

IMS Connect Command Enhancements

New Dynamic Capabilities

- IMS Connect customers can now add new ports and IMS datastore connections without having to restart IMS Connect
- Use new type-2 commands
- Benefit: provide higher availability by eliminating the need to restart IMS Connect to activate definitional changes

The new type-2 CREATE commands are targeted for all IMS Connect customers. The CREATE commands are intended to be used to dynamically create definitions that may not have been known or needed during the creation of the static definitions.

Prerequisites

- **Software requirements**
 - Structured Call Interface (SCI)
 - Operations Manager (OM)

- **Hardware requirements**
 - Same as IMS 13

Type-2 Command – CREATE IMSCON TYPE(PORT)

```
CREATE IMSCON TYPE(PORT)
        NAME(portnum1, portnum2, ...)
        LIKE(portnum_model)
        SET(attribute1, attribute2, ...)
```

- Equivalent to defining TCPIP (regular port) or ODACCESS (DRDA port) statement in HWSCFGxx
- Parameters
 - NAME(*portnum*) specifies port number
 - LIKE(*portnum_model*) allows an existing port to be used as a model and is optional
 - SET(*attribute*) explicitly defines settings (optional)
 - PORTTYPE(REG|DRDA) specifies a regular or DRDA type of port
 - ➔ **DEFAULT** – “Regular” is equivalent to defining PORT sub-statement in TCPIP statement
 - TCPIP=(PORT=(ID=, KEEPAV=, EDIT=), ...)
 - “DRDA” is equivalent to defining DRDAPORT sub-statement in ODACCESS statement
 - ODACCESS=(DRDAPORT=(ID=, KEEPAV=, PORTTMOT=), ...)

Users can now dynamically create an IMS Connect port using the type-2 CREATE IMSCON TYPE(PORT) command.

The port number is specified on the NAME() parameter and the remaining attributes are defined with SET().

An existing port can also be referenced on the LIKE() parameter to set attribute values, for example the port type.

The type can be set to either regular or DRDA, whose equivalents in the IMS Connect Configuration member (HWSCFGxx) are shown here.

If you compare SET() parameters with those specified HWSCFGxx, you will find that the only SET() parameter without an equivalent is SET(PORTTYPE()), since the port type is specified in its own respective statement within the member (TCPIP for regular and ODACCESS for DRDA).

Type-2 Command – CREATE IMSCON TYPE(PORT)

```
CREATE IMSCON TYPE(PORT)
      NAME(portnum1, portnum2, ...)
      LIKE(portnum_model)
      SET(attribute1, attribute2, ...)
```

- SET(*attribute*) explicitly defines settings (optional)
 - KEEPAV() sets the “keep alive” value and defaults to 0 (in which case the TCP/IP stack’s setting is used)
 - EDITRTN() specifies a port edit exit routine and is only valid for regular ports
 - PORTTMOT() sets the timeout value for a DRDA port
 - Default is 6000 hundredths of a second (1 minute)

The SET() parameters also allow a user to define a keep alive value for a port. For a regular port, the Port Edit Exit routine can also be specified here, and for a DRDA port, the timeout value can be set.

Examples/Equivalents - CREATE IMSCON TYPE(PORT)

- Regular port

- New type-2 command

```
CREATE IMSCON TYPE (PORT) NAME (8888)  
SET (EDITRTN (PORTEDX0) , KEEPAV (120) )
```

- Equivalent configuration statement

```
PORT=(ID=8888, KEEPAV=120, EDIT=PORTEDX0)
```

- DRDA port

- New type-2 command

```
CREATE IMSCON TYPE (PORT) NAME (7777)  
SET (PORTTYPE (DRDA) , KEEPAV (120) , PORTTMOT (3000) )
```

- Equivalent configuration statement

```
DRDAPORT=(ID=7777, KEEPAV=120, PORTTMOT=3000)
```

In the example showing the dynamic creation of a regular port, note that the SET(PORTTYPE()) parameter was omitted since the default setting is a regular port type.

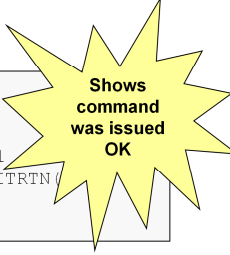
Example Output – CREATE IMSCON TYPE(PORT)**Command input**

```
CREATE IMSCON TYPE(PORT) NAME(8888)
SET(EDITRTN(PORTEDX0),KEEPAV(120))
```

Command output

```
PLEX1                IMS Single Point of Control
Command ==>


----- Plex . . PLEX1 Route . . HWS1
Response for: CREATE IMSCON TYPE(PORT) NAME(8888) SET(EDITRTN(
Port      MbrName      CC
8888     HWS1          0
```



Shows
command
was issued
OK

Console output of command result

```
HWSS0790I LISTENING ON PORT=8888      STARTED; M=SDOT
```



Shows
port was
created OK

05 System Part 2: 35

Here, we show an example of using the type-2 CREATE IMSCON command to dynamically create a port. The commands are asynchronous and note that a completion code (CC) of 0 shown on the TSO SPOC output screen indicates that the command was submitted to IMS Connect. When the port is successfully dynamically created, an HWSS0790I message is output to the console. The M=SDOT shown represents the module code that issued the message (M= is included in all IMS Connect messages).

In the TSO SPOC command input example, note that the command was routed to the IMS Connect named HWS1. If all IMS Connects in the IMSplex are sharing the same port, the user could create the port definition for each IMS Connect by leaving the Route field blank so that each IMS Connect receives the command. In the TSO SPOC output shown, the port is dynamically created for the HWS1 IMS Connect.

Type-2 Command – CREATE IMSCON TYPE(DATASTORE)

```
CREATE IMSCON TYPE (DATASTORE)
      NAME (datastore_name)
      LIKE (datastore_model)
      SET (attribute1, attribute2, ...)
```

- Equivalent to defining the DATASTORE statement in HWSCFGxx
 - DATASTORE=(APPL=, ACKTO=, CM0ATOQ=, DRU=, GROUP=, ID=, MAXI=, MEMBER=, OAAV=, RRNAME=, SMEMBER=, TMEMBER=)
- Parameters
 - NAME(*datastore_name*) specifies datastore name
 - LIKE(*datastore_model*) allows an existing datastore to be used as a model and is optional
 - SET(*attribute*) explicitly defines settings (optional unless otherwise indicated)
 - ACKTO() designates ACK timeout value (default is 120 seconds)
 - APPL() sets TCP/IP application name defined to RACF in PTKTDATA statement (**required** if using PassTicket and user message exits)

05 System Part 2: 36

- **NAME:** Required. Specifies the 1-8 character name of the IMS datastore. The name can consist of alphanumeric characters and must be unique within IMS Connect. This is equivalent to the ID parameter of the DATASTORE statement in the IMS Connect configuration member.
- **LIKE:** Specifies the existing IMS Connect DATASTORE resource to use as a model. The new resource is created with all the same attributes as the model. Attributes set explicitly by the CREATE command overrides the model attributes. Later changes to the model are not propagated to resources that were created from it.
- **SET:** Specifies the attributes of the IMS datastore to be created. If the LIKE keyword is omitted, the required attributes must be specified and the optional attributes will take their default values. If the LIKE keyword is specified, the attributes that are specified overrides the model attributes. Note that for the SET() parameter, the only value that is required is for MEMBER(). If a LIKE() parameter is not specified, values for GROUP() and TMEMBER() are also required. The rest of the values for SET() have defaults.
- **SET(ACKTO):** Specifies a timeout interval for acknowledgements to OTMA for CM0 and CM1 output messages and for IMS to IMS transaction messages. The timeout value can be from 0 to 255 seconds. This parameter is optional and defaults to 120. If the timeout value is 0 or is not specified, the OTMA ACK timeout default value of 120 seconds is set. For IMS to IMS transaction messages, if an acknowledgement is not received by OTMA before the timeout interval expires, OTMA reroutes the transaction message to the timeout queue, DFS\$\$TOQ.
- **SET(APPL):** Specifies a 1-8 character alphanumeric TCP/IP APPL name defined to RACF in the PTKTDATA statement. This parameter is optional and defaults to blanks. If you are using PassTicket and user message exits, you must specify this APPL parameter.

Type-2 Command – CREATE IMSCON TYPE(DATASTORE)

```
CREATE IMSCON TYPE (DATASTORE)
      NAME (datastore_name)
      LIKE (datastore_model)
      SET(attribute1, attribute2, ...)
```

- SET(*attribute*)
 - CM0ATOQ() sets CM0 ACK timeout queue (defaults to blanks)
 - DRU() specifies DRU exit (defaults to DFSYDRU0)
 - GROUP() indicates XCF group name (**required** if LIKE not specified)
 - MAXI() sets OTMA input message flood control (default is 5000)
 - MEMBER() specifies IMS Connect XCF member name (**required**)
 - OAAV() sets OTMA ACEE aging value (default is 999999 seconds)
 - RRNAME() sets the alternate destination of client's reroute request (default is HWS\$DEF)
 - SMEMBER() specifies OTMA super member name (defaults to blanks)
 - TMEMBER() sets IMS XCF member name (**required** if LIKE not specified)

- SET(CM0ATOQ): Specifies name of the OTMA CM0 ACK timeout queue and overrides both the OTMA default value of DFS\$\$TOQ and any value set on the HWS statement of the IMS Connect configuration member.
- SET(DRU): Specifies name of the OTMA destination resolution user (DRU) exit that is passed to OTMA. Defaults to DFSYDRU0.
- SET(GROUP): Specifies a name of the z/OS cross-system coupling facility group for the IMS OTMA.
- SET(MAXI): Specifies the OTMA input message flood control value. The valid range is from 0-9999. If you do not specify a value, or specify a value of 0, the OTMA default value of 5000 is used. If you specify a value between 1-200, the OTMA minimum value of 200 is used. This parameter is optional and defaults to 5000.
- SET(MEMBER): Specifies an XCF member name that identifies IMS Connect in the XCF group specified by the GROUP parameter. Required.
- SET(OAAV): Defines the OTMA ACEE aging value, in seconds, for this IMS data store. When reached, OTMA refreshes the ACEE before it processes the next input message received from IMS Connect.
- SET(RRNAME): Specifies a name of the alternate destination of a client's reroute request.
- SET(SMEMBER): Specifies a name of the OTMA super member to which this IMS data store belongs. If specified, this value overrides the value of the SMEMBER parameter specified on the HWS statement in the IMS Connect configuration member.
- SET(TMEMBER): Specifies the XCF member name of the IMS that this IMS Connect communicates with in the XCF group.

Type-2 Command – CREATE IMSCON TYPE(DATASTORE)

```
CREATE IMSCON TYPE (DATASTORE)
      NAME (datastore_name)
      LIKE (datastore_model)
      SET (attribute1, attribute2, ...)
```

- SET(*attribute*)
 - OAAV() sets OTMA ACEE aging value (default is 999999 seconds)
 - RRNAME() sets the alternate destination of client's reroute request (default is HWS\$DEF)
 - SMEMBER() specifies OTMA super member name (defaults to blanks)
 - TMEMBER() sets IMS XCF member name (**required if LIKE not specified**)

- SET(OAAV): Defines the OTMA accessor environment element (ACEE) aging value, in seconds, for this IMS data store. When reached, OTMA refreshes the ACEE before it processes the next input message received from IMS Connect. Valid values are from 0-999999. If you specify 0, OTMA uses the default value of 999999. If you specify a value from 1-300, OTMA uses a value of 300 seconds.
- SET(RRNAME): Specifies the name of the alternate destination of a client's reroute request. The name can consist of alphanumeric characters (A – Z, 0 – 9) and special characters (@, #, \$). IMS Connect translates lowercase characters to uppercase characters. This parameter is optional and defaults to HWS\$DEF.
- SET(SMEMBER): Specifies the name of the OTMA super member to which this IMS data store belongs. If specified, this value overrides the value of the SMEMBER parameter specified on the HWS statement in the IMS Connect configuration member. To disable the value of SMEMBER specified on this IMS data store, specify the parameter with no value, for example, SMEMBER(). This parameter is optional and defaults to blanks.
- SET(TMEMBER): Specifies the XCF member name of the IMS that this IMS Connect communicates with in the XCF group.

Examples/Equivalentents - CREATE IMSCON TYPE(DATASTORE)

- Example using the LIKE keyword referencing an existing Datastore definition

- New type-2 command

```
CREATE IMSCON TYPE (DATASTORE) NAME (IMS3) LIKE (IMS1)
SET (MEMBER (ICON3) , TMEMBER (IMS3) )
```

- Equivalent configuration statement

```
DATASTORE=(ID=IMS3 , MEMBER=ICON3 , TMEMBER=IMS3 ,
APPL=APPL1 , ACTO=120 , CM0ATOQ=TOQ1 , DRU=DFSYDRU0 ,
GROUP=XCFGRP1 , MAXI=5000 , OAAV=7200 , RRNAME=RR1)
```

Here is an example of dynamically creating a DATASTORE, again using the type-2 CREATE IMSCON command but this time referencing the name of an existing datastore to set the attribute values.

The TMEMBER() parameter setting is optional, but in this case it was appropriate to explicitly specify it to override what would have set otherwise. Had we not specified TMEMBER(IMS3) here, the default would have been TMEMBER(IMS1) since that is the setting associated with the IMS1 datastore referenced with LIKE(IMS1).

The equivalent settings in the DATASTORE configuration statement in HWSCFGxx are shown here.

