



IMS 13

# *IMS 13 Application Programming*

Information Management software

© 2014 IBM Corporation

## ***Application Programming Enhancements***

- Synchronous Program Switch
- SSA Qualify by Position and Length
- IMS Universal Driver Enhancements
- IMS Native SQL support for COBOL

## Synchronous Program Switch

## ***Synchronous Program Switch Topics***

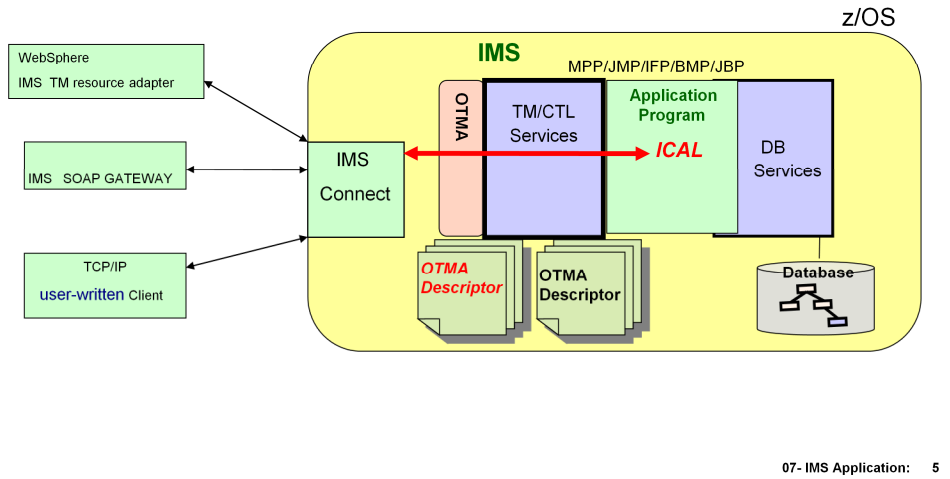
- Background
- Highlights
- Restrictions
- Detail
- Exit Routine Enhancements
- Application Considerations
  - Transaction Expiration
  - Late Responses
  - LTERM Override
  - ALTPCB destinations
  - Examples
- Migration/Setup

The topics on this visual show the comprehensive support IMS 13 has added for this new application model.



## Background

- DL/I ICAL support from previous IMS releases
  - Provided synchronous callout capability to resources **outside** IMS

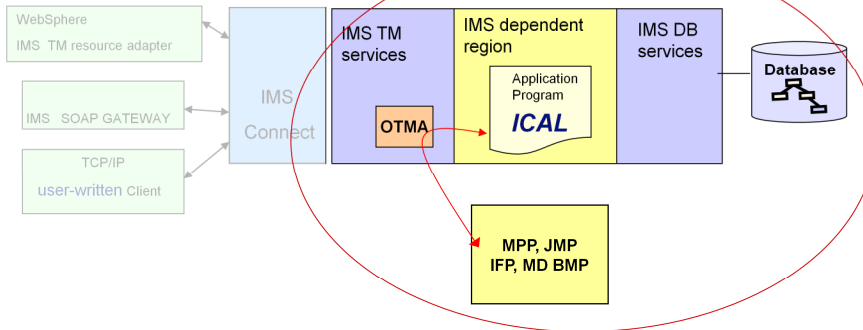


The existing DL/I ICAL in IMS allows IMS applications to synchronously call out to external resources. Specifically, IMS applications can request access to and wait for replies from:

- WebSphere EJBs/MDBs using the IMS TM Resource Adapter
- To any Web Service Provider using the IMS SOAP Gateway
- User-written IMS Connect clients

## Synchronous Program Switch

- **New** capability that enhances the DL/I ICAL support
  - Allows an IMS application program to *synchronously* call and wait for a reply from *another IMS application program*
    - Within the calling program's UOR
    - Called program is a separate UOR

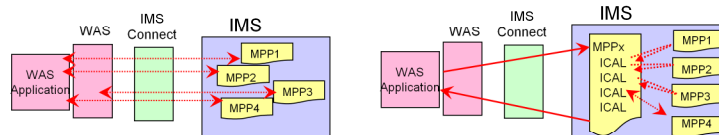


With IMS V13, a new capability is added to the DL/I ICAL support so that not only is access to resources outside IMS supported, but also the ability to synchronously call another IMS transaction running in any of the dependent region types. The called or target program can be an MPP, JMP, IFP, or Message-Driven BMP in the same or different IMS.

