NULL)
REPDS3RMD(ds3_rmd_dsname | NULL)
AUTOOPEN (YES | NO)
SECURITYCLASS(securityclassname | NULL)
```

- REPOSITORY = name of the repository to be updated
  - REPDSxRID = index data set name
  - REPDSxRMD = member data set name
  - AUTOOPEN = determines when data sets are opened
    - YES will open data sets when repository is started (default setting)
    - NO will open data sets when RM first connects to the repository
- } x = 1/2/3 for primary/secondary/spare

288

Use the Batch ADMIN *UPDATE* command to modify a user repository definition within the RS catalog repository **datasets** (specifically, to change the data sets, auto-open option or security class associated with a specific repository). The only required parameter for this command is the REPOSITORY parameter. The parameters associated with this command have the same meaning as they do when issued with the batch ADMIN *ADD* command.

Note that a user repository must be stopped before it can be updated. We will cover how to stop a repository later in the session.

## UPDATE Command

```
UPDATE REPOSITORY(repository-name)
REPDS1RID(ds1_rid_dsname | NULL)
REPDS1RMD(ds1_rmd_dsname | NULL)
REPDS2RID(ds2_rid_dsname | NULL)
REPDS2RMD(ds2_rmd_dsname | NULL)
REPDS3RID(ds3_rid_dsname | NULL)
REPDS3RMD(ds3_rmd_dsname | NULL)
AUTOOPEN (YES | NO)
SECURITYCLASS(securityclassname | NULL)
```

- SECURITYCLASS = name of security class to be used for secured repository access
  - Must be 8 bytes long, left-aligned and padded with blanks if necessary
  - Default is NULL if not specified (no repository security)

## ***RENAME Command***

```
RENAME REPOSITORY(repository-name)  
      REPOSITORYNEW(repository-newname)
```

- REPOSITORY (repository-name) = existing repository name to be changed
- REPOSITORYNEW (repository-newname) = name to replace existing repository name

Use the batch *ADMIN RENAME* command to rename a user repository name defined within the RS catalog repository.

## ***DELETE Command***

```
DELETE REPOSITORY(repository-name)
```

- REPOSITORY = name of the user repository to be deleted from the RS catalog repository
- Physical data sets are not deleted
  - Use the z/OS Access Method Services (IDCAMS) utility or a similar method after issuing batch ADMIN *DELETE* command

Use the batch ADMIN DELETE command to remove a user repository from the RS catalog repository. Note that once you have deleted the user repository, you must delete its associated physical data sets in a separate step using the IDCAMS utility or similar method.

## ***DSCHANGE Command***

```
DSCHANGE REPOSITORY(repository-name)
RDS(1 | 2 | 3) ACTION(SPARE | DISCARD)
```

- REPOSITORY = name of the repository whose data sets are to be changed
- RDS = number representing which repository data set pair will be changed
  - 1 for RDS1
  - 2 for RDS2
  - 3 for RDS3
- ACTION = what disposition the specified RDS will be changed to
  - SPARE will change the RDS to the spare data set within repository
    - Can only be executed against an RDS with DISCARD status
    - Both data sets in the RDS pair must be empty
  - DISCARD will prepare an RDS to be replaced with a newly defined data set
    - Repository must be stopped if primary or secondary RDS (not required for spare)
    - An RDS must have this disposition in order to be replaced

292

There are certain times when it is appropriate to change the disposition of a repository data set (RDS) to either SPARE or DISCARD, which can be done using the batch ADMIN *DSCHANGE* command.

If an error occurs on the primary or secondary repository data set (RDS), recovery is driven by the Repository Server automatically if a spare RDS is present. Once this occurs, the user must then allocate and define a new RDS to replace the one that had the failure. This new RDS should be designated as the spare RDS, which can be done using the batch ADMIN *DSCHANGE* command with *ACTION(SPARE)* specified. This will change the disposition of a repository data set pair (RDS) to SPARE. **More detail about recovery in the event of an RDS error will be covered later in the session.**

If you want to replace an existing RDS with a different RDS, you must first stop the repository and change the disposition/status of the RDS to DISCARD. This can be done by issuing the batch ADMIN *DSCHANGE* command with *ACTION(DISCARD)* specified. Once an RDS has a disposition of DISCARD, it can be replaced with a newly defined data set.

## ***LIST Command***