Example Output – CREATE IMSCON TYPE(DATASTORE)**Command input**

```
CREATE IMSCON TYPE (DATASTORE) NAME (IMS2)
SET (MEMBER (ICON2), TMEMBER (IMS2), GROUP (XCFGRP1))
```

Command output

```
PLEX1          IMS Single Point of Control
Command ==>

----- Plex . . PLEX1  Route . . HWS1
Response for: CREATE IMSCON TYPE (DATASTORE) NAME (IMS2) SET (MEM...
DataStore MbrName      CC
IMS2      HWS1         0
```

Shows
command
was issued
OK

Console output of command result

```
HWSD0290I CONNECTED TO DATASTORE=IMS2 ; M=DSC1
```

Shows
datastore
was created
OK

0 System Part 2: 40

Here, we show an example of using the type-2 CREATE IMSCON command to dynamically create a datastore. As previously discussed, the commands are asynchronous and note that a completion code (CC) of 0 shown on the TSO SPOC output screen indicates that the command was submitted to IMS Connect. When the datastore is successfully dynamically created, an HWSS0790I message is output to the console. The M=DSC1 shown represents the module code that issued the message (M= is included in all IMS Connect messages).

In the TSO SPOC command input example, note that the command was routed to the IMS Connect named HWS1. If all IMS Connects in the IMSplex are sharing the same datastore, the user could create the datastore definition for each IMS Connect by leaving the Route field blank so that each IMS Connect receives the command. In the TSO SPOC output shown, the port is dynamically created for the HWS1 IMS Connect.

Operational Considerations

- Definitions created by CREATE commands do not persist across IMS Connect restart
 - To persist changes, make the following updates to HWSCFGxx manually
 - Regular ports: TCPIP statement, PORT sub-parameter
 - DRDA ports: ODACCESS statement, DRDAPORT sub-parameter
 - Datastores: DATASTORE statement

In order to preserve newly created ports and datastores across restarts of IMS Connect, make sure that the respective definitions in HWSCFGxx are updated. For dynamically created regular ports, update the PORT sub-parameter of the TCPIP statement. For dynamically created DRDA ports, update the DRDAPORT sub-parameter of the ODACCESS statement. For dynamically created datastores, update the DATASTORE statement.

Security Considerations

- Add RACF definitions for OM in OPERCMDS class to restrict access to CREATE IMSCON command

IMS Command	Command Keyword	RACF access authority	Resource name
CREATE	IMSCON	UPDATE	IMS.plxname.CRE.IMSCON

Benefits

- IMS Connect customers can now add new ports and IMS datastore connections without having to restart IMS Connect
- Improved IMS Connect availability
- Existing type-2 command architecture leveraged

In summary, the IMS 13 IMS Connect command enhancements leverage existing type-2 command architecture to allow IMS Connect users to dynamically define and enable regular/DRDA ports and datastores. This eliminates the need to restart IMS Connect to activate definitional changes, increasing overall IMS Connect availability.

IMS Repository Enhancements

Repository Enhancements - Highlights

- Improved documentation and examples

- Runtime definition changes can now be applied to an IMS when it warmstarts or emergency restarts
 - Changes made to active systems with the IMPORT SCOPE(ALL) command are applied to an inactive IMS when it is restarted

- Benefits
 - Simplification of IMSplex management
 - Easier to maintain synchronization of IMS systems' resources in a DRD with repository environment

Improved Documentation and Samples

- **IMS 13 System Definition (SC19-3660)**
 - Procedures used in IMS environments (Chapter 18)
 - Sample JCL and procedures to load the Repository Server from SYSGEN, MODBLKS and IMS log records:

CSLURLFL - Procedure to load Repository Server from log records
CSLURLFM - Procedure to load Repository Server from MODBLKS
CSLURLFS - Procedure to load Repository Server from SYSGEN

CSLURJL0 - Sample JCL for invoking the CSLURLFL procedure
CSLURJM0 - Sample JCL for invoking the CSLURLFM procedure
CSLURJS0 - Sample JCL for invoking the CSLURLFS procedure

CSLURST2 - Create the STAGE2 to RDDS to Catalog load job.
DFSURST0 - Updated to support NOMODBLKSHLQ parameter.

IMPORT Command with SCOPE()

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

- SCOPE() is an IMPORT parameter that indicates which IMS systems the IMPORT will apply to
 - SCOPE(ALL) applies the IMPORT command to each IMS in the IMSplex, and is recommended (default)
 - Route command to all IMS systems
 - No import performed for those systems without definitions in the repository
 - Active systems that have resources defined in the repository will perform the import

05 System Part 2: 47

When you issue an IMPORT SOURCE(REPO) command, indicate whether it should apply to all IMSs including active and inactive systems or only the active IMS systems in the IMSplex, using the SCOPE() parameter.

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 both active/inactive IMS systems and is recommended to maintain synchronized definitions across the IMSplex that match the repository definitions.

IMPORT Command with SCOPE() ...

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

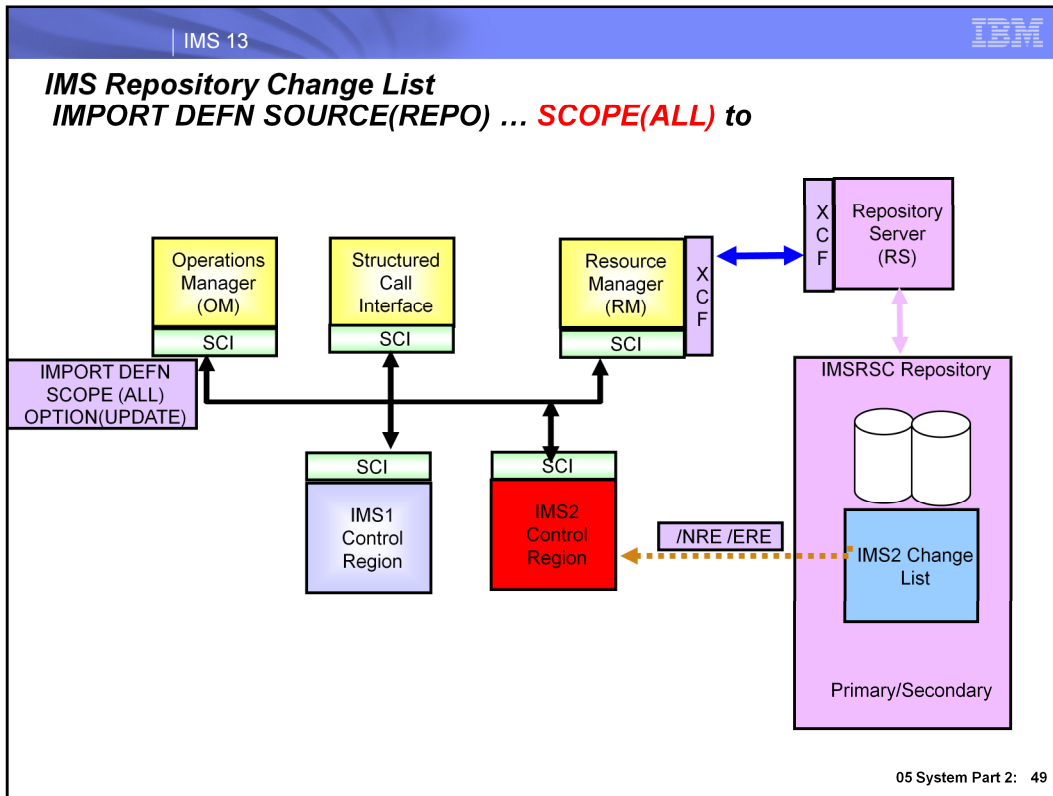
- SCOPE() is an IMPORT parameter that indicates which IMS systems the IMPORT will apply to
 - SCOPE(ALL) ...



- Inactive systems that have resources defined in the repository will have an **IMS change list** created for it, which contains the resource names it would have imported had it been up
- IMS change list will be read during warm/emergency restart and will thus be kept synchronized with the other IMS systems in the IMSplex

05 System Part 2: 48

If an IMS system is inactive, an IMPORT with SCOPE(ALL) will result in an **IMS change list** being created for it in the repository, which contains a list of the resources the IMS would have imported had it been active. The IMS change list will contain a subset of the IMS resource list, once again reflecting the resources that weren't imported due to the IMS being inactive. When the inactive IMS warmstarts or emergency restarts, it will apply the IMS change list and import the resources specified in it, thereby maintaining synchronization with the other active IMSs in the IMSplex that have already completed the import when the command was previously issued. This scenario especially applies to a cloned environment where an EXPORT with SET(IMSID(*)) was issued, thereby updating all IMS resource lists associated with IMSs that are both active and inactive. A subsequent IMPORT with SCOPE(ALL) would then read in all of these definitions that were updated by the previous EXPORT and maintain definitional synchronization across the multiple cloned IMS systems.



This diagram shows the use of the IMPORT command with SCOPE of ACTIVE. In this example IMS2 is not active when the IMPORT command was entered. Note a change list for IMS2 is created.

When IMS2 is started via Normal or Emergency restart changes are automatically imported so that the IMS is synchronized with all other IMS systems in the IMSplex. If a change list is created in the repository, a response line is returned on the IMPORT command for each resource name in the change list with the IMSID of the IMS for which the change list is created, along with the import type (IMPTYPE) of the change list.

Note: A change list is created only if the command master IMS is IMS Version 13 or later and the RM that processes change list requests is at V13 level or higher.

If the IMPORT DEFN SCOPE(ALL) command is issued without the OPTION(UPDATE) keyword and routed to an IMS system that has one or more resources or descriptors already defined, the IMPORT command results in a nonzero return code. The import might be successful at other active IMS systems, and the command might be successful in creating the change list in the IMSRSC repository for inactive IMS systems.

To avoid a nonzero return code and reason code from the IMPORT DEFN SCOPE(ALL) command, do one of the following:

- Specify the IMPORT DEFN SCOPE(ALL) command with the OPTION(UPDATE) keyword and route it to all IMS systems (ROUTE(*))
- Route the IMPORT DEFN SCOPE(ALL) command to active IMS systems in which the resource or descriptors do not exist

IMPORT Command Usage with SCOPE(ALL)

- Use **IMPORT DEFN SOURCE(REPO) ... SCOPE(ALL)** to read definitions into all IMS systems in the IMSplex (active and inactive)
 - An active IMS will:
 - Import the specified stored resource definitions from the repository
 - These definitions become runtime resource definitions in the control region
 - An inactive IMS will:
 - Have an IMS change list created for it within the repository
 - Contains a subset of its IMS resource list names that it would have imported had it been active
 - Internally import the updated stored resource definitions at warmstart or emergency restart
 - These definitions become runtime resource definitions in the control region
 - Have its IMS change list deleted at the end of restart after the log records processed...or if it coldstarts since autoimport will read its updated stored definitions

We've mentioned the IMPORT command several times during the session, but let's now more closely explore what happens when an IMS is inactive and an IMPORT with SCOPE(ALL) is issued.

We know at this point that the active IMS systems will import the stored definitions into the control region, but for an inactive IMS system, recall the IMS change list that will be created for it when this command is issued.

An IMS change list contains a subset of the inactive IMS's resource list that it would have imported had it been active at the time the command was issued. When this inactive IMS warmstarts or emergency restarts, it will apply the IMS change list and import the resources specified in it, thereby maintaining synchronization with the other active IMSs in the IMSplex that have already completed the import when the command was previously issued. If the IMS coldstarts, the IMS change list is deleted and the IMS reads its entire resource list via automatic import.

Messages Issued During IMS Change List Processing

- A successful IMS change list processing sequence results in the following messages being issued

```
DFS4408I REPOSITORY CHANGE LIST PROCESSING INITIATED
DFS4414I REPOSITORY CHANGE LIST PROCESSING INITIATED FOR DESCRIPTOR
      NAME=descname TYPE=desctype
DFS4410I REPOSITORY CHANGE LIST PROCESSING SUCCEEDED FOR RSCTYPE
      rsctype COUNT count
      .
      .
DFS4410I REPOSITORY CHANGE LIST PROCESSING SUCCEEDED FOR RSCTYPE
      rsctype COUNT count
DFS4412I REPOSITORY CHANGE LIST PROCESSING COMPLETED
```

05 System Part 2: 51

IMS issues message DFS4408I to indicate change list processing is initiated. IMS issues message DFS4414I for every descriptor to be imported from the change list, including the descriptor name and type. If the change list is successfully processed, IMS issues the DFS4410I message for each resource and descriptor type it successfully processes, including a count of the total number of resources or descriptors that were processed for that particular type. Since there are 4 resource types and 4 descriptor types, there could be up to 8 DFS4410I messages issued here. If the import from the change list fails (DFS4411E message appears), the descriptor is out of sync with the stored resource definitions in the IMSRSC repository. If there are no errors, the DFS4411E message will not be issued and instead, message DFS4412I indicating that change list processing is completed is issued. Note that none of these DFS messages will be issued if the IMS is restarting without the IMSRSC repository enabled.


IMS Change List Processing - Example Error Scenarios

- An error during internal import could result in one of the following messages (import will also be aborted)
 - `DFS4401E RM requestname REQUEST FAILED, RC=rc RSN=rsn
ERRORTXT=errortext`
 - RM return/reason codes indicate why import processing failed are documented in CSLRRR macro, for example
 - RM reason code 5508 = Repository server not available
 - RM reason code 5518 = Repository not available
 - `DFS4411E REPOSITORY CHANGE LIST PROCESSING FAILED RC=rc
RSN=rsn`
 - IMS return/reason codes indicate why IMS change list processing failed and are documented in DFSCMDRR macro, for example
 - IMS reason code 4104 = No RM address space
 - IMS reason code 4108 = No SCI address space

Here, we highlight two potential error messages that could be issued during internal import processing of an IMS change list: DFS4401E for an import error and DFS4411E for an IMS change list processing error. If any of the resources/descriptors in the change list fails import, the import is aborted and none of the resource/descriptor definitions from the IMS change list are imported. The DFS4411E message is new in IMS 13 indicates the error return/reason code for the error in internal import from the IMS change list. The resources and descriptors remain as NOTINIT, until IMS is restarted and the next internal import from change list succeeds, or user issues an IMPORT DEFN SOURCE(REPO) command to successfully import the resource and/or descriptor definitions from the IMSRSC repository. The QUERY DB, QUERY PGM, QUERY RTCODE, and QUERY TRAN commands can be issued to display the resources that are in the NOTINIT-xx-REPOCHGLIST state due to the internal import from the IMS change list. There are no QUERY commands for displaying NOTINIT descriptors, since descriptors don't have status, so look for message DFS4414I for the descriptor names that are still NOTINIT if the internal import from change list failed.