## Synchronous Program Switch...

### ■ Benefits

- Modernization of the IMS application infrastructure
  - Provides an internal service flow of IMS transactions to complete a business process
    - In the same IMS or a different IMS
- Implementation of a Process Server or Broker inside IMS
  - Reduces unnecessary network traffic when accessing multiple applications in the same IMS or IMSplex



07- IMS Application: 7

There are many benefits for IMS DB/TM and IMS TM environments that are looking to modernize their application infrastructure. Synchronous program switching provides many opportunities including:

- Modernization of the application infrastructure while continuing to use DL/I functionality to access various destinations
- Support for an internal service flow template that allows multiple IMS transactions to synchronously participate in a business process
- Encouragement of new IMS application development to support business logic on the host.
- Reduction of unnecessary network traffic with a broker implementation that uses synchronous program switching techniques

## Highlights

- **Automatic invocation of OTMA**
  - Without requiring OTMA to be defined or commands to be issued
  - New OTMA destination descriptor TYPE
    - IMSTRAN
- **Enhancements to the DL/I ICAL**
  - Allows an IMS transaction to be the target destination
    - Accepts multi-segment requests/responses
  - Provides additional AIB return and reason codes
- **Support for Late Reply messages**
  - Can be purged or rerouted
- **Security authorization**
  - Ensures userid of program issuing ICAL can access the target transaction

The highlights include:

- Automatic invocation of OTMA which is an inherent part of IMS.
- A new OTMA destination descriptor type of IMSTRAN which designates an IMS transaction as the target destination for both single and multi-segment ICAL messages.
- Extensions to the DL/I ICAL to provide additional AIB return and reason codes to provide information about errors that could be experienced in this new environment.
- Control of late replies. If the ICAL times out before its target transaction returns a reply, the late response message is automatically purged unless a reroute designation was previously defined in the destination descriptor.
- Support for security. The userid associated with the transaction issuing the ICAL is the same userid that is used to determine whether the target of the ICAL can be accessed.

### **Restrictions**

- No ICAL support for BMP or JBP applications running in DBCTL environments
  - ICAL is part of the IMS TM capability
  
- IMS application program issuing ICAL for a synchronous program switch
  - Can be a protected transaction
    - But the target transaction of the ICAL is not part of the RRS commit scope
  
- The switched-to program (target of the ICAL)
  - Has read-only access to the main storage data base (MSDB)
  - Cannot be an IMS Conversational transaction
  - Does not invoke IMS Message Format Service (MFS)

Note the upfront restrictions:

- ICAL is only supported in IMS TM or IMS TM/DB environments. BMPs or JBPs running in DBCTL are not able to issue the ICAL.
- The IMS application program issuing the ICAL can be a participant in a protected transaction but the target of the ICAL cannot be part of the RRS (resource recovery services) commit scope.
- Additionally, the IMS program that is invoked by the ICAL can read but not update MSDBs and cannot be an IMS conversational transaction. MFS services are never invoked.

## Restrictions ...

- **Exit routines**
  - DFSYIOE0 (OTMA Input/Output Edit Exit routine)
    - Not called when processing synchronous program switch messages and responses
      - Increases the transparency of using OTMA
    - Can be called for a **late response** being routed to an OTMA destination
      - To build or override the 1K OTMA user message data prefix in the response header
  - DFSBSEX0 (Build Security Environment Exit routine)
    - Not invoked for target transactions of the synchronous program switch
  - DFSMSCE0 (TM and MSC Message Routing and Control User exit routine )
    - Not invoked for the DL/I ICAL synchronous program switch
  
- **Use of new ICAL capability in a Shared Queues environment**
  - Requires all IMS systems to be at IMS 13 and DBRC MINVERS of 13.1

Restrictions involving exit routines include:

- The DFSYIOE0 (OTMA Input/Output Edit Exit routine) is not called for synchronous program switch messages and responses. It can be called, however, for a **late response** that is not purged but instead routed to an OTMA destination.

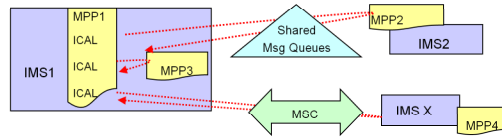
- The DFSBSEX0 (Build Security Environment Exit Routine) is not invoked for the target transaction of the synchronous program switch.

- Unlike the regular ISRT/CHNG call, DFSMSCE0 user exit will not be called for the synchronous program switch using the DL/I ICAL call.

IMS shared queues support requires all systems must be at IMS 13 and a DBRC MINVERS of 13.1.

## Synchronous Program Switch – The Details

- An enhancement to the DL/I ICAL to invoke another IMS application
  - In the same IMS
  - In a different IMS
    - In a Shared Queues back-end
    - Across an MSC link
  - And **synchronously** receive the response back during the same unit of recovery
- Where IMS internally schedules the transaction initiated by the ICAL call as an OTMA transaction
  - Uses a new type of OTMA destination descriptor (TYPE=IMSTRAN) which has been introduced specifically for synchronous program switch support
- And the target transaction can be
  - An IFP, MPP, MD BMP, or JMP in the IMS TM or TM/DB environments



07- IMS Application: 11

In IMS V13, the function of this DL/I ICAL has been extended to allow customers to call another IMS application, which can be in the same IMS, in the shared-queues back-end IMS, or in the remote IMS via MSC. The ICAL synchronously receives the response back during the same unit of work. This new function is called “Synchronous Program Switch.”