```
LIST REPOSITORY(repository-name) | STATUS
```

- REPOSITORY = name of the repository whose details will be displayed
- STATUS = details associated with all of the user repositories defined to the RS catalog repository
  - User repository name
  - User repository status
  - Date of last update
  - USERID that last updated user repository

Use the batch ADMIN *LIST* command to display the details of a single repository, including its status, or display all user repository names. The information that will be shown if a user repository is specified for the REPOSITORY parameter is listed on this slide. Note that you can also issue the command as just *LIST STATUS* (without specifying the REPOSITORY parameter) to see only a list of user repository names defined to the RS catalog repository.

## Offline Repository Management - Examples

```
* ----- *
LIST REPOSITORY(IMSRSC_REPOSITORY)
Repository Name . : IMSRSC_REPOSITORY

Last updated date/time : 2010/07/27 00:52:46  USRT001
Status . . . . . : STOPPED
Auto-open . . . . . : YES
Security Class . . . . : NOT DEFINED

Repository Data Set pair 1
Index (RID) . . : IMSTESTS.FRP1.IMSPRI.RID
Member (RMD) . . : IMSTESTS.FRP1.IMSPRI.RMD
Status . . . . . : COPY1

Repository Data Set pair 2
Index (RID) . . : IMSTESTS.FRP1.IMSSEC.RID
Member (RMD) . . : IMSTESTS.FRP1.IMSSEC.RMD
Status . . . . . : COPY2

Repository Data Set pair 3
Index (RID) . . : IMSTESTS.FRP1.IMSSPR.RID
Member (RMD) . . : IMSTESTS.FRP1.IMSSPR.RMD
Status . . . . . : SPARE

FRP4750I - LIST command processing completed successfully
* -----END-OF-JOB----- *
```

294

This slide shows an example of the output that would be received from a batch ADMIN LIST command when a user repository name is specified. Notice that you can see detailed information about this user repository, such as its status, its auto-open value, and the different RDSs that it contains as well as their statuses (dispositions).



## START Command

```
START REPOSITORY(repository-name) OPEN(YES | NO)
MAXWAIT(seconds,IGNORE | CONTINUE | ABORT)
```

- Use this command after adding a repository to the RS catalog repository with *ADD* command
  - REPOSITORY = name of repository to be started
  - OPEN = whether or not the repository data sets should be opened/allocated when repository is started
    - YES means that the data sets will be opened when it is started
    - NO means that data sets will be opened when a user first attempts to connect (unless AUTOOPEN=YES has been specified on a batch ADMIN ADD or UPDATE command for this repository)
  - MAXWAIT = how many seconds to wait for start command to complete and which action to take when time elapsed
    - Seconds
      - 0-9999 valid range
      - IGNORE will continue command processing and set rc=0
      - CONTINUE will continue command processing and set rc=4
      - ABORT will terminate command processing and set rc=8
    - MAXWAIT(5,CONTINUE) is the default
- } optional parameters

295

Use the batch ADMIN *START* command to start a specific user repository, for example after it has been defined to the RS catalog repository with the batch ADMIN *ADD* command. Note that with the optional OPEN() parameter, you can override the AUTOOPEN= parameter value that was originally specified when the repository was added to or last updated in the RS catalog repository (with batch ADMIN *ADD* or *UPDATE* commands, respectively). This parameter value indicates whether the user repository's RDSs will be open when it is started (with OPEN(YES)), or when a user first connects to it (with OPEN(NO)). **Note that you can only override the AUTOOPEN= parameter if it was originally specified as AUTOOPEN=NO.**

You can optionally include the MAXWAIT parameter to indicate how many seconds should elapse before a particular action that you also specify is taken. You can specify a wait time of up to 9999 seconds and opt to have the command continue processing with a return code of either 0 or 4, or opt to have it terminate with a return code of 8. By default, if 5 seconds has elapsed once this command has been issued, the command will continue processing and will give a return code of 4. The specific parameter values for the MAXWAIT() syntax are shown on this slide.

## STOP Command

```
STOP REPOSITORY(repository-name)
MAXWAIT(seconds, IGNORE | CONTINUE | ABORT)
```

- Use this command in preparation for updating user repository definitions within a RS catalog repository with *UPDATE* command
    - A stopped repository rejects connection attempts and is deallocated/closed by the Repository Server
  - REPOSITORY = name of user repository defined to the RS catalog repository to be stopped
  - MAXWAIT = how many seconds to wait for stop command to complete and which action to take when time elapsed
    - Seconds
      - 0-9999 valid range
      - IGNORE will continue command processing and set rc=0
      - CONTINUE will continue command processing and set rc=4
      - ABORT will terminate command processing and set rc=8
    - MAXWAIT(5,CONTINUE) is the default
- } optional parameter

296

Use the batch ADMIN *STOP* command to stop a specific user repository that is defined to the RS catalog repository. A stopped repository will reject user connection attempts. The command also results in the repository being closed and deallocated by the Repository Server. Note that this command has the same MAXWAIT() parameter value as the batch ADMIN *START* command.

Much like a /DBR command that prevents programs and transactions from accessing a database, the batch ADMIN *STOP* command, when issued with MAXWAIT(xx,IGNORE or CONTINUE), can continue processing after xx seconds have elapsed. At this point, the command continues processing (just like the /DBR command) and a specific return code is received, determined by whether IGNORE (rc=0) or CONTINUE (rc=4) was specified. Of course, if ABORT was specified instead of IGNORE or CONTINUE, the command would terminate processing and a rc=8 would be received when xx seconds has elapsed.