- A DELETE DEFN command will delete the resource name from a IMS change list in the IMSRSC repository if a change list exists for one or more IMS systems specified on the FOR() keyword as the resource definition is being deleted and will not be available to be imported when the IMS system restarts. **Note:** The IMS change list in the IMSRSC repository is not created when a DELETE DEFN command is issued and one or more IMS systems specified on the FOR() keyword are down. IMS recommendation is to delete the IMS runtime definition from IMS using the DELETE command followed by deleting the resource definitions from the IMSRSC repository using the DELETE DEFN command. However if the recommended procedure is not followed and resource definition is deleted from the IMSRSC repository for the IMS before it is deleted in the IMS system then it still exists in the IMS after restart. In this case, user must issue the DELETE command to delete the runtime resource definition after IMS restarts.
- In an XRF configuration, the recommendation is to issue the EXPORT DEFN command with the IMSid of the active and the alternate so the resource definitions are added to the IMS resource lists of the active and alternate system. When an IMPORT DEFN command is issued on the XRF configuration, the IMS change list is created for the alternate system in the repository. The IMS change list for the XRF alternate is not reprocessed at IMS takeover as all the information in the IMS change list is already processed as a part of the X'22' and the checkpoint log records. The IMS change list for the XRF alternate is deleted at end of takeover.

IMSRSC Repository Contents

- What is contained within an IMSRSC repository?
 - Stored resource definitions for DRD resources for one or more DRD-enabled IMS systems (programs/transactions/databases/FP routing codes)
 - Resource lists for each IMS
 - **IMS change lists** 
 - **Contain resource and descriptor names that were imported using the **IMPORT SCOPE(ALL)** command when an IMS was down**
 - Behind the scenes, a separate IMS change list is actually created for each resource/descriptor type
 - **Applied at next warm/emergency restart after IMS log is processed – internal import is done**
 - **IMS change list is deleted at the end of warm/emergency restart and at end of coldstart**
 - Allows synchronization with the other IMS systems in the IMSplex

The IMS repository data sets store IMS resource definitions. In IMS 12, there is one type of IMS repository, the IMSRSC repository for managing DRD definitions.

An IMSRSC repository contains three major types of information:

- 1) Resource lists for each IMS system that is using the repository server. These contain a list of all the programs, transactions, databases, and FP routing codes defined in a particular IMS system.
- 2) The actual stored resource definitions for all programs, transactions, databases, and FP routing code definitions that will be managed by the repository server. Different attribute values for a particular resource definition are supported by a generic definition and IMS-specific definitions where the attribute is different. An example of this is for the MSC SIDR and SIDL settings. Note that with RDDs, manual processes are required to maintain different attribute values.
- 3) IMS change lists that contain resource and descriptor names that correspond to resource changes made when an IMS was down when an **IMPORT SCOPE(ALL)** is issued. If a change list exists for the IMS that is being restarted, the database, program, transaction, and routing code resources and descriptors in the IMS change list and that apply to the IMS environment are quiesced and are not available for use until the stored resource definitions are imported from the repository. These are applied at the next warm or emergency restart after the IMS log is processed via an internal import process for that particular IMS system that was down. For the resources or descriptors that are in the IMS change list and that do not exist in IMS, the runtime resource definitions are created from the stored resource definitions in the repository. For the resource or descriptors that exist in IMS, the runtime resource definitions are updated with the stored resource definitions from the repository. The change list for the IMS system is deleted at the end of the cold, warm, or emergency restart. Note that with RDDs, manual coordination is required to handle IMSs that are down.

The IMS change list is created for an XRF alternate IMS during the **IMPORT DEFN SOURCE(REPO) SCOPE(ALL)** command if the resource definition is defined for the XRF alternate system.

When the XRF alternate is in restart mode, it is considered inactive. The IMS change list that is created for the XRF alternate is deleted during takeover without being processed, because the resource definitions are created or updated on the alternate from the log records that are sent from the IMS active system.

If one exists, the IMS change list for the IMS active system is processed during the restart of the IMS active system, and the X'22' log record that was written during the internal import is processed on the XRF alternate to obtain the resource definitions. The IMS change list is ignored and deleted during takeover.

The IMS change list for the XRF alternate is not reprocessed at IMS takeover because all the information in the change list is already processed as a part of the X'22' log record and the checkpoint log records. The IMS change list for the XRF alternate is deleted at the end of takeover.

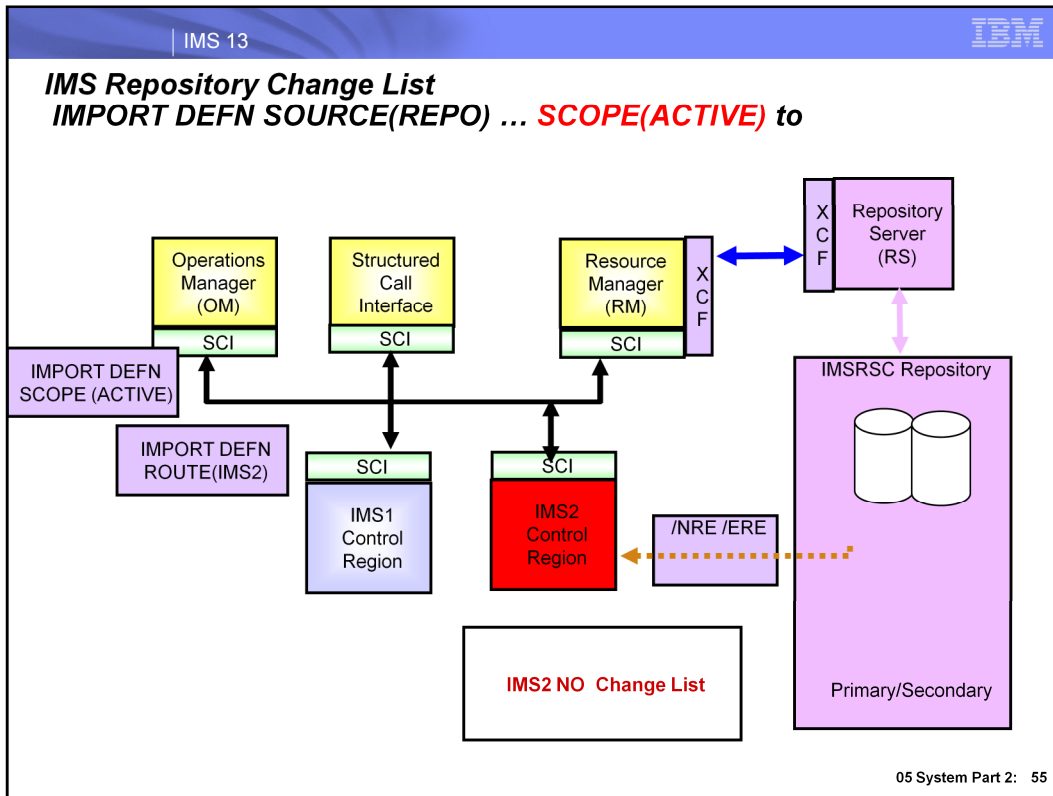
IMPORT Command with SCOPE()

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

- **SCOPE()** is an **IMPORT** parameter that indicates which **IMS** systems the **IMPORT** will apply to
 - **SCOPE(ALL)** ...continued
 - Applies **IMPORT** command to all **IMS** systems in the **IMSplex** for which the resource is defined. This is recommended (default)
 - **SCOPE(ACTIVE)** will apply the **IMPORT** command to all of the active **IMS** systems in the **IMSplex** (same as in **IMS 12**)
 - **IMS systems that are down will not have an IMS change list created and will not be synchronized with the other systems (manual IMPORT required) at /NRE or /ERE (coldstart and /ERE COLDSYS ok)**

05 System Part 2: 54

SCOPE(ACTIVE) applies the import to only the active **IMS** systems. Any inactive **IMS** system will not complete the import since an **IMS** change list will not be built for it in this case. Therefore when it warmstarts or emergency restarts, its definitions will not be synchronized with the definitions of the other **IMS** systems in the **IMSplex**. To re-establish synchronization, you can enter an **IMPORT** command to import the resources that the other active **IMS** systems imported while it was inactive. If the inactive **IMS** is coldstarted, it will be synchronized with the other **IMS** systems since it will read its entire **IMS** resource list. This is the same as in **V12** and is mentioned here only to contrast it with the **SCOPE(ALL)** specification.



This diagram shows the use of the IMPORT command with SCOPE of ACTIVE.

In this example IMS2 is not active when the IMPORT command was entered.

Note a change list for IMS2 is not created.

When IMS2 is started via Normal or Emergency restart changes are not automatically imported

After IMS2 completes init processing the IMPORT DEFN SOURCE(REPO) ROUTE(IMS2) command must be entered for IMS2 and any other IMS systems that were not active to synchronize runtime resource definitions in all IMS systems in the IMSplex, you must issue the IMPORT command manually.

IMPORT Command Usage with Repository DRD

- Use `IMPORT DEFN SOURCE(REPO) ... SCOPE(ACTIVE)` to read definitions only into active IMSs in the IMSplex
 - Active IMS systems will import the specified definitions
 - **Inactive IMS systems will not participate and will not receive updates**
 - **IMS change list is not created, so warmstart/emergency restart will not apply any changes**
- To ensure resource/descriptor definition synchronization across all IMS systems, complete one of the following:
 - Coldstart the inactive IMS so that all of the resource definitions defined for the IMS are automatically imported from the repository
 - Use `IMPORT` with `SCOPE(ALL)` instead of `SCOPE(ACTIVE)`
 - Warmstart or emergency restart the inactive IMS
 - Then issue another `IMPORT` command to re-establish its synchronization with the other IMSs; `ROUTE` command to previously inactive IMS

You must issue the `IMPORT` command manually if you want all IMS systems in the IMSplex to be in sync with the resource and descriptor definitions.

`SCOPE(ACTIVE)` applies the import to only the active IMS systems. Any inactive IMS system will not complete the import since an IMS change list will not be built for it in this case. Therefore when it warmstarts or emergency restarts, its definitions will not be synchronized with the definitions of the other IMS systems in the IMSplex. To re-establish synchronization, you can enter an `IMPORT` command to import the resources that the other active IMS systems imported while it was inactive. If the inactive IMS is coldstarted, it will be synchronized with the other IMS systems since it will read its entire IMS resource list. Another way to achieve synchronization is to warmstart or emergency restart and then issue an `IMPORT` command to read in the stored definitions that may have been updated while it was inactive. Finally, another way to maintain synchronization among the runtime definitions in the IMSplex is to issue the `IMPORT` command with the recommended `SCOPE(ALL)` parameter.

Migration Considerations

- If automation includes IMPORT with SCOPE(ALL), account for the fact that it will now apply to inactive IMS systems
 - IMS change list will be created and applied at warmstart/emergency restart

- PM77568 Repository Change-list Enhancement
 - requires new V12 coexistence PM80588
 - APAR PM80588 is co-existence APAR for IMS V12 with IMS V13.
 - Required by all IMS V12 users of the IMSRSC repository in a mixed environment of IMS V12 and IMS V13 and must be applied before IMS V13 APAR PM77568 is applied.
 - ➔ If no V12 SCI, RM, OM in IMSPEX then PM80588 not required
 - Description:
 - When the QUERY, UPDATE and DELETE of a IMS change list is being processed by a V12 Resource Manager (RM) address space. APAR PM80588 adds logic to disable support of unsupported functions in IMS V12

Summary

- Runtime definition changes can now be applied to an IMS when it warmstarts or emergency restarts
 - Changes made to active systems with the IMPORT SCOPE(ALL) command are applied to an inactive IMS when it is restarted
- Benefits
 - Simplification of IMSplex management
 - Easier to maintain synchronization of IMS systems' resources in a DRD with repository environment

User Exit Enhancements

User Exit Dynamic Refresh Capability – IMS 12 SPE

- Delivered via IMS 12 SPE PM56010 (PTF UK79071)
- Command:
`REFRESH USEREXIT TYPE(exittype)`
 - Refreshes the user exit types specified without bringing IMS down
 - Eligible exit types are:
 - ICQSEVNT
 - ICQSSTEV
 - INITTERM
 - PPUE
 - RESTART
- USER_EXITS section in DFSDFxxx (where xxx is the value of the DFSDF initialization parameter) read to pick up the EXITDEF statements for the user exit type(s) specified on the command
 - An optional MEMBER() parameter can be used to point to a different DFSDFxxx member

The REFRESH USEREXIT command was delivered in IMS 12 as an SPE under APAR PM56010 (PTF UK79071). This gives the ability to dynamically refresh supported user exit types while IMS is active and can be used to add, update or delete user exits. All exit types that support the enhanced user exit services are eligible for REFRESH, and are listed here.

The USER_EXITS section of the DFSDFxxx member is read and the exit routines listed for the type being refreshed are loaded. If there are no errors encountered while processing the user exit type, the new user exit routines will replace the current routines and all subsequent calls to the user exit will call the new routines. The old routines will be deleted when there are no more callers in any of the exit routines defined for the user exit type.

The MEMBER() parameter provides a means to point to a different DFSDFxxx member for testing. If the parameter is used in production, the user must ensure that the default member is updated to ensure that the correct user exit routines are loaded if IMS must be restarted.