This new synchronous program switch function expands the usage of the OTMA destination descriptor to serve the ICAL requests so that the transaction data can be sent to another IMS application for processing. Multi-segment messages and responses are supported and the target of the call can be an IFP, MPP, BMP, JMP, or JBP running in the same or another IMS TM or IMS TM/DB system

IBM

## The DL/I ICAL call

**Same Format**

>>-ICAL--aib--request\_area--response\_area-----<<

Call Name	DB/DC	DBCTL	DCCTL	DB Batch	TM Batch
ICAL	X		X		

```

01 AIB.
02 AIBRID PIC x(8) VALUE 'DFS AIB ',
02 AIBALEN PIC 9(9) USAGE BINARY,
02 AIBSFUNC PIC x(8) VALUE 'SENDREC',
02 AIBRSNM1 PIC x(8) VALUE 'OTMDEST1',
02 AIBRSNM2 PIC x(8),
02 AIBRESV1 PIC x(8),
02 AIBOALEN PIC 9(9) USAGE BINARY VALUE 28,
02 AIBOAUSE PIC 9(9) USAGE BINARY VALUE 30,
02 AIBRSFLD PIC 9(9) USAGE BINARY VALUE 5000,
02 AIBRESV2 PIC x(8),
02 AIBRETRN PIC 9(9) USAGE BINARY,
02 AIBREASN PIC 9(9) USAGE BINARY,
02 AIBERRXT PIC 9(9) USAGE BINARY.
```

Request Data (example of multi-segment):

LLZZ+Trancode+Data ★

LLZZ+Data

LLZZ+Data

Response Data in multi-segment:

LLZZ+Data

LLZZ+Data

LLZZ+Data

07- IMS Application: 12

The overall structure of the DL/I ICAL continues to be the same as in previous releases. The functionality, however, has been expanded to allow the IMS application program to call another IMS application.

### request\_area

Specifies the request area to use for this call. This parameter is an input parameter. This request area contains the request message data that is sent from the IMS application program to the application that is specified in the OTMA descriptor. The AIBOALEN field specifies the length of the request message data. If the data is destined for a non-IMS application program or service that runs in a z/OS or distributed environment, the ICAL call will bypass IMS TM message queuing and the format of the request area does not require the LLZZ fields. If the data is destined for an IMS application program, the request data will require the LLZZ fields and the transaction code needs to be specified in the first 8 bytes of the data area following the LLZZ. For transactions specified with MULTSEG, the request data will need to include the entire segments. The standard IMS LLZZ format is required for each segment. The transaction code is only required in the first segment.

### response\_area

Specifies the response area to use for this call. This parameter is an output parameter. This response area should be large enough to hold the response that is returned from the application that is specified in the OTMA descriptor. If the response area is not large enough to contain all of the returned data, IMS returns partial data. When partial data is returned, the AIBOAUSE field contains the length of the returned data in the response area, and AIBOALEN contains the actual length of the response message. If the request data is destined for a non-IMS application program or service that runs in a z/OS or distributed environment, the format of the response area does not require the LLZZ fields. If the request data is destined for an IMS application program, the format of the response data will follow the standard LLZZ format for each segment in the response data area, and the response data area will include the entire output segment if the space has been defined as large enough.



## The DL/I ICAL Call ...

- Examples of common Return codes for synchronous program switch
  - Full list in the IMS documentation

Return Code (Hex)	Reason Code (Hex)	Extended Reason code (Hex)	Brief Description
0100	000C	0000	Partial data returned
0100	0100	0000	Error message returned
0100	0104	0020	ICAL timed out
0100	010C	0000	/PSTOP cmd issued
0100	0110	0000	Invalid transcode
0100	0110	0004	Security violation
0100	0110	0005	DFSYICAL stopped
0100	0110	0020	Input length invalid
0100	0110	0031	Trans is stopped
0100	0110	0061	Target trans does not insert back to IOPCB

- If the ICAL “SENDRECV” receives a partial data status
  - The new ICAL “RECEIVE” subfunction can retrieve the entire message
    - Discussed in the OTMA Enhancements section

07- IMS Application: 13

Additionally, new return codes, reason codes and extended reason codes for the ICAL are introduced to explain errors that could be encountered in this new type of interaction. The visual shows some examples but the full list is available in the IMS documentation.

Note that if the response\_area is too small and only partial data is returned, then the new ICAL with a sub-function of “RECEIVE” can be used to retrieve the entire message. Detailed information on the RECEIVE subfunction can be found in the OTMA Enhancements section.