```
//FRPBAT EXEC PGM=FRPBATCH,PARM='XCFGROUP=FRP2PLEX'  
//SYSPRINT DD SYSOUT=*  
//SYSIN DD *  
/*  
ADD REPOSITORY(IMS_REPOS) -  
  REPDSN1RID(IMSTESTS.REPO.IMPRI.RID) -  
  REPDSN1RMD(IMSTESTS.REPO.IMPRI.RMD) -  
  REPDSN2RID(IMSTESTS.REPO.IMSSEC.RID) -  
  REPDSN2RMD(IMSTESTS.REPO.IMSSEC.RMD) -  
  AUTOOPEN(NO)  
/*  
START REPOSITORY(IMS_REPOS) MAXWAIT(30,CONTINUE)  
/*  
LIST REPOSITORY(IMS_REPOS)  
/*  
STOP REPOSITORY(IMS_REPOS) MAXWAIT(30,CONTINUE)  
/*  
RENAME REPOSITORY(IMS_REPOS) REPOSITORYNEW(IMS_PROD_REPOS)  
/*  
UPDATE REPOSITORY (IMS_PROD_REPOS) -  
  REPDS1RID(IMSTESTS.PRODREPO.IMPRI.RID) -  
  REPDS1RMD(IMSTESTS.PRODREPO.IMPRI.RMD) -  
  AUTOOPEN(NO)  
/*  
DELETE REPOSITORY(IMS_PROD_REPOS)
```

**Example JCL  
for Batch ADMIN  
Commands**

297

This slide shows an example of the batch ADMIN utility which issues the commands just discussed.

## ADMIN Command

```
F reposervername,ADMIN
DSCHANGE(repositoryname, s|D, 1|2|3) ←
DISPLAY(repositoryname | <blank>)
START(repositoryname)
STOP(repositoryname)
```

- ADMIN command performs repository administrative tasks
- DSCHANGE = change the disposition of a repository data set pair
  - Specify the name of the repository that contains the target data set pair
  - New disposition specified by
    - S for a new disposition of SPARE
    - D for a new disposition of DISCARD
  - The target repository data set pair is indicated by
    - 1 for RDS1
    - 2 for RDS2
    - 3 for RDS3

298

Use the z/OS modify interface ADMIN command to perform various administrative tasks. Use it with the DSCHANGE parameter to change the disposition of a data set contained in a repository to DISCARD or SPARE status. The circumstances under which this would be appropriate were explained in the previous section when the batch ADMIN *DSCHANGE* command was covered and will also be discussed later in the session when repository data set recovery is covered.

## ADMIN Command

```
F reposervername,ADMIN
DSCHANGE(repositoryname, S|D, 1|2|3)
DISPLAY(repositoryname | <blank>) ←
START(repositoryname)
STOP(repositoryname)
```

- **DISPLAY** = show the user repository names defined to the RS catalog repository with the respective data set names they contain
  - Specify the name of the repository that contains primary, secondary and spare data set pairs
  - If left blank, only a list of repository names are shown
  - Similar to batch ADMIN *LIST STATUS* command

Use the ADMIN command with the DISPLAY parameter to display a list of user repository names defined to the RS catalog repository. If a repository name is specified for the DISPLAY() parameter, other details such as the RDS names and statuses will be shown. This command is similar to the batch ADMIN *LIST STATUS* command.

## ADMIN Command - Example

F REPO1,ADMIN DISPLAY(IMSRSC\_REPOSITORY)