User Exit Dynamic Refresh Capability – IMS 12 SPE

- Refresh of each exit type is independent
- If multiple exit types are specified and the refresh fails for one exit but succeeds on another exit, that exit will be refreshed
- Can also be used to add or delete a user exit
 - Update DFSDFxxx, then issue REFRESH USEREXIT command

If multiple user exit types are refreshed with the same command, a failure for a particular user exit type does not affect the other user exit types that are part of the refresh.

In addition to refreshing an exit with an updated version, you can use the REFRESH USEREXIT command to either add or delete a user exit type to IMS. To add a user exit type, insert EXITDEF statements into the USER_EXITS section of the DFSDFxxx member for the user exit type you want to add and then enter the REFRESH USEREXIT command for that exit type. To delete a user exit type, remove the EXITDEF statements from the USER_EXITS section of the DFSDFxxx member for the user exit type you want to delete and then enter the REFRESH USEREXIT command for that exit type.

REFRESH USEREXIT Example

- Edit DFSDFxxx before issuing REFRESH USEREXIT command

```
<SECTION=USER_EXITS>
EXITDEF=(TYPE=ICQSSTEV, /* IMS CQS STR EVENT */
        EXITS=(DFSCSTX2,DFSCSTX0,DFSCSTX1)) /* EXIT LIST */
EXITDEF=(TYPE=ICQSEVNT, /* IMS CQS EVENT */
        EXITS=(DFSCQEX1,DFSCQEX2,DFSCQEX0)) /* EXIT LIST */
EXITDEF=(TYPE=INITTERM, /* IMS Init/Term Exit */
        EXITS=(DFSITRX2,DFSITRX3)) /* EXIT LIST */
```

Modules DFSCSTX2 and DFSITRX3 are not in the load library

This charts show the USER_EXITS section of the DFSDFxxx member that is used for the command example. It contains additional exit definitions, but they are not relevant to this command. Note that two of the module names are not present in the load library, which ultimately prevents the ICQSSTEV and INITTERM exit types from successfully being refreshed.

REFRESH USEREXIT Example – Screen 1 of Output

Command input: REFRESH USEREXIT TYPE(ICQSTEV,ICQSEVNT,INITTERM)

```

PLEX1                      IMS Single Point of Control
Command ==>
No clients returned return code 0. Check return code(s).
----- Plex . .      Route . .      Wait . .
Response for: REFRESH USEREXIT TYPE(ICQSSTEV,ICQSEVNT,INITTERM...)
ExitType ModName  MbrName  CC CText
ICQSSTEV          SYS3      92 COMMAND PROCESSING ERROR
INITTERM          SYS3      92 COMMAND PROCESSING ERROR
ICQSEVNT DFSCQEX0 SYS3      0
ICQSEVNT DFSCQEX1 SYS3      0
ICQSEVNT DFSCQEX2 SYS3      0

```

CC 92 - exit types associated with the module names are not present in the load library and cannot be refreshed

This chart shows the command response lines for the command example. The exit types associated with the module names not present in the load library are not able to be refreshed, which is reflected with a completion code (CC) of 92. The modules that are of the ICQSEVNT exit type are successfully refreshed and the returned CC is 0. To get additional detail associated with the command response, press PF4 to view the log.

REFRESH USEREXIT Example – Screen 2 of Output

Command input: REFRESH USEREXITTYPE (ICQSTEV, ICQSEVNT, INITTERM)

```

PLEX1                      IMS Single Point of Control
Command ==>
----- Plex . .           Route . .           Wait . .
Log for . . : REFRESH USEREXIT TYPE(ICQSSTEV,ICQSEVNT,INITTERM... More:  +>

IMSpLex . . . . . : PLEX1
Routing . . . . . :
Start time . . . . : 2012.109 09:51:07.14
Stop time . . . . . : 2012.109 09:51:07.27
Return code . . . . : 0200000C
Reason code . . . . : 00003008
Reason text . . . . : None of the clients were successful.
Command master . . : SYS3

      Return      Reason
      MbrName     Code      Code      Reason text
-----
SYS3      0000000C   00003000   At least one request successful

```

PF4 to see Log

Here, we see the log after pressing PF4. Additional information is returned -- but to see the most useful information, page to the bottom of the log data by pressing PF8.

REFRESH USEREXIT Example – Screen 3 of Output

Command input: REFRESH USEREXIT TYPE (ICQSTEV, ICQSEVNT, INITTERM)

```

PLEX1                IMS Single Point of Control
Command ==>
-----
Plex . .             Route . .             Wait . .
Log for . . : REFRESH USEREXIT TYPE(ICQSSTEV,ICQSEVNT,INITTERM... More: +>

                MbrName  Messages
-----
SYS3    DFS4570E USER EXIT SETUP FAILED.  EXIT TYPE - ICQSSTEV MODULE NAME -
                DFSCSTX2 REASON - BLDL ERROR          RC - 04 SYS3
SYS3    DFS4570E USER EXIT SETUP FAILED.  EXIT TYPE - INITTERM MODULE NAME -
                DFSITRX3 REASON - BLDL ERROR          RC - 04 SYS3

```

These lines will appear to the right (page to right to see these)

PF8 to page down

This screen shows more detail of the command response (reached with PF8 from the previous screen). Note that the names of the two exit routine names that were missing from the load library are shown. As a sidenote, these specific lines of output will appear scrolled off of the screen and to see them, a user needs to page to the right. The fact that there is more information available to the right is signified with the "More: +>" shown on the right.

IMS 13 User Exit Enhancements

- Enhanced user exit services extended to additional IMS control region user exits
- Users can now dynamically refresh more IMS user exit routines to bring in an updated version of the exit (or add/delete)
 - Significantly reduces downtime since IMS control region no longer requires restart
- Users can now display information about more user exits that are defined in the USER_EXITS section of DFSDFxxx
 - Provides useful exit information to the user
- Users can now code an exit to leverage IMS's ability to call multiple routines of the same type from a single point within the exit
- **Benefits**
 - Expanded flexibility
 - IMS availability is increased
 - Management of user exits eased

This line item adds enhanced user exit services support to a number of additional user exits. This support not only includes the ability to dynamically add, update or delete a user exit without restarting the IMS control region, but also to display useful exit information and enable IMS to call multiple user exit routines from a single point within an exit. Enabling additional exits to use the enhanced user exit services expands flexibility, improves the availability of the IMS online environment, and makes it easier to manage user exits.

Enhanced User Exit Services Added to More Exit Types

- AOIE (DFSABOE00, Type-2 Automated Operator Exit)
- BSEX (DFSBSEX0, Build Security Environment Exit)
- NDMX (DFSNDMX0, Non-Discardable Message Exit)
- RASE (DFSRAS00, Resource Access Security Exit)
- OTMAYPRX (DFSYPRX0, OTMA Destination Resolution Exit)
- OTMAYDRU (DFSYDRU0, OTMA User Data Formatting Exit)
- OTMARTUX (DFSYRTUX, OTMA Resume TPIPE Security Exit)
- OTMAIOED (DFSYIOE0, OTMA Input/Output Edit Exit)
- LOGWRT (DFSFLGX0, Logger Exit)
- LOGEDIT (DFSFLGE0, Log Edit Exit)

These are the user exits that can now use the enhanced user exit services in IMS 13.

Note that for the OTMAYDRU (OTMA User Data Formatting) exit type, only the system default DRU exit DFSYDRU0 is refreshable. Any other DRU exit name specified by an OTMA descriptor or OTMA client is not refreshable.

Enhanced User Exit Services Added to More Exit Types

- The exits listed can now leverage enhanced user exit services to:
 - Use REFRESH USEREXIT to bring in new copy of an exit based on type
 - Either all exits of a certain type as listed in DFSDFxxx will be refreshed or none
 - Can also add/delete as long as DFSDFxxx updated first
 - Use QUERY USEREXIT to display useful exit information
 - Code multiple exits of the same exit type to be called from a single entry point

Once again, the enhanced user exit services that now apply to a wider scope of user exits enable an exit to be dynamically added, changed, or deleted with a USER REFRESH command, and have its information dynamically displayed with a QUERY USEREXIT command. The enhanced services also allow multiple exits of the same type to be called and executed in the order they are specified in on the EXITDEF statement of DFSDFxxx.

Considerations

- New instances of an exit routine brought in via REFRESH USEREXIT command use the same static work area as original copy
 - SXPL flag set indicating SXPL shared (reset when original copy deleted)
 - Both original and new user exits may execute simultaneously if refresh is done when original exit is called
 - **CAUTION:** If a new user exit reformats the static work area, it should wait until this flag is reset since the original exit may rely on the area to remain intact/unmodified
 - Original exit deleted when there are no more callers after it has been refreshed, then flag is reset
- Removing DFSRAS00 from DFSDFxxx, then issuing REFRESH USEREXIT while also specifying ISIS=C/A in DFSPBxxx will result in a message being issued that indicates the exit will no longer be called

If a user exit is called while it is being refreshed with a new copy, they will each share the static work area. During this time, it is possible that each exit accesses the work area and the new exit brought in with the REFRESH USEREXIT command must not reformat the work area. For example, if a new user exit reformats the static work area, it should wait until this flag is reset since the original version of the exit might be relying on the work area to remain intact/unmodified. An SXPL flag representing whether or not the work area is being shared is set until the original exit is deleted. Behind the scenes, there is logic that is part of the REFRESH USEREXIT command that determines whether the original exit is being called and if it is not, it is deleted and the flag is reset. It is at this point that the new, refreshed version of the exit can proceed with reformatting the static work area.

Migration/Setup Considerations

- Exit routines are defined in the USER_EXITS section of the DFSDFxxx member
- Enhanced user exit services are optional for existing exits
 - To activate enhanced user exit services for an exit type that is newly supported in IMS 13:
 - Add the exit to DFSDFxxx
 - Issue a REFRESH USEREXIT command
 - Enhanced user exit services will then be available for use
- Migration is on a user exit type basis
 - A single user exit type can be migrated to use the new services while the other user exit types can continue with the non-enhanced services

Include exits that you intend to add, update or delete in the IMS runtime environment in the USER_EXITS section of DFSDFxxx. If you are currently using exits that do not use the enhanced user exit services, you continue using these exits. To activate the enhanced services for these exits, add the exits to DFSDFxxx, then issue a REFRESH USEREXIT command. You can also opt to use the enhanced services for some exit types, while continuing to use non-enhanced services for other exit types.

Security Considerations

- Add RACF definitions for OM in OPERCMDS class to restrict access to REFRESH USEREXIT command

IMS Command	Command Keyword	RACF access authority	Resource name
REFRESH	USEREXIT	UPDATE	IMS.plxname.REFRESH.USEREXIT
QUERY	USEREXIT	READ	IMS.plxname.QRY.USEREXIT

To restrict access to the type-2 commands REFRESH USEREXIT and QUERY USEREXIT, define resource profiles in the OPERCMDS class and grant the required user access authorities as shown in the table.

Summary

- **Enhanced user exit services added to several new exits**
 - Dynamic refresh
 - Display information
 - Call multiple exits of same type from single point
- **Benefits**
 - Expanded flexibility
 - IMS availability is increased
 - Management of user exits eased

SECURITY Macro Removal

Highlights

- All IMS security settings can now be defined as IMS startup parameters
 - Updates to SECURITY macro in system definition (SYSGEN) no longer required due to its removal
 - Previously, certain settings could only be defined in SECURITY macro
- Move security user exits out of the IMS nucleus into 31-bit storage
 - DFSCSGN0
 - DFSCTRN0
 - DFSCCTSE0

Benefits

- **Removal of SECURITY macro**
 - Significant reduction in system programmers' time/effort required in maintaining IMS systems
 - Management of security definitions eased
 - System definition (SYSGEN) process made more simple
- **Removal of selected user exits from IMS nucleus**
 - Exits are easier to maintain
 - Linking to IMS nucleus no longer necessary when one of the exits has been changed
 - Usage of 24-bit storage reduced

New IMS Startup Security Parameters

- RCLASS parameter added to DFSPBxxx PROCLIB member
 - RCLASS support in DFSDCxxx PROCLIB member will remain
 - DFSPBxxx RCLASS parameter value will override DFSDCxxx if specified in both
- SECCNT parameter added to DFSDCxxx PROCLIB member
- Changes to DFSPBxxx/DFSDCxxx will be supported by IMS Syntax Checker
- Retrofit SPE APARs/PTFs available for IMS startup security parameter enhancement activation in IMS 11 and IMS 12
 - PM48203/UK74050 (IMS 11)
 - PM48204/UK74051 (IMS 12)
 - If specifying RCLASS in DFSPBxxx/DFSDCxxx, can also have the following APARs/PTFs applied to avoid an error message being issued when it shouldn't be
 - PM72199/UK82616 (IMS 11)
 - PM73558/UK82617 (IMS 12)

You are now able to specify the RCLASS and SECCNT settings as IMS startup parameters.

RCLASS has been added to the DFSPBxxx member, which will co-exist with the ability to set this parameter within the DFSDCxxx member. Support for the DFSDCxxx remains for users who had previously specified it here instead of the SECURITY macro. However, if the RCLASS parameter is specified in both members, DFSPBxxx will take precedence. As a review, the RCLASS parameter specifies an identifier of 1 to 7 alphanumeric characters that is to be used to identify the IMS system as a resource class to RACF for transaction authorization and user ID verification.

SECCNT has been added to the DFSDCxxx member, which specifies the maximum number of terminal and password security violations to be accepted per physical terminal and the number of transaction command violations per transaction prior to master terminal notification of such violations. The default is 0, which nullifies notification to the master terminal. The number specified must be 0, 1, 2, or 3.