## The DL/I ICAL Call ...

- Continues to use OTMA Destination Routing Descriptors
  - Which externalize the routing definitions and specifications for callout messages and synchronous program switch messages
  - Allowing up to 510 destination routing descriptor entries defined in DFSYDTx member of IMS.PROCLIB
- With new TYPE= IMSTRAN for synchronous program switches

D entry\_name keywords Where entry\_name is descriptor entry name and can be masked by ending in an \*

keywords are: TYPE=IMSTRAN

LTERMOVR=name  
 TMEMBER=name  
 TPIPE=name  
 SMEM=NO|YES  
 EXIT= NO|YES  
 REPLYCHK=YES|NO  
 SYNCTP=NO|YES  
 SYNTIMER=timeout value

For example: D OTMDEST1 TYPE=IMSTRAN SYNTIMER=500

D OTMDEST2 TYPE=IMSTRAN TMEMBER=SCOTTHWS1 TPIPE=BRYCE EXIT=YES

07- IMS Application: 14

The DFSYDTx member of IMS.PROCLIB has been enhanced to support a new type of destination descriptor, **IMSTRAN**, to perform the synchronous program switch via the DL/I ICAL call in the IMS application. They are read and loaded at IMS initialization.

The optional parameters include: TMEMBER, TPIPE, SMEM, SYNCTP, EXIT, LTERMOVR and REPLYCHK. The TMEMBER, TPIPE, SYNCTP, and SMEM parameters for this new type of descriptor can be used to specify the destination of the late response for the synchronous program switch request. The EXIT parameter allows the message control/error exit routine (DFSCMUX0) user exit to override the destination for the late response for the synchronous program switch. The LTERMOVR parameter can be specified to set the LTERM name of the IOPCB for the target transaction of the synchronous program switch. The REPLYCHK parameter can be optionally used when multiple response messages are competing to be sent back to the ICAL call.

These new descriptor type and the corresponding parameters for the synchronous program switch can be specified by using the DFSYDTx member of IMS PROCLIB or the type-2 commands

## OTMA destination routing descriptor for DL/I ICAL

### ■ TYPE= IMSTRAN

- Required for synchronous program switch requests

Supported parameters for TYPE=IMSTRAN are as follows:

**LTERMOVR**= Specifies a value to override the LTERM name in the target IMS application program's I/O PCB

**TMEMBER**= A 1- to 16-character OTMA TMEMBER name. Optional when TYPE=IMSTRAN. When specified, IMS queues the late response of a synchronous program switch to this OTMA TMEMBER (otherwise the late reply is purged) TPIPE= is required when TMEMBER= is specified.

**TPIPE**= A 1- to 8-character TPIPE name. Optional unless TMEMBER is specified for late replies.

**SMEM**= Specifies whether (YES) or not (NO) the TMEMBER is a supermember. Optional parameter.

**EXIT**= Specifies whether (YES) or not (NO) the IMS user exit (DFSCMUX0) can override the descriptor routing information for late messages when TYPE=IMSTRAN. Optional parameter.

**REPLYCHK**= Specifies whether (YES) or not (NO) IMS application replies to the IOPCB. If YES and the target application does not reply to the IOPCB nor message switches to another transaction, IMS returns a bad return code X'0100', reason code X'0110', and extended reason code X'0061' instead of a timeout to the ICAL call.

**SYNCTP** = Specifies whether (YES) or (NO) a synchronous TPIPE is to be created with recoverable sequence numbers for input and output messages. Optional parameter. Primarily used with WebSphere MQ and can apply to late response messages

**SYNTIMER**= Specifies the ICAL timeout value for synchronous program switch. Optional parameter. If timeout value is also specified in the AIB interface, IMS compares the timeout values and selects the lower value.

- The other existing parameters OTMA destination descriptors are not applicable for TYPE=IMSTRAN

The supported parameters for TYPE=IMSTRAN are as follows:

- LTERMOVR= Specifies the LTERM name used to override the LTERM name in the IMS application program's I/O PCB.
- TMEMBER= A 1- to 16-character OTMA TMEMBER name. This parameter is optional when TYPE=IMSTRAN. When specified, IMS will queue the late response of a synchronous program switch to this OTMA TMEMBER. And TPIPE= is required when TMEMBER= is specified.
- TPIPE= A 1- to 8-character TPIPE name. This parameter is optional. IMS uses this TPIPE to queue the late response for synchronous programs switch when TYPE=IMSTRAN. And TMEMBER= is required when TPIPE= is specified.
- SMEM= Specifies whether (YES) or not (NO) this destination is a supermember.
- EXIT= Specifies whether (YES) or not (NO) the IMS user exit (DFSCMUX0) can override the descriptor routing information for late messages when TYPE=IMSTRAN. It is an optional parameter and defaults to NO.
- REPLYCHK= Specifies whether (YES) or not (NO) IMS application replies to the IOPCB. When REPLYCHK=YES and the ICAL switch-to application does not reply to the IOPCB nor message switch to another transaction, IMS will return a bad return code X'0100', reason code X'0110', and extended reason code X'0061' instead of a timeout to the ICAL call. It defaults to YES.
- SYNCTP = Specifies whether (YES) or (NO) a synchronous TPIPE is to be created with recoverable sequence numbers for input and output messages. Optional parameter. Primarily used with WebSphere MQ and can apply to late response messages
- SYNTIMER= Specifies the ICAL timeout value for synchronous program switch. If timeout value is also specified in the AIB interface, IMS will compare the timeout values and select the lower one for this ICAL call. This is an optional parameter.