```

/*****/
/* Display the IMSRSC_REPOSITORY via the ADMIN cmd */
/* FRP2100I - ADMIN DISPLAY repository IMSRSC_REPOSITORY */
/*      - Last updated date/time : USRT001 */
/*      - Status . . . . . : OPEN */
/*      - Auto-open . . . . . : YES */
/*      - Security Class . . . . : NOT DEFINED */
/* FRP2101I - ADMIN DISPLAY repository RDS1: */
/*      - Index (RID) . . : IMSTESTS.FRP1.IMSPRI.RID */
/*      - Member (RMD) . : IMSTESTS.FRP1.IMSPRI.RMD */
/*      - Status . . . . : COPY1 */
/* FRP2101I - ADMIN DISPLAY repository RDS2: */
/*      - Index (RID) . . : IMSTESTS.FRP1.IMSSEC.RID */
/*      - Member (RMD) . : IMSTESTS.FRP1.IMSSEC.RMD */
/*      - Status . . . . : COPY2 */
/* FRP2101I - ADMIN DISPLAY repository RDS3: */
/*      - Index (RID) . . : */
/*      - Member (RMD) . : */
/*      - Status . . . . : NONE
/*****/

```

300

This slide shows an example of the output that would be received from a z/OS modify interface ADMIN,DISPLAY command when a user repository name is specified. Notice that you can see detailed information about this user repository, such as its status, its auto-open value, and the different RDSs that it contains as well as their statuses (dispositions).

## ADMIN Command

```
F reposervername,ADMIN
DSCHANGE(repositoryname, S|D, 1|2|3)
DISPLAY(repositoryname | <blank>)
START(repositoryname) ←
STOP(repositoryname) ←
```

- START = start the specified repository
- STOP = stop the specified repository

Use the ADMIN command with the START or STOP parameters to start and stop a repository, respectively.

## **AUDIT Command**

```
F reposervername,AUDIT LEVEL(NONE|HIGH) | RESTART
```

- AUDIT command dynamically changes the AUDIT\_LEVEL setting originally specified in FRPCFG member
- LEVEL determines whether or not repository log records are written to the log
  - NONE deactivates writing log records
  - HIGH activates writing log records
- RESTART resumes audit logging after logging was suspended due to an error when Repository Server was initializing and trying to connect to the log stream
  - Used with AUDIT\_FAIL=CONTINUE in FRPCFG member (indicates that the Repository Server will continue starting despite an error connecting to the log stream, but logging is suspended)

302

Use the AUDIT command to dynamically change the audit level setting specified on the AUDIT\_LEVEL parameter in the FRPCFG configuration member. To activate the writing of log records, specify LEVEL(HIGH) and to deactivate, specify LEVEL(NONE).

When the Repository Server is starting and attempting to connect to the log stream, an error can occur. Depending on what value you specified for AUDIT\_FAIL in the FRPCFG member, the Repository Server can either continue starting or can terminate. As covered in Part 1 of this session, if you specify AUDIT\_FAIL=CONTINUE, logging will be suspended in the event that the Repository Server encounters an error while attempting to connect to the log stream. You can later activate the logging of records by issuing the z/OS modify interface AUDIT command with RESTART included.



## ***SECURITY Command***

```
F reposervername,SECURITY REFRESH
```

- SECURITY command refreshes in-storage RACF profiles when changes have been made to them
  - FRPCFG member is not re-read
  - Repository definitions are not re-read
- Example

```
F REPOSVR1,SECURITY REFRESH
```

```
FRP2105I - In-core user security profiles refreshed
```

For repository security, if you need to make changes to your RACF (or SAF equivalent) definitions, they will only be active if you refresh the RACF in-storage profiles. Use the z/OS modify interface *SECURITY* command to accomplish this. The command will refresh the RACF profiles contained in storage to reflect the updated profile definitions. More detail on when this command should be issued will be discussed later in the session in the “Security Considerations” section.

## ***SHUTDOWN Command***

```
F reposervername, SHUTDOWN ALL
```

- SHUTDOWN command performs shutdown for a specified Repository Server or all Repository Server address spaces
- ALL parameter is optional
  - If included, all Repository Servers within the same XCF group as the Repository Server specified in the command are shut down
  - If not included, only the individual Repository Server will be shut down
    - If the specified Repository Server is the master, one of the existing subordinate servers will become the master
  - Repository Server name specified on command can be either master or subordinate

304

To shut down one or more Repository Server(s), issue the z/OS modify interface *SHUTDOWN* command. Including the optional ALL keyword will target all Repository Servers contained within the same XCF group, whereas omitting this keyword will just target the specified Repository Server for shutdown. If the master Repository Server is shutdown, one of the subordinate servers will become the new master.

## ***STOP Command***

**P** reposervername

- **STOP** command will stop and shut down a single specified Repository Server
  - If the specified repository server is the master, one of the existing subordinate servers will become the master

The z/OS modify interface *STOP* command is also available to shut down a single Repository Server. Here again, if the master Repository Server is shutdown, one of the subordinate servers will become the new master.