New IMS Startup Security Parameters

- If SECURITY macro remains in IMS 13 STAGE1, it will result in a return code 2 and the following error message will be issued
G115 SECURITY MACRO IS NOT SUPPORTED. SPECIFY SECURITY OPTIONS USING IMS EXECUTION PARAMETERS.
- SECURITY macro will be ignored and IMS initialization will continue

The SECURITY macro in IMS SYSGEN (STAGE1) can no longer be used to specify IMS security options. If the macro is specified in IMS SYSGEN, sysgen then there will be a message generated and the macro will be ignored. The SECCNT parameter in the COMM and MSGEN macro is still supported. Specifying SECCNT in the DFSDCxxx IMS PROCLIB member overrides the SECCNT specification in IMS SYSGEN.

User Exits removed from Nucleus

- User exits DFSCSGN0, DFSCTRN0 and DFSCTSE0 now bound separately, loaded from STEPLIB (if present) into 31-bit storage
 - New DFS1937I message indicates which user exits have been loaded
 - Can be used in automation to ensure that exits are being used
- DFSCSGN0 is called during IMS initialization and can get storage that can be shared between exits
 - Workarea that can be shared among the exits uses the new CTSECSEP and CTRNCSEP parameters
 - Replaces the old method of v-type addressing which was used for sharing storage when the exits were all in the nucleus

Prior to IMS 13, if DFSCTSE0 was used, its CSECT was included in DFSCTRN0. In IMS 13, all the security exit routines can be linked independently.

Additionally, in IMS 13, DFSCSGN0 can be called at IMS initialization (after the Initialization exit routine DFSINTX0) if it is included in STEPLIB. This mechanism allows the exit to obtain a work storage area and pass the address back to IMS. IMS can then pass this address to the other security exit routines every time they are called. The process uses the new CTSECSEP and CTRNCSEP parameters. Please see the exit interface documentation for more detail on how to update the DFSCSGN0, DFSCTRN0 and DFSCTSE0 user exits to use the new work area.

Operational Considerations

- Automation can check for DFS1937I message to determine whether user exits have been loaded by IMS and have control
- Can change exits to be 31-bit and/or use new parameter to share storage

Migration/Setup Considerations

- Apply SPEs for RCLASS/SECCNT support in DFSPBxxx/DFSDCxxx, respectively
 - PM48203/UK74050 (IMS 11)
 - PM48204/UK74051 (IMS 12)
- Define startup parameters equivalent to security parameters in SECURITY macro
 - Handling exit-related parameters that invoke DFSCCTSE0, DFSCCTRN0 and DFSCSGN0
 - On the SECURITY macro, TYPE=TRANEXIT or TYPE=SIGNEXIT if specifying that an exit will be called for transaction and signon authorization (these do not have startup parameter equivalents)
 - To continue using these security exits, include them in STEPLIB to ensure they are called (if macro definitions remain, they will be ignored)
 - If possible, remove macro entirely from STAGE1 before moving to IMS 13

In order to define RCLASS and SECCNT as execution parameters, ensure that the PTFs shown here are applied. Since the SECURITY macro is being removed, all parameters currently defined there must be defined as execution parameters with the exception of the exit-related parameters TYPE=TRANEXIT and TYPE=SIGNEXIT. In order to continue using transaction authorization and signon authorization, simply define the associated exits in STEPLIB and they will automatically be called without requiring that a parameter exists indicating that an exit will be used. If the exit-related parameters are left on the SECURITY macro in IMS 13, they will be ignored. If possible, remove the SECURITY macro being migrating to IMS 13.

Migration/Setup Considerations

- To address the DFSCTRN0, DFSCCTSE0 and DFSCSGN0 user exits:
 - Recommended:
 - Ensure all three are bound as separate modules
 - Split DFSCCTSE0 and DFSCCTRN0 into two separate load modules.
 - Use the new CTSECSEP and CTRNCSEP parameters so that the exit routines can use a common work area.
 - Alternatively,
 - If the exit routines cannot be linked separately or cannot use a common work area, they must be linked in the following manner:
 - If the CSECT of DFSCCTSE0 is part of DFSCCTRN0 source, DFSCCTSE0 must be linked as an ALIAS of DFSCCTRN0.
 - If virtual address spaces are used to exchange data between DFSCSGN0, DFSCCTRN0, and DFSCCTSE0, both DFSCCTSE0 and DFSCSGN0 must be linked as an ALIAS of DFSCCTRN0.
 - Link the modules as reentrant (RENT) and AMODE/RMODE 31. If the load modules are not linked as reentrant, IMS loads them multiple times.
- Keep an original version of DFSCSGN0, DFSCCTRN0 and DFSCCTSE0 when updating them to enable quick/easy fallback if need be

Since the DFSCCTRN0, DFSCCTSE0 and DFSCSGN0 user exits were removed from the nucleus in IMS 13, consideration must be given to maintaining their ability to communicate with one another. There are two options for this: treating the exits as standalone modules and using the new parameter to share the storage obtained during IMS initialization (recommended), or binding the modules together using ALIASing (sample JCL is shown on the next slide). In either case, the modules should be bound as re-entrant and also as AMODE/RMODE 31 to prevent them from being loaded multiple times. As a review, a re-entrant module can be used by multiple callers simultaneously in which concurrent activity is taking place. It is written so that none of its code is modifiable (no values are changed) and it does not keep track of anything. The callers keep track of their own progress (variables, flags, etc.), thus one copy of the re-entrant routine can be shared by any number of callers.

It is also recommended to keep an original version of each user exit to enable easy fallback to IMS 11 or IMS 12 if need be.

Migration/Setup Considerations

Sample JCL for the alternative option when the exits cannot be bound separately

```
//LINK1 EXEC PGM=IEWL,
//      PARM=(,
//      NCAL,XREF,LIST,RENT RMODE=ANY,AMODE=31)
//SYSPRINT DD SYSOUT=A
//TEXT DD UNIT=SYSVIO,DISP=(OLD,PASS),DSN=IMSDC33
//SYSLMOD DD DSN=IMSTESTL.TNUC0,DISP=SHR,UNIT=SYSDA,
//      VOL=SER=USER01
//RESLIB DD DSN=IMSTESTL.TNUC1,DISP=SHR,UNIT=SYSDA,
//      VOL=SER=USER01
//      DD DSN=IMSBLD.I13ATSMM.ARESLIB,DISP=SHR
//SYSUT1 DD UNIT=SYSVIO,DISP=(,DELETE),SPACE=(CYL,(10,1),RLSE)
//SYSLIN DD *
INCLUDE TEXT(DFSCTRN0)
INCLUDE RESLIB(DFSCSGN0)
ENTRY DFSCTRN0
ALIAS DFSCCTSE0
ALIAS DFSCSGN0
NAME DFSCTRN0(R)
```

Here is a JCL sample for the second option presented on the previous slide for maintaining communication ability between the DFSCTRN0, DFSCCTSE0 then DFSCSGN0 user exits. Note that the modules have been bound as re-entrant (RENT) with AMODE/RMODE 31 to prevent multiple loads. Also note that the 3 modules have been linked together using ALIASing, as can be seen in the lower portion of the JCL with DFSCCTSE0 and DFSCSGN0 being linked as ALIASes of DFSCTRN0.

Benefits

- All IMS security settings can now be defined as IMS startup parameters, SECURITY macro being removed
 - Reduction in system programmers' time/effort required in maintaining IMS systems
 - Management of security definitions eased
 - System definition (SYSGEN) process made more simple
- Move security user exits out of the IMS nucleus into 31-bit storage
 - Exits are easier to maintain
 - Linking to IMS nucleus no longer necessary when one of the exits has been changed
 - Usage of 24-bit storage reduced

RACF Password Phrase Support (IMS 12 SPE)

Highlights

- RACF password phrases (“passphrases”) can now optionally be used within TM Resource Adapter messages to IMS Connect and when signing on to IMS
- Security checking
 - Passphrase sent to RACF at authentication time
- Benefit
 - Passphrases are
 - More robust
 - 9 to 100 bytes
 - Can contain mixed-case letters, numbers and special characters
 - Easier to remember

Users of TM Resource Adapter, IMS Connect, the /SIGN command and VTAM are now able to sign/log onto IMS using RACF password phrases that are a minimum of 9 bytes and a maximum of 100 bytes. Password phrases are superior to 8-byte passwords since they are easier to remember, and are more robust. Use of a passphrase is optional, as 8-byte passwords can continue to be used.

Enhancements to Existing Function – TMRA & ICON

- **IMS TM Resource Adapter (TMRA)**
 - Will use new functionality to build a message including password phrase and send it to IMS Connect
- **IMS Connect**
 - Will be able to accept messages from IMS TM Resource Adapter that contain a passphrase
 - Users will be able to change their password phrase in the application data section of the message received from TMRA
 - Format: 'oldphrase' 'newphrase' 'newphrase'
 - To preserve single quotes that are part of the passphrase, add another single quote next to each
 - Layout of TMRA message:

LLLL	IRM	OTMA	LLZZApplication_Data
------	-----	------	----------------------

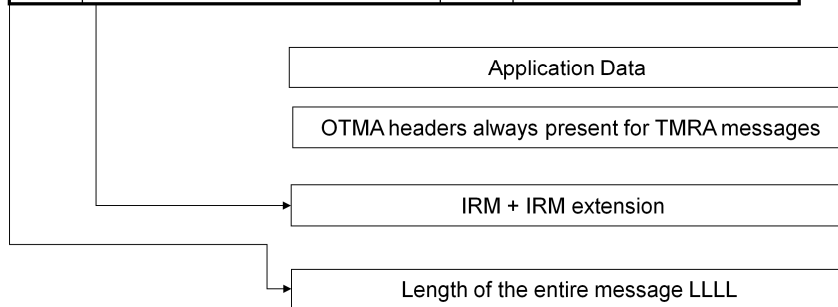
IMS TM Resource Adapter (or TMRA) can now include a passphrase in a message that it builds and passes to IMS Connect. IMS Connect, in turn, has been enhanced to accept this type of message from TMRA that includes a passphrase. Also, users can now change their passwords using the message that is sent from TMRA to IMS Connect using the format specified here. The layout of the message is also shown, and details of it are shown on the next two slides.

Enhancements to Existing Function – TMRA & ICON

Usage

- IMS TMRA externalizes the password/passphrase specification as a connection property
- Passphrase is passed in an IRM extension to IMS Connect
- IMS Connect uses the passphrase for authentication of the request

LLLL	IRM + LLZZ*PHRASE* passphrase	OTMA	LLZZTRANCODE DATA
------	-------------------------------	------	-------------------



IMS TMRA builds the IRM and the OTMA headers which are sent to IMS Connect with each input message. IMS TMRA externalizes the ability to define the passphrase through its connection factory interface in a fashion similar to how users have previously been able to define userids and passwords. For this new capability, IMS TMRA adds an IRM extension containing the password phrase to the message header. When IMS Connect receives the request, the userid along with the new passphrase are used for authentication.

IMS 13 IBM

Enhancements to Existing Function – TMRA & ICON

LLLL	IRM	OTMA	LLZZApplication_Data
------	-----	------	----------------------

LLZZHWSPWCH 'THE NIGHT IS ''YOUNG'' 'TODAY IS A NEW DAY' 'TODAY IS A NEW DAY'

Old passphrase	New passphrase x 2
----------------	--------------------

Password and passphrase change request eyecatcher

05 System Part 2: 88

The password change data is included in the Application Data section of the message that TMRA sends to IMS Connect. Notice that the password change data comes after an eyecatcher of “HWSPWCH”. In our example, we use the password change format that was shown previously.

Enhancements to Existing Function

- **MFS Panels**
 - Customizable base/sample panels which accommodate password phrase fields > 8 bytes are provided
 - If an 8-byte password phrase is used on these new panels, it will be passed to RACF as a password

MFS panels have been updated to accommodate password phrases, including new default panels. If the new defaults are used, users will need to be informed of this, but will still be able to enter passwords that are less than 9 bytes.

Enhancements to Existing Function

- **IMS /SIGN command -- RACF password phrase support added**
 - /SIGN **ONP** used when specifying a **passphrase**
 - Most appropriate for use with MFS panel
 - Must be 100 bytes passed to IMS (MFS panel will ensure this)
 - IMS will remove leading/trailing blanks
 - If < 9 bytes, IMS will pass it as a password to RACF
 - /SIGN **ONQ** used when specifying a **passphrase** with single quotes
 - Most appropriate for use when signing on from a terminal
 - IMS will not remove leading/trailing blanks
- **When changing password phrases, must specify another password phrase (same for passwords – can only change to another 8-byte password)**
- **Traditional password support with /SIGN ON <userid> <userpw> maintained**
- **VTAM logon user data**
 - Enclose the user ID and password phrase with quotes required by LOGON, for example:

```
LOGON APPLID(APPL1) DATA('USRT001 "this is my passphrase"')
```

05 System Part 2: 90

The IMS /SIGN command has been enhanced to support passphrases. Two new flavors of this command are now available: /SIGN ONP and /SIGN ONQ.

The use of /SIGN ONP is most appropriate with an MFS panel, since this command requires a passphrase that is 100 bytes. This can be ensured by an MFS panel that the password phrase is entered on. These password phrases can contain any character and are meant to be easier to remember than the possibly cryptic 8-character password.

The use of /SIGN ONQ is most appropriate when a user is entering signon credentials from a terminal. There is no requirement for the passphrase to be 100 bytes when this command is used and involves the use of single quotes.

Logon user data passed to VTAM also now supports the use of passphrases. An example of the logon format is shown. Note the use of both single and double quotes. The entire DATA() parameter containing the logon credentials is encapsulated in single quotes, and the passphrase itself is encapsulated in double quotes.