## ***OTMA destination routing descriptor for DL/I ICAL ...***

### ▪ Consideration

- The same TYPE=IMSTRAN descriptor
  - Can be used on behalf of multiple target transactions
    - Actual trancode is in the request\_area of the ICAL and not in the descriptor
    - Need for different descriptors is based on different processing requirements
      - E.g., different REPLYCHK processing requirements for one set of transactions versus another
  - Minimizes the number of descriptors needed for synchronous program switch support

The destination descriptor with TYPE=IMSTRAN for synchronous program switch requests can be used on behalf of multiple target transactions. The actual trancode is specified in the message sent by the ICAL and not in the destination descriptor. Creating multiple of the TYPE=IMSTRAN descriptors is useful if different processing characteristics are needed.

### ***OTMA destination routing descriptor for DL/I ICAL ...***

- Type-2 commands can also be used to update, create, delete, or query the descriptor entries for synchronous program switch

```
For example: UPDATE OTMADESC NAME(OTMDEST1) SET(SYNTIMER(800))
             QRY OTMADESC NAME(OTMDEST1) SHOW(ALL)
```

- CREATE OTMADESC and UPDATE OTMADESC
  - New optional sub-parameters for SET
    - SYNCTP, REPLYCHK, EXIT, and LTRMOVR
- QUERY OTMADESC
  - Includes information for the optional new parameters
    - SYNCTP, REPLYCHK, EXIT, and LTERMOVR

In lieu of creating descriptors, the Type-2 commands can be used to update, create, or delete OTMA descriptor entries. Additionally, a QUERY command can be used to display information associated with a descriptor. For synchronous program switch support, the new optional sub-parameters (SYNCTP, REPLYCHK, EXIT, and LTERMOVR) can be SET as well as queried.

## **OTMA Support for Synchronous Program Switches**

- **Non-XCF related OTMA services are used**
  - Internal invocation of OTMA services to process the target transaction request
    - Without specifying OTMA=Y in the DFSPBxxx member of IMS PROCLIB and without issuing /START OTMA command
      - No need to start the XCF connection with any OTMA client for the synchronous program switch
  - The target transaction is processed as an OTMA transaction
    - If authorization is required, IMS checks to see if user can access target
    - **Using OTMA send-then-commit (CM1) protocol with SyncLevel=CONFIRM**
      - Target transaction of the ICAL is processed as an OTMA transaction
    - **IMS creates an internal OTMA member DFSYICAL and internal tpipe DFSTPIPE to process the transaction**
      - And generates internal ACK/NAK for the CONFIRM request

IMS internally schedules transactions initiated by the ICAL as OTMA transactions. Using OTMA, however, for the synchronous program switch support will be transparent and automatic. This means that there will be no need to specify OTMA=Y in the IMS PROCLIB member DFSPBxxx nor will there be a need to issue a /START OTMA command.

Once a DL/I ICAL for synchronous program switch is accepted, IMS first checks to see if authorization is required and, if so, check to ensure the user is allowed to access the target transaction. Once the authorization process is complete, IMS internally use the OTMA send-then-commit (CM1) protocol with SyncLevel=CONFIRM to process this request. An internal ACK for the CM1 response will always be generated. The response is not sent until the IMS syncpoint is complete.

The OTMA TMEMBER DFSYICAL with the TPIPE name DFSTPIPE will be created.

### OTMA Support for Synchronous Program Switches ...

```

SDSF SYSLOG      2.103 STL1 STL1 01/25/2012 1W          2,561  COLUMNS 52- 131
COMMAND INPUT ==>
0010 IEE6001 REPLY TO 65 IS;/DIS OTMA.                SCROLL ==> PAGE
0010 DFS0001 GROUP/MEMBER          XCF-STATUS  USER-STATUS SECURITY  TIB
0010 DFS0001 INPT SMEM             IMS1
0010 DFS0001 DRUEXIT  T/O          ACEEAGE
0010 DFS0001 HARRY                 IMS1
0010 DFS0001 -IMS1                 ACTIVE     SERVER     NONE     0
0010 DFS0001 8000                 IMS1      N/A       0
0010 DFS0001 -IMS1                 N/A       0
0010 DFS0001 -VCS                 IMS1     ACTIVE     ACCEPT TRAFFIC NONE     0
0010 DFS0001 5000                 IMS1     120      999909
0010 DFS0001 -DFSICAL              NOT DEFINED SYNC P2P   NONE     0
0010 DFS0001 5000                 IMS1     N/A       0
0010 DFS0001 -DFSICAL              IMS1     N/A       0
0010 DFS0001 *12025/145131*       IMS1
  
```

04/021

Connected to remote server/host stbmi1.svl.ibm.com using port 23

ussvlng-A487-04-A-Silicon Valley Lab on ussvlng

A /DIS OTMA command can be used to see the existence of the DFSYICAL TMEMBER and associated properties.

## IMS Commands

- /DIS ACTIVE REGION
  - Displays the target transaction for the synchronous program switch
  - Displays the calculated end time of the ICAL
    - Based on the timeout value

```

SDSF SYSLOG      2.103 STL1 STL1 01/25/2012  11.4.10.00  COLUMNS 52- 131
COMMAND INPUT ==>
0010 IEE000I REPLY TO 69 IS:/DIS A REGION          SCROLL ==> PAGE
0010 DFS000I REGID JOBNAME TYPE TRAN/STEP PROGRAM STATUS
CLASS IMS1
0010 DFS000I      2 MPP1B TP IAPMD127 IAPMD127 WAIT-CALLOUT
      8. 8. 8. 8 IMS1
0010 DFS000I      5. 5. 5. 5 IMS1
0010 DFS000I      TRAN: SKS1 END TIME: 12025/191057
IMS1
0010 DFS000I JMPRON JMP NONE IMS1
0010 DFS000I JBPRON JBP NONE IMS1
0010 DFS000I BATCHREG BMP NONE IMS1
0010 DFS000I FPRON FP NONE IMS1
0010 DFS000I DBTRGN DBT NONE IMS1
0010 DFS000I DBRXCSAM DBRC IMS1
0010 DFS000I DLIXCSAM DLS IMS1
0010 DFS000I *12025/174531* IMS1
0010 *70 DFS096I *IMS READY* IMS1
S READY* IMS1
***** BOTTOM OF DATA *****
04/021
  
```

Note: Display output will be different for synchronous callout ICAL versus synchronous program-to-program ICAL

07- IMS Application: 20

In this example, message processing program IAPMD127 is processing transaction IAPMD127 which is waiting for a response to a synchronous callout request (WAIT-CALLOUT). This request is making a synchronous program switch using DL/I ICAL to a target transaction SKS1. The timeout value of this ICAL was used to compute the end time of this ICAL request which is displayed in the END TIME field.



## IMS Commands ...

- **/PSTOP REGION *rgn#* SYNC *tran name* ...**
  - Allows a program in a wait state to terminate
- **/STOP TMEMBER DFSYICAL [ TPIPE DFSTPIPE ]**
  - Stops all of the synchronous program switches in this IMS
- **/START TMEMBER DFSYICAL [ INPUT *flood\_limit* ]**
  - Starts synchronous program switches after a /STOP command
    - Optional use of the INPUT keyword provides a flood control value
- **/DISPLAY OTMA**
  - Displays the big picture of OTMA members, even when OTMA is stopped
  - New user status:
    - SYNC P2P
    - SYNC P2P+FLOOD
- **/DISPLAY TMEMBER DFSYICAL TPIPE DFSTPIPE SYNC**
  - Displays information about the tpipes created for the synchronous program switches
  - Displays the total number of executed synchronous program switches,

•The /PSTOP command wakes up an application program that is waiting for the response from a DL/I ICAL wait so the program can terminate. If a tran name is specified following the SYNC parameter, it will also apply to an ICAL that is performing synchronous program switch. Additionally, the ICAL call that is in the wait state will be posted and received AIB return code X'0100' with reason code X'010C'.

•The /STOP TMEMBER DFSYICAL can be used to disable the synchronous program switch support. Application programs issuing ICAL requests after this command will receive: AIB return code X'00000100', reason code X'00000110' and extended reason code X'00000005'. If the command also includes TPIPE DFSTPIPE the subsequent synchronous program switches from IMS application issuing ICAL will be rejected with AIB return code X'00000100', reason code X'00000110' and extended reason code X'00000006'.

•The /START TMEMBER DFSYICAL command allows the support to once again be activated. If an INPUT *flood\_limit* is specified then the *flood\_limit* value initiates flood control support and limits the number of synchronous program switch requests that can be active.