Enhancements to Existing Function

- User exits may require changes if RACF password phrase used
 - DFSGMSG0 (Greeting Messages user exit) can be changed to the default MFS panel to use password phrase
 - DFSCSGN0 (Signon/off security user exit) will need User Verification String (UVS) to change if password phrase is used
 - DFSLGNX0 (Logon user exit) must accommodate signon data including a password phrase passed to it

If RACF password phrases are used, the new default DFSGMSG0 MFS panel that supports passphrases can start being used, and the DFSCSGN0 and DFSLGNX0 exits will need to be updated so they can handle passphrases being passed to them.

Usage Considerations

- Both passwords and passphrases will be supported by IMS
- If using the following terminal types, it is recommended to use passwords instead of passphrases due to limited space
 - MTO terminal
 - WTOR/system console
 - TCO input
- Password phrases can be used for above, but use /SIGN ONQ since it does not require 100 bytes like /SIGN ONP does

While the usage of RACF passphrases is optional, the MTO, WTOR and TCO all have limited space in terms of entering user ID and password/passphrase data. Due to this, it is recommended to use passwords when signing on from these types of terminals. Of course, password phrases are able to be used -- but since there is limited space it is recommended to sign on with /SIGN ONQ since it does not require the 100 bytes that /SIGN ONP does.

Migration/Setup Considerations

- User exits may need to be changed to handle password phrases
- If not using RACF, check with vendors to determine if the External Security Manager supports password phrases
- Functionality provided with IMS 12 SPE APARs
 - APAR PM91898 (TM RA V13.2.0)
 - Note: IMS Connect V13 PM91312 and IMS V13 PM85849 are required for this function to work properly.
 - Both APARs can be applied independently to TMRA and IMS Connect, however, both APARs must be applied to use the password phrase feature

Summary

- RACF password phrases can now be used with
 - IMS TMRA
 - IMS Connect
 - /SIGN command
 - VTAM logon user data

- Benefits of using RACF password phrases
 - More robust
 - Up to 100 bytes
 - Can contain mixed-case letters, numbers and special characters
 - Easier to remember

Diagnose Command Enhancements

Let's now explore the enhancements made to the type-1 /DIAGNOSE command in IMS 13.

Diagnose Command Enhancements – Highlights

- **Process for capturing diagnostic data used in troubleshooting IMS issues has been simplified**
 - SYSOUT option now available for /DIAGNOSE SNAP output
 - Documentation can be gathered/stored in a readable format that is easy to retrieve
 - Time-consuming SYSLOG searches and manual data formatting prior to transmission no longer required
 - /DIAGNOSE SNAP command extended to include more resources + more coverage of existing resources

- **Benefits**
 - Cost effective, non-disruptive alternative to console dumps
 - /DIAGNOSE command is now more interactive
 - Can be used more as a tool for easing the real-time diagnosis process
 - Decreased time and effort required in capturing diagnostic information
 - Improved turn-around time in problem resolution

SYSOUT Output Formatting Routine – Support Added

- Users can now send formatted /DIAGNOSE SNAP command output to a SYSOUT data set, enabling easy submission to IBM support
 - Potential users include IMS application developers, DBAs, sysprogs, support technicians, and/or developers
- SYSOUT data set will contain documentation that is
 - Formatted and readable (no more manual copying/reformatting of MVS SYSLOG or cutting/pasting into a text file)
 - Easy to retrieve (no more lengthy SYSLOG searches)
- New OPTION(SYSOUT) parameter added to /DIAGNOSE SNAP command
 - Contain sub-parameters that manage output processing
 - Let's take a look at these now...

/DIAG SNAP...OPTIONS(SYSOUT) Sub-Parameters

- **CLASS()** specifies an output class for the SYSOUT data set
 - Any valid JES output class can be specified; must be A-Z and 0-9
 - Default: IMS control region's assigned class
 - Example: `/DIAGNOSE SNAP REGION(1) SHOW(ALL) OPTION(SYSOUT,CLASS(S))`

- **LIMIT()** specifies a limit for the # of lines of formatted SNAP data to process in response to the command; similar to `OPTION(DISPLAY LIMIT())`
 - Valid range: 1 – 99999
 - Defaults to 19,999

- **FORMAT()** specifies format of the output produced (also now applies to `OPTION(DISPLAY)`)
 - `FORMAT(LONG)` is default and produces a complete display -- block name, description, location, dump of storage for block
 - `FORMAT(LOCATION)` produces only block name, description and location

Example Showing OPTION(SYSOUT) Messages and Data Set Output**/DIAGNOSE SNAP LTERM(CTRL) OPTION(SYSOUT) – Messages**

```

DFS3789I DIAGNOSE COMMAND SNAP LTERM QUEUED TO SYSOUT TKN(00000003) SYS3
DFS058I 07:46:03 DIAGNOSE CMMAND COMPLETED SYS3
DFS3788I DIAGNOSE SYSOUT DATA SET USRT001.IMS1.JOB00077.D0000105.? OPENED
FOR SNAP LTERM TKN(00000003) SYS3
DFS3788I DIAGNOSE SYSOUT DATA SET USRT001.IMS1.JOB00077.D0000105.? CLOSED
FOR SNAP LTERM TKN(00000003) SYS3

```

Figure 202: /DIAGNOSE SNAP LTERM(CTRL) OPTION(SYSOUT) – Messages

/DIAGNOSE SNAP LTERM(CTRL) OPTION(SYSOUT) – Output

```

/DIAGNOSE SNAP STORAGE DISPLAY

Resource: LTERM(CTRL)

CNT      Communication Name Table (Trgt)  Loc: 0C68B0F8
-----
0000 8C6D47CC 00000000 00000000 00000003 |.....|
0010 00000000 12820077 00000004 C3E3D9D3 |.....b.....CTRL|
0020 40404040 40000001 0C6D4890 00000000 |.....|
0030 00000000 00000000 0AB07E04 00000000 |.....|=.....|
0040 0C68B170 00000000 0C6D4820 00000000 |.....|
0050 000A000A 00000000 00000000 00000000 |.....|
0060 00000000 00000000 00000004 00000000 |.....|
*12262/074603*

```

SYSOUT Output Formatting Routine – Support Added

- Value
 - Reduced time/effort involved in diagnostic data capture process
 - No more formatting and having to copy/paste into text file
 - No more searching SYSLOG for desired data
 - Fits with automated diagnostic data capture
 - Helpful for intermittent problems
 - Facilitates historical tracking of a resource

/DIAGNOSE SNAP LTERM(), NODE(), USER() – SHOW() Support Added

- Users can now use the SHOW() parameter when gathering diagnostic information for logical terminals, nodes, and users (previously, no filtering mechanism available), just like other resource types
 - SHOW(PRI) – Captures the primary control blocks for the resource and is **default**
 - SHOW(ALL) – Captures all control blocks available for the resource
 - SHOW(OPT) – Captures all optional control blocks for the resource
 - SHOW(blockname) – Captures the specified block by block name
 - SHOW(blockname,blockname) – Captures multiple block names separated by commas
- More SHOW() values that only apply to logical terminals, nodes and users
 - SHOW(RECANY) captures the RAQE and RAQERES blocks
 - SHOW(SA) is a synonym for SHOW(SAVEAREA) and captures the SAVEAREA block
 - The above two SHOW() parameters can also be used with LINE() and LINK()
 - SHOW(DEF) captures the default, primary blocks for a logical terminal; same functionality as SHOW(PRI) – for use with LTERM()
 - SHOW(TAR) captures the target CNT for a local logical terminal – for use with LTERM()

05 System Part 2: 101

The SHOW() parameters shown in the first bullet are not new in IMS 13. What is new is that they can now be used with a /DIAG SNAP command that captures information for a logical terminal, node or user. What is the difference between a “primary” and “optional” control block? A primary control block “represents” a resource in that it is the primary/main block for the resource. Primary blocks are returned by a known and trusted source (like find dest) so we “trust” that the address and storage are valid (if the address/storage was bad, find dest fails first). All other blocks are considered optional. Optional blocks are “not trusted” in that an optional block is never directly referenced. Instead we use the copy routine and its ESTAE to copy the block to a known location safely and then reference fields in the block from the copy.

The second bullet lists the SHOW() parameters that are new in IMS 13 and that only apply to capturing information for a logical terminal, node or user. SHOW(RECANY) will display the VTAM receive in/out buffer control blocks (originated from a customer requirement). The SHOW(SA) is a shortcut that can be used for capturing the save area set control block (equivalent to SHOW(SAVEAREA)), and the remaining two new parms, SHOW(DEF) and SHOW(TAR) are for use with LTERMS only and are described in the visual.

/DIAGNOSE SNAP LTERM(), NODE(), USER() – SHOW() Support Added

- Several new control blocks now able to be captured (highlighted later)
- Value
 - Ensures consistent control over the type and amount of data produced by the command for these resources
 - Reduces overall time and effort in analyzing/solving problems
 - Potential users include IMS sysprogs, support technicians, and/or developers
- Examples
 - `/DIAGNOSE SNAP LTERM(ltermname) SHOW(blockname)`
 - `/DIAGNOSE SNAP NODE(nodename) SHOW(blockname)`
 - `/DIAGNOSE SNAP USER(username) SHOW(blockname)`

05 System Part 2: 102

Not only can the LTERM(), NODE(), and USER() resources now use the SHOW() filter with the /DIAG command to narrow the diagnostic data that is captured, but they can also have additional control blocks captured compared to previous IMS releases. The value of these enhancements that we've just discussed enable users to have consistent control over the type and amount of data being produced for resource involved in the /DIAG SNAP command. With this improved level of control, the overall amount of time and effort required in capturing diagnostic data has been reduced. Potential users of the enhancements just covered include IMS system programmers, support technicians as well as developers.

A few examples of the /DIAG SNAP input command are shown here; remember that the Appendix shows command output detail for certain command flavors.

/DIAGNOSE SNAP LTERM(), NODE(), USER() – RM() Support Added

- Users can now optionally use the RM() parameter to specify the scope of the search when gathering diagnostic information for nodes, logical terminals and users
 - RM(YES) is the default and performs a local search to find and use local resource copy
 - If resource not found locally, a search will be performed on RM structure to find and use global resource copy
 - RM(NO) performs only a local search to find the resource
 - RM(ONLY) performs only a global search on RM structure
 - If RM not available, message issued:
`DFS2859I DIAGNOSE COMMAND FAILED - RM(ONLY) INVALID, RM UNAVAILABLE`
- Value
 - Allows support technicians to view global resources alone without including local resource information (and vice versa), pinpointing desired information
 - Reduces search overhead with more narrow search scope
 - Reduces overall time and effort in analyzing/solving problems

05 System Part 2: 103

Another way the /DIAG command was enhanced for the LTERM(), NODE(), and USER() resources is that it can now include the RM() parameter to specify the search scope for the resources involved in the command processing. A user can now choose to capture diagnostic data for only a local resource, only a global resource or for both. The parameter values associated with each of these search scopes are shown on the slide. Note that RM(YES) is the default. In this case if the resource is not found locally, the RM resource structure will be searched in order to find a global resource for use in the diagnostic data capture. In any scenario that RM is requested but is unavailable, an error message will be issued to this effect and is shown on the slide.

The value of this new parameter is that it allows users to more efficiently designate the search scope for the data being captured, while providing increased flexibility. Because users can now exclusively capture diagnostic data for a local resource versus a global resource (and vice versa), they can compare the local copy with the global copy to determine whether they match. Since users are now able to more efficiently capture the diagnostic data that they are specifically interested in, overall time/effort expended in problem analysis is reduced.

/DIAGNOSE SNAP BLOCK() – Scope Expanded

- Users can now specify multiple single instance blocks on the /DIAG SNAP BLOCK command
 - Previously, only one single instance block could be specified at a time (or ALL)

- Value
 - Gives user more control over which blocks should be captured
 - Ensures consistency in capturing diagnostic data
 - Reduces overall time and effort in analyzing/solving problems
 - Potential users include IMS sysprogs, support technicians, and/or developers

- Example of capturing information for the Log Control Directory control blocks (Journal and Monitor)
 - `/DIAGNOSE SNAP BLOCK(LCD,LCDM)`

The scope of the /DIAGNOSE SNAP BLOCK() command has been expanded in that users can now include multiple single block instances

/DIAGNOSE SNAP BLOCK() – More Blocks Added

- Additional blocks can now be specified as BLOCK() parameter keywords, new blocks highlighted here with an asterisk...

Block Name	Block Description
ALL	Captures information for all valid control blocks currently available.
CATA*	Captures information for the Catalog Anchor Block control block. The CATA is available only in a DB/DC or DBCTL environment. If the CATA is requested in a DCCTL environment, a DFS110I error message will be issued in response. This block is ignored when the ALL option is specified in a DCCTL environment.
CMDE	Captures information for the Commands SCD Extension control block.
CSCD	Captures information for the APPC/OTMA SMQ SCD Extension control block. The CSCD is available only in a DB/DC or DCCTL environment. If the CSCD is requested in a DBCTL environment, a DFS110I error message will be issued in response. This block is ignored when the ALL option is specified in a DBCTL environment.
CSLA*	Captures information for the Common Service Layer Anchor Block control block.
DCCB*	Captures information for the Data Communications control block. The DCCB is available only in a DB/DC or DCCTL environment. If the DCCB is requested in a DBCTL environment, a DFS110I error message will be issued in response. This block is ignored when the ALL option is specified in a DBCTL environment.
DFA*	Captures information for the Definition Anchor Block control block.

05 System Part 2: 105

/DIAGNOSE SNAP BLOCK() – More Blocks Added

- Additional blocks can now be specified as BLOCK() parameter keywords, new blocks highlighted here with an asterisk...