•The /DISPLAY OTMA command can be used to determine if this IMS has been used to process any synchronous program switch requests. The information displayed includes whether or not there are any active requests waiting for a response and how many of them there are. The OTMA member DFSYICAL is created only for processing the ICAL calls for synchronous program switches. The TIB column can be used to identify how many active synchronous program switches exist in this IMS. Since the DRU exit has meaning to the DFSYICAL member, the DRUEXIT column for its member will have N/A. A new user status "SYNC P2P" has been introduced for the OTMA internal member DFSYICAL which initiated the synchronous program switch using the DL/I ICAL calls. When the flood limit has been set via the /START TMEMBER DFSYICAL INPUT *flood\_limit* command and the flood value has been reached, the user status will show "SYNC P2P+FLOOD". The TIB column also displays a value. For the OTMA internal member DFSYICAL, this indicates the current number of IMS regions waiting for the response of synchronous program switch. When this number reaches the optional flood limit (if set) then no more ICAL for synchronous program switch will be accepted.

## Messages and Status Codes

### Messages

- DFS4687E ERROR PROCESSING SYNC PROGRAM SWITCH
  - Provides information about a processing failure
- DFS1190I REGION *nnnn* NOT WAITING ON *yyyyyyyy xxxxxxxx*
  - Indicates that a /PSTOP command might have been issued in error

### Application program status Codes

- A1
  - CHNG call was issued using a descriptor with TYPE=IMSTRAN
  - OTMA ALTPCB output destination specified reserved name DFSYICAL
- AX
  - An OTMA user exit (DFSYPRX0, DFSYDRU0, or client DRU exit) returned invalid routing information

Additional messages that can be issued in a synchronous program switch scenario include:

- DFS4687E is a new message that indicates that an error has occurred in the synchronous program switch processing. It provides a short summary that describes the processing failure
- DFS1190I is issued to indicate that a /PSTOP AOITOKEN or /PSTOP REGION SYNC command was entered but the region was not waiting for the specified AOI token or ICAL response.

IMS application programs might additionally receive the following status codes:

A1:

- The OTMA destination descriptor entry used for a CHNG call specified a TYPE of IMSTRAN. A TYPE=IMSTRAN is only applicable to the ICAL.
- An OTMA ALTPCB output destination was specified with an OTMA member name of DFSYICAL which is a reserved name.

AX:

- An OTMA user exit (DFSYPRX0, DFSYDRU0, or client DRU exit) returned invalid routing information. The OTMA return codes in the 67D0 log record can provide more information.

## Security Considerations

- Security for ICAL is similar to CHNG/ISRT DLI calls
  - Based on transaction security specifications (TRN)
    - RACF and/or DFSCTRN0 are used for user authorization checking to determine if the ICAL can invoke the target transaction
      - Unless OTMA security is set to NONE
  
- If the ICAL is authorized to do the synchronous program switch
  - Target transaction is scheduled as an OTMA transaction with the default OTMA security level of FULL
    - Security level can be changed by issuing the command:
      - /SECURE OTMA NONE, or
      - /SECURE OTMA TMEMBER DFSYICAL CHECK|NONE

Security for ICAL is similar to CHNG/ISRT DLI calls. Depending on the transaction security specifications (TRN) the IMS region will call RACF and/or DFSCTRN0 user exit to check if the user is authorized to issue the ICAL for the target transaction. For APPC/OTMA transaction it also depends on the security option specified for APPC/OTMA. If the security of NONE has been specified for APPC/OTMA transactions, RACF and/or DFSCTRN0 will not be called even if TRN=Y has been specified.

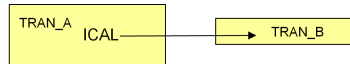
As mentioned earlier, IMS schedules the transaction initiated by the ICAL as an OTMA transaction which means that OTMA security settings (NONE/ CHECK/ FULL/ PROFILE) are all supported. With or without the activation of OTMA, the OTMA security is set to FULL as the default. This security for ICAL processing can be changed by issuing /SECURE OTMA TMEMBER DFSYICAL CHECK|NONE command. Note that the DFSBSEX0 user exit routine will not be called.

When the /SECURE OTMA command is issued, it sets the security setting for the OTMA internal member DFSYICAL. This member has been created internally by OTMA to process the DL/I ICAL calls for synchronous program switches.

## Security Considerations ...

- If the ICAL is authorized to do the synchronous program switch ...
  - Even when DFSYICAL does not yet exist in the system or when the target is in another IMS system
    - You can issue /SECURE OTMA... to preset the security level

The transaction security specification of TRAN\_A will be used to perform the security authorization for the ICAL.



The default OTMA security level of FULL is used to process TRAN\_B in the region.