Block Name	Block Description
DGA*	Captures information for the Diagnostics Anchor Block control block.
DGSD*	Captures information for the Diagnostic Data Set Structures control block.
DGSW*	Captures information for the Diagnose Work Area Storage control block.
EDBT*	Captures information for the RSR FP Global DB Tracking control block. The EDBT is available only in an IMS system where Fast Path is defined. If the EDBT is requested in a non-FP environment, a DFS154I error message will be issued in response. This block is ignored when the ALL option is specified in a non-FP environment.
ESCD*	Captures information for the Extended System Contents Directory control block. The ESCD is available only in an IMS system where Fast Path is defined. If the ESCD is requested in a non-FP environment, a DFS154I error message will be issued in response. This block is ignored when the ALL option is specified in a non-FP environment.
GDBT*	Captures information for the System RSR Global DB Tracking control block. The GDBT is available only in a DB/DC or DBCTL environment. If the GDBT is requested in a DCCTL environment, a DFS110I error message will be issued in response. This block is ignored when the ALL option is specified in a DCCTL environment.
LCD*	Captures information for the Log Control Directory (Journal) control block.

05 System Part 2: 106

/DIAGNOSE SNAP BLOCK() – More Blocks Added

- Additional blocks can now be specified as BLOCK() parameter keywords, new blocks highlighted here with an asterisk...

Block Name	Block Description
LCDM	Captures information for the Log Control Directory (Monitor) control block.
LSCD	Captures information for the APPC SCD Extension control block. The LSCD is available only in a DB/DC or DCCTL environment. If the LSCD is requested in a DBCTL environment, a DFS110I error message will be issued in response. This block is ignored when the ALL option is specified in a DBCTL environment.
MWA	Captures information for the Modify Work Area control block.
QSCD	Captures information for the Queue Manager SCD Extension control block. The QSCD is available only in a DB/DC or DCCTL environment. If the QSCD is requested in a DBCTL environment, a DFS110I error message will be issued in response. This block is ignored when the ALL option is specified in a DBCTL environment.
RECA*	Captures information for the VTAM Receive Any I/O Buffers control blocks. The RECA is available only in a DB/DC or DCCTL environment. If the RECA is requested in a DBCTL environment, a DFS110I error message will be issued in response. This block is ignored when the ALL option is specified in a DBCTL environment. Due to the volume of data that can be produced snapping the VTAM Receive Any I/O Buffers, the RECA option is not included when processing the BLOCK(ALL) option.

/DIAGNOSE SNAP BLOCK() – More Blocks Added

- Additional blocks can now be specified as BLOCK() parameter keywords, new blocks highlighted here with an asterisk

Block Name	Block Description
RSR*	Captures information for the Remote Site Recovery Anchor control block.
SCD	Captures information for the System Contents Directory control block.
SDTT*	Captures information for the Shutdown Trace Table control block.
SQM	Captures information for the Shared Queues Master Control Block control block. The SQM is available only in an IMS system where Shared Queues is defined. If the SQM is requested in a non-SQ environment, a DFS154I error message will be issued in response. This block is ignored when the ALL option is specified in a non-SQ environment.
TIME*	Captures information for the Timer Services SCD Extension control block.
TRA*	Captures information for the Table Trace control block.
TSCD	Captures information for the OTMA SCD Extension control block. The TSCD is available only in a DB/DC or DCCTL environment. If the TSCD is requested in a DBCTL environment, a DFS110I error message will be issued in response. This block is ignored when the ALL option is specified in a DBCTL environment.

/DIAGNOSE SNAP DB() – Support for All Blocks Added

- Users can now gather diagnostic information on all control blocks associated with a specified database with /DIAG SNAP DB() command
 - Previously, only the DDIR control block could be captured
- Value
 - Ensures consistency and simplifies data capture process
 - Requires less overall time and effort in analyzing/solving problems
 - Potential users include IMS application developers, DBAs, sysprogs, support technicians, and/or developers
- Example
 - /DIAGNOSE SNAP DB (*dbname*) SHOW (ALL)

/DIAGNOSE SNAP DB() – Support for All Blocks Added

- Snap blocks common to all DB types
- Snap blocks specific to a particular DB type
 - Full Function
 - Fast Path
- List of blocks able to be captured depends on DB type

Example Showing Key Full Function DB() Control Blocks

```

→ /DIA SNAP DB(DI41M101) SHOW(PSDB,SDB,FDB)
DFS4445I CMD FROM MCS/E-MCS CONSOLE USERID=ZS1CMST1: DIA SNAP DB(DI41M10
1) SHOW(PSDB,SDB,FDB) IMS1
DFS4444I DISPLAY FROM ID=IMS1 003
/DIAGNOSE SNAP STORAGE DISPLAY

Resource: DB(DI41M101)

★ PSDB      Physical Segment Descriptor Blk      Loc: 0C2B8180
-----
0000 01000101 00000101 00020028 230F0000 |.....|
0010 0C2B85D4 0C28E6BC 00000000 00000028 |..e.m..w.....|
0020 00000000 00000001 |.....|

★ SDB      Segment Descriptor Blocks            Loc: 0C28E6BC
-----
0000 C1F1F1F1 F1F1F1F1 0104F410 20000001 |A1111111..4....|
0010 00680000 0A2C3D38 00000000 0C28E724 |.....X.....|
0020 0C2B8180 0A469358 0C28E4CC 00000000 |...a...l...U....|
0030 01010000 0A4693B4 00000000 00000000 |.....l.....|
0040 00000000 00000000 00000000 00010900 |.....|
0050 00FC0100 04000000 00000000 00000000 |.....|
0060 00000000 00000000 |.....|

★ FDB      Field Descriptor Block              Loc: 0C2B85D4
-----
0000 C1F1F1F1 F1F1F1F1 00004309 |A1111111....|

FDB      Field Descriptor Block              Loc: 0C2B85E0
-----
0000 F1F1F1F1 F1F1F1F1 000E8309 |11111111..c..|

★ PSDB      Physical Segment Descriptor Blk      Loc: 0C2B81A8
-----
0000 02010201 01000101 0002000A 230F000A |.....|
0010 0C2B85EC 0C28E724 00000000 0000000A |..e...X.....|
0020 00000000 00000001 |.....|

★ SDB      Segment Descriptor Blocks            Loc: 0C28E724
-----
0000 C1C1F2F2 F2F2F2F2 0204F410 20000001 |AA222222..4....|
0010 00680208 0A2C3D38 00000000 0C28E78C |.....X.....|
0020 0C2B81A8 0C28E6BC 0C28E4CC 00000000 |...ay..w...U....|
0030 01010100 0A4693BE 00000000 00000000 |.....l.....|
0040 00000000 00000000 00000000 00010700 |.....|

```

/DIAGNOSE SNAP MSNAME() – Support Added

- Users can now gather diagnostic information for all control blocks associated with a specified MSNAME (logical link path) using /DIAG SNAP MSNAME
 - Previously, no support existed
 - Includes all standard TM blocks and SHOW() option for filtering
 - SHOW(PRI)
 - SHOW(blockname)
 - SHOW(ALL)
 - SHOW(OPT)
- Value
 - Ensures consistency and simplifies data capture process
 - Requires less overall time and effort in analyzing/solving problems
 - Potential users include IMS sysprogs, support technicians, and/or developers
- Example
 - `/DIAGNOSE SNAP MSNAME (msname) SHOW(blockname)`

/DIAGNOSE SNAP LINE() – Support for More Blocks Added

- Users can now gather diagnostic information for more control blocks associated with a communication line
- Special SHOW() parameter keywords available in addition to common keywords covered earlier
 - *SHOW(RECANY) captures the RAQE and RAQERES blocks*
 - *SHOW(SA) is a synonym for SHOW(SAVEAREA) and captures the SAVEAREA block*
- All blocks are shown on the next slide with the new ones highlighted with an asterisk

More control blocks can now be captured for a communication line with the /DIAG SNAP command issued for the LINE() resource. In addition, the new SHOW() parameter values introduced earlier apply to the LINE() resource. Users can now filter the captured diagnostic data by VTAM Receive Any Input and Output buffer control blocks with the SHOW(RECANY) parameter, and can also use SHOW(SA) to capture the save area control block. Let's now take a look at the control blocks that can be captured with the /DIAGNOSE SNAP LINE() command.

/DIAGNOSE SNAP LINK() – Support for More Blocks Added

- Users can now gather diagnostic information from additional control blocks associated with a logical link
- Special SHOW() parameter keywords available in addition to common keywords covered earlier
 - *SHOW(SA) is a synonym for SHOW(SAVEAREA) and captures the SAVEAREA block*
- All blocks are shown on the next slide with the new ones highlighted with an asterisk

05 System Part 2: 114

More control blocks can now be captured for a logical link with the /DIAG SNAP command issued for the LINK() resource. In addition, the new SHOW() parameter value introduced earlier apply to the LINK() resource: users can now filter the captured diagnostic data by save area control block using the SHOW(SA) filter. Let's now take a look at the control blocks that can be captured with the /DIAGNOSE SNAP LINK() command.

New Messages Added

DFS3613I	DGS TCB INITIALIZATION COMPLETE
DFS2857E	DIAGNOSE COMMAND INTERNAL ERROR - MOD=modname RSN=nnnn
DFS2858E	DIAGNOSE COMMAND SEVERE ERROR - reason text
DFS2859I	DIAGNOSE COMMAND UNSUCCESSFUL - reason text
DFS3785E	DIAGNOSE AWE INITIALIZATION FAILED - reason text
DFS3786E	DIAGNOSE AWE PROCESSING ERROR - reason text
DFS3787E	DIAGNOSE SYSOUT PROCESSING ERROR - reason text
DFS3788I	DIAGNOSE SYSOUT DATA SET dsname action FOR SNAP resource TKN(token)
DFS3789I	DIAGNOSE COMMAND SNAP resource QUEUED TO SYSOUT TKN(token)

Summary

- SYSOUT option now available for /DIAGNOSE SNAP output
- /DIAGNOSE SNAP command extended to include more resources + more coverage of existing resources
 - Now supports over 300 blockname options across all resource types
- Benefits
 - Process for capturing diagnostic data used in troubleshooting IMS issues has been simplified
 - Decreased time and effort required in capturing diagnostic information
 - Improved turn-around time in problem resolution
 - Cost effective, non-disruptive alternative to console dumps

Summary

■ Benefits (cont'd)

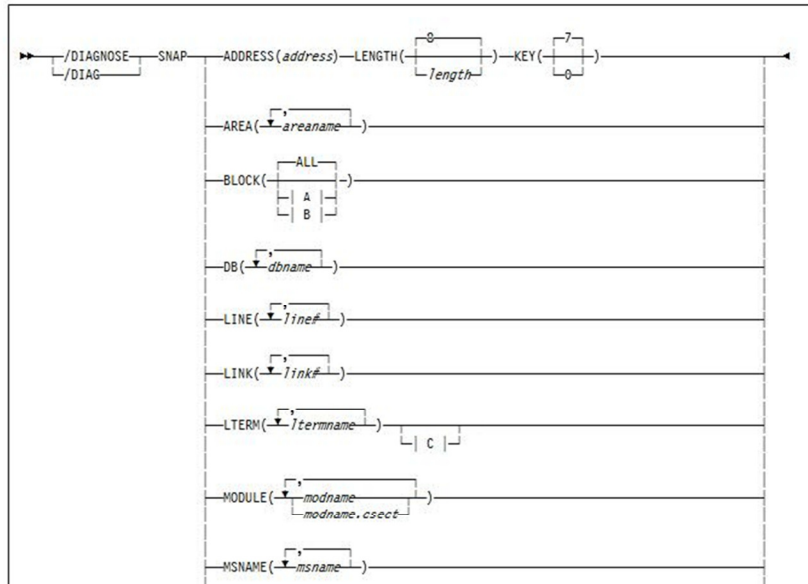
- SYSOUT option
 - Documentation can be gathered/stored in a readable format that is easy to retrieve
 - Time-consuming SYSLOG searches and manual data formatting prior to transmission no longer required
 - Fits with automated diagnostic data capture
 - Helpful for intermittent problems
 - Facilitates historical tracking of a resource

- /DIAG command expanded to include more resources and capture additional blocks for existing supported resources
 - Search overhead reduced by allowing user to narrow search scope with RM()
 - Now more interactive and can be used more as a tool for easing the real-time diagnosis process

Appendix

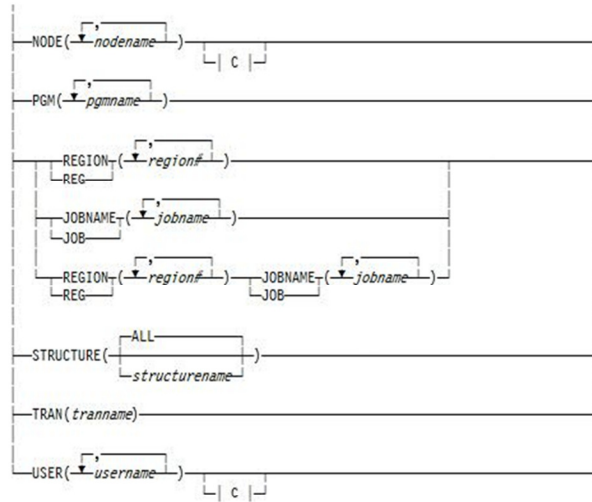
Command Syntax Diagram

Command Syntax

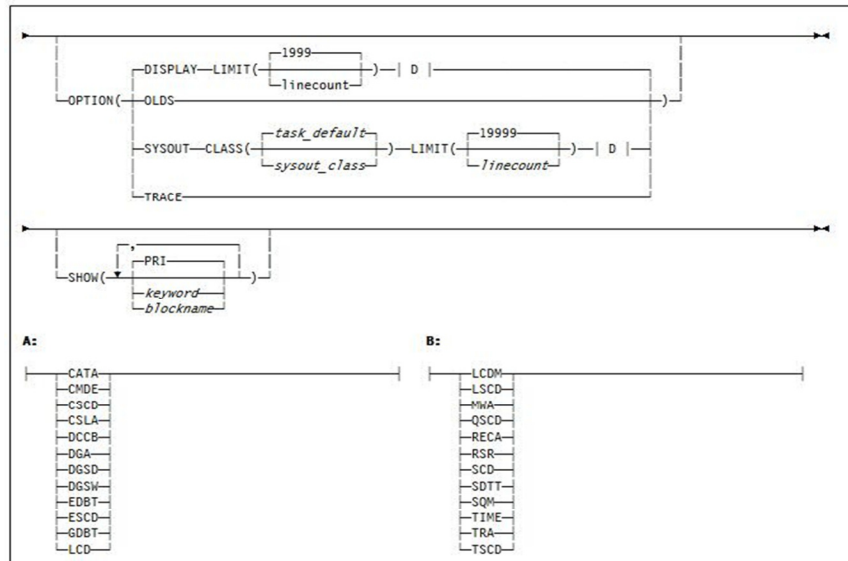


05 System Part 2: 120

Command Syntax



Command Syntax



Command Syntax

C:

RM(

YES
NO
ONLY

)

D:

FORMAT(

LONG
LOC
LOCATION

)

Alphabetical Sorting of Resource Control Block Names

LTERM() Blocks...

Name	Block Description	Macro	Primary	Optional
CNT	Communication Name Table (Trgt)	ICLI	•*	
LNB	Line Name Block	LNB	•**	
RCNT	Remote Communication Name Table	RCNT	•**	
CCB	Conversational Control Block	ICLI		•
CTB	Communication Interface Block	ICLI		•
CLB	Communication Line Block	ICLI		•
CNT	Communication Name Table	ICLI		•
CRB	Communications Restart Block	ICLI		•
CRTCN	Command CART & 4-Byte Console ID	DFSMCSC		•
CTB	Communication Terminal Block	ICLI		•
CTT	Communication Translate Table	ICLI		•
CULE	Common Use List Element Block	DFSCULE		•

* Primary blocks for a local logical terminal
 ** Primary blocks for a remote logical terminal

05 System Part 2: 125

LTERM() Blocks...

Name	Block Description	Macro	Primary	Optional
DSPWRK1	Dispatcher Work Area	IDSPWRK		•
ECNT	Extended Communication Name Table	DBFECNT		•
EMHB	Expedited Message Handler Block	DBFEMHB		•
EPF	Event Control Block Prefix	IEPF		•
INBUF	Input Line Buffer			•
MSGBP	Basic 01/03 Message Prefix	QLOGMSGP		•
OUTBUF	Output Line Buffer			•
PROLOG	Module Prolog Information			•
RAQE	VTAM Receive Any IO Buffer (In)	BUFVTPRE		•
RAQERES	VTAM Receive Any IO Buffer (Out)	BUFVTPRE		•
SAP	Save Area Prefix	ISAP		•
SAVEAREA	Save Area Set	REQUATE		•

***LTERM()* Blocks**

Name	Block Description	Macro	Primary	Optional
SMB	Scheduler Message Block	IAPS		•
SPQB	Subpool Queue Block	ICLI		•
SPQBEXT	Subpool Queue Extension Block	ICLI		•
TIB	APPC Transaction Instance Block	DFSTIB		•
UOWE	Unit of Work Table Entry	DFSUOWE		•
YTIB	OTMA Transaction Instance Block	DFSYTIB		•

NODE() Blocks...

Name	Block Description	Macro	Primary	Optional
CLB	Communication Line Block	ICLI	•	
EPF	Event Control Block Prefix	IEPF	•	
CCB	Conversational Control Block	ICLI		•
CIB	Communication Interface Block	ICLI		•
CNT	Communication Name Table	ICLI		•
CRB	Communications Restart Block	ICLI		•
CRTCN	Command CART & 4-Byte Console ID	DFSMCSC		•
CTB	Communication Terminal Block	ICLI		•
CTT	Communication Translate Table	ICLI		•
CULE	Common Use List Element Block	DFSCULE		•
DSPWRK1	Dispatcher Work Area	IDSPWRK		•
ECNT	Extended Communication Name Table	DBFECNT		•
EMHB	Expedited Message Handler Block	DBFEMHB		•
INBUF	Input Line Buffer			•
MSGBP	Basic 01/03 Message Prefix	QLOGMSGP		•

05 System Part 2: 128

NODE() Blocks

Name	Block Description	Macro	Primary	Optional
OUTBUF	Output Line Buffer			•
PROLOG	Module Prolog Information			•
RAQE	VTAM Receive Any IO Buffer (In)	BUFVTPRE		•
RAQERES	VTAM Receive Any IO Buffer (Out)	BUFVTPRE		•
SAP	Save Area Prefix	ISAP		•
SAVEAREA	Save Area Set	REQUATE		•
SMB	Scheduler Message Block	IAPS		•
SPQB	Subpool Queue Block	ICLI		•
SPQBEXT	Subpool Queue Extension Block	ICLI		•
TIB	APPC Transaction Instance Block	DFSTIB		•
UOWE	Unit of Work Table Entry	DFSUOWE		•
YTIB	OTMA Transaction Instance Block	DFSYTIB		•

USER() Blocks...

Name	Block Description	Macro	Primary	Optional
SPQB	Subpool Queue Block	ICLI	•	
CCB	Conversational Control Block	ICLI		•
CIB	Communication Interface Block	ICLI		•
CLB	Communication Line Block	ICLI		•
CNT	Communication Name Table	ICLI		•
CRB	Communications Restart Block	ICLI		•
CRTCN	Command CART & 4-Byte Console ID	DFSMCSC		•
CTB	Communication Terminal Block	ICLI		•
CTT	Communication Translate Table	ICLI		•
CULE	Common Use List Element Block	DFSCULE		•
DSPWRK1	Dispatcher Work Area	IDSPWRK		•
ECNT	Extended Communication Name Table	DBFECNT		•
EMHB	Expedited Message Handler Block	DBFEMHB		•
EPF	Event Control Block Prefix	IEPF		•

05 System Part 2: 130

USER() Blocks

Name	Block Description	Macro	Primary	Optional
INBUF	Input Line Buffer			•
MSGBP	Basic 01/03 Message Prefix	QLOGMSGP		
OUTBUF	Output Line Buffer			
PROLOG	Module Prolog Information			•
RAQE	VTAM Receive Any IO Buffer (In)	BUFVTPRE		•
RAQERES	VTAM Receive Any IO Buffer (Out)	BUFVTPRE		•
SAP	Save Area Prefix	ISAP		•
SAVEAREA	Save Area Set	REQUATE		•
SMB	Scheduler Message Block	IAPS		•
SPQBEXT	Subpool Queue Extension Block	ICLI		•
TIB	APPC Transaction Instance Block	DFSTIB		•
UOWE	Unit of Work Table Entry	DFSUOWE		•
YTIB	OTMA Transaction Instance Block	DFSYTIB		•

DB() Common Blocks

Name	Block Description	Macro	Primary	Optional
DDIR	Database Directory Block	DFSDDIR	.	
DDIREXT	DDIR Extension	DFSDDIR		.
RSCX	Resource Extension Block	DFSRSCX		.
DYNALMBR	Dynamic Allocate Member	DFSMDA		.
DBQLE	Database Quiesce List Entry	DFSDBQQL		.
EEQE	Error Queue Element	DFSEEQE		.
RRE	Residual Recovery Element	DFSRRE		.
TDBC	Tracking Data Base Control	DFSTDBC		.
SDTE	Segment Delta Block Table Entry	DFS5FLDD		.

05 System Part 2: 132

DB() Full Function Blocks

Name	Block Description	Macro	Primary	Optional
DMB	Data Management Block	DFSDMB		•
PSDB	Physical Segment Descriptor Block	DFSDMB		•
SDB	Segment Descriptor Block	DFSSDBM		•
FDB	Field Descriptor Block	DFSFDB		•
DMBCPAC	Segment Edit/Compression Block	DFSDMB		•
DMBSEC	DMB Secondary List	DFSDMB		•
DMBDACS	Randomizer Control Block	DFSDMB		•
DMBAMPPR	Access Method Prefix Block Prefix	DFSDMB		•
DMBAMP	Access Method Prefix Block	DFSDMB		•
DCBACBP	Primary DCB/ACB Block	DCBD		•
DCBACBS	Secondary DCB/ACB Block	DCBD		•
DMBXBLCK	Data Management Exit Block	DFSDMB		•
DMBXARRY	Exit Array Entry Block	DFSDMB		•
DMBXT	Exit Description Block	DFSDMB		•

DB() Fast Path Blocks

Name	Block Description	Macro	Primary	Optional
DMCB	DEDB Master Control Block	DBFDMCB		•
BHDR	MSDB Header	DBFBMSDB		•

MSNAME() Blocks...

Name	Block Description	Macro	Primary	Optional
LNB	Link Name Block (Trgt)	LNB	•	→
CIB	Communication Interface Block	ICLI		•
CRB	Communications Restart Block	ICLI		•
CRTCN	Command CART & 4-Byte Console ID	DFSMCSC		•
CTT	Communication Translate Table	ICLI		•
CULE	Common Use List Element Block	DFSCULE		•
DSPWRK1	Dispatcher Work Area	IDSPWRK		•
ECNT	Extended Communication Name Table	DBFECNT		•
EMHB	Expedited Message Handler Block	DBFEMHB		•
EPF	Event Control Block Prefix	IEPF		•
INBUF	Input Line Buffer			•
LCB	Link Control Block	LCB		•
LLB	Link Line Block	ICLI		•

MSNAME() Blocks

Name	Block Description	Macro	Primary	Optional
LNB	Link Name Block	ICLI		•
LTB	Link Terminal Block	ICLI		•
LXB	Link Extension Block	LXB		•
MSGBP	Basic 01/03 Message Prefix	QLOGMSGP		•
OUTBUF	Output Line Buffer			•
PROLOG	Module Prolog Information			•
SAP	Save Area Prefix	ISAP		•
SAVEAREA	Save Area Set	REQUATE		•
SMB	Scheduler Message Block	IAPS		•
SPQB	Subpool Queue Block	ICLI		•
SPQBEXT	Subpool Queue Extension Block	ICLI		•
TIB	APPC Transaction Instance Block	DFSTIB		•
UOWE	Unit of Work Table Entry	DFSUOWE		•
YTIB	OTMA Transaction Instance Block	DFSYTIB		•

LINE() Blocks...

Name	Block Description	Macro	Primary	Optional
EPF	Event Control Block Prefix	IEPF	•	
CLB	Communication Line Block	ICLI	•	
CCB	Conversational Control Block	ICLI		•
CIB	Communication Interface Block	ICLI		•
CNT	Communication Name Table	ICLI		•
CRB	Communications Restart Block	ICLI		•
CRTCN	Command CART & 4-Byte Console ID	DFSMCSC		•
CTB	Communication Terminal Block	ICLI		•
CTT	Communication Translate Table	ICLI		•
CULE	Common Use List Element Block	DFSCULE		•
DSPWRK1	Dispatcher Work Area	IDSPWRK		•
ECNT	Extended Communication Name Table	DBFECNT		•
EMHB	Expedited Message Handler Block	DBFEMHB		•

LINE() Blocks

Name	Block Description	Macro	Primary	Optional
INBUF	Input Line Buffer			•
MSGBP	Basic 01/03 Message Prefix	QLOGMSGP		•
OUTBUF	Output Line Buffer			•
PROLOG	Module Prolog Information			•
RAQE	VTAM Receive Any IO Buffer (In)	BUFVTPRE		•
RAQERES	VTAM Receive Any IO Buffer (Out)	BUFVTPRE		•
SAP	Save Area Prefix	ISAP		•
SAVEAREA	Save Area Set	REQUATE		•
SMB	Scheduler Message Block	IAPS		•
SPQB	Subpool Queue Block	ICLI		•
SPQBEXT	Subpool Queue Extension Block	ICLI		•
TIB	APPC Transaction Instance Block	DFSTIB		•
UOWE	Unit of Work Table Entry	DFSUOWE		•
YTIB	OTMA Transaction Instance Block	DFSYTIB		•

05 System Part 2: 138

LINK() Blocks...

Name	Block Description	Macro	Primary	Optional
EPF	Event Control Block Prefix	IEPF	•	
LLB	Link Line Block	ICLI	•	
CCB	Conversational Control Block	ICLI		•
CIB	Communication Interface Block	ICLI		•
CRB	Communications Restart Block	ICLI		•
CRTCN	Command CART & 4-Byte Console ID	DFSMCSC		•
CTT	Communication Translate Table	ICLI		•
CULE	Common Use List Element Block	DFSCULE		•
DSPWRK1	Dispatcher Work Area	IDSPWRK		•
ECNT	Extended Communication Name Table	DBFECNT		•
EMHB	Expedited Message Handler Block	DBFEMHB		•
				•
INBUF	Input Line Buffer			•
LCB	Link Control Block	LCB		•

LINK() Blocks

Name	Block Description	Macro	Primary	Optional
LNB	Link Name Block	ICLI		•
LTB	Link Terminal Block	ICLI		•
LXB	Link Extension Block	LXB		•
MSGBP	Basic 01/03 Message Prefix	QLOGMSGP		•
OUTBUF	Output Line Buffer			•
PROLOG	Module Prolog Information			•
SAP	Save Area Prefix	ISAP		•
SAVEAREA	Save Area Set	REQUATE		•
SMB	Scheduler Message Block	IAPS		•
SPQB	Subpool Queue Block	ICLI		•
SPQBEXT	Subpool Queue Extension Block	ICLI		•
TIB	APPC Transaction Instance Block	DFSTIB		•
UOWE	Unit of Work Table Entry	DFSUOWE		•
YTIB	OTMA Transaction Instance Block	DFSYTIB		•