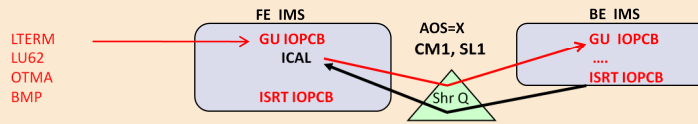
Optionally, issue /SECURE OTMA TMEMBER DFSYICAL CHECKNONE ahead of time

Even when the DFSYICAL member does not exist in the IMS system, this command can still be issued to create this member and to set the security level for the subsequent DL/I ICAL calls.

## Shared Queues Support

- IMS leverages the APPC/OTMA XCF shared queues (SQ) function
  - For both request and response messages
  - Specifying AOS= and/or RRS= in the IMS PROCLIB members is not required

- ICAL automatically and always uses AOS=X (SL1 uses XCF)
  - Compatible with any existing AOS= and RRS= currently set for other transactions



Requires DBRC MINVERS of 13.1 for all members of the SQ group  
 Otherwise ICAL will get a rejection

RC	RS	Extended Reason	Explanation
X'0100	X'0100'	X'0115'	Request message is rejected. The synchronous program switch was executed in the Shared Queues environment, but the IMS systems in the Shared Queues do not have the same MINVERS value of 13.1.

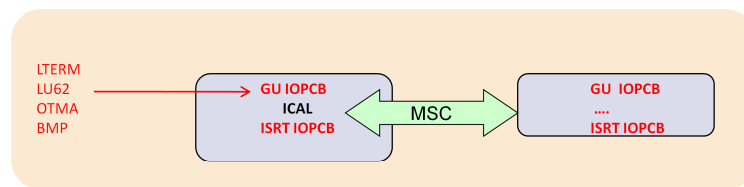
For IMS shared queues (SQ) customers, all of the IMS systems in the same SQ group must have the same MINVERS value of 13.1 in order to use this function.

Note that IMS leverages the APPC/OTMA XCF shared queues function (introduced in IMS 12) to process the switched-to transaction at a shared queues front-end and back-end IMS. This processing for synchronous callout message does not require the specification of AOS= or RRS= in the IMS PROCLIB members, and is compatible with any existing AOS= and RRS= settings used by the customers. What this means is that if an IMS system already uses AOS= and RRS= specifications to process its shared queues transactions, the non-ICAL transactions will continue to use these settings but synchronous program switch message using ICAL calls will be executed in the shared queues environment independently using AOS=X.

As a reminder, AOS=X allows synchronous transactions with synclevel of NONE|CONFIRM to be processed in a Shared Queues back-end system using XCF communications. The processing of synclevel SYNCPT requests is equivalent to AOS=N.

## MSC Support

- IMS supports sending an ICAL request to an MSC back-end system
  - MSC/VTAM
  - MSC/TCPIP with IMS Connect (introduced in IMS 12)



All of the IMS subsystems in an MSC network can process synchronous programs switch messages, as long as the MSC link exists in the front-end IMS.

## New OTMA Message Header

- For synchronous program switch support

- OTMA Message Header

- Control data header – new message type field

TMAMCMGT DS	X	Message type
TMAMCDTA EQU	X'80'	Msg type=Data
TMAMCTXN EQU	X'40'	Msg type=Transaction
TMAMCRSP EQU	X'20'	Msg type=Response
TMAMCCMD EQU	X'10'	Msg type=Protocol Command
TMAMCCMT EQU	X'08'	Msg type=Commit Confirmation
TMAMSP2P EQU	X'04'	Msg type=Synchronous program switch

- State data header

- Specification of CM1, Synclevel=Confirm
- Recovery token from LCRETOKN

- Security data header

- Includes the userid from PSTUSID

- User data header

A new type of OTMA message header is introduced for the synchronous program switch. This type of message consists of OTMA control data header, OTMA state data header, OTMA security header, OTMA user data header, and the request data of the ICAL.

In the control data header, a new flag x'04' is added to the message type field to identify the message for the synchronous program switch:

TMAMSP2P EQU X'04' Msg type=Synchronous program switch

In the state data header, the commit-mode=1, Synclevel=CONFIRM, and recovery token from LCRETOKN are specified.

In the security data header, the userid from PSTUSID is included.

In the user data header, OTMA prepares a special format of user data for the processing of the synchronous program switch.

## *Exit Routine Enhancements*

- *DFSCMUX0*
- *DFSYPRX0*
- *DFSYDRU0*
- *DFSYIOE0*



### ***DFSCMUX0 (Message Control/Error Exit Routine)***

- Invoked when EXIT=YES is specified in the IMSTRAN OTMA descriptor
  - Can take action for **late response** messages as follows:
    - Dequeue
    - Or, Reroute
      - To an LTERM or OTMA destination
      - With the descriptor information (TMEMBER/TPIPE) associated with the synchronous program switch
        - Passed to the exit in the MSNB COPY
        - A new exit flag, MSNBDESC, can be set to reroute the late response message to the defined TMEMBER and TPIPE.
      - If a non-supported action is specified for the late response message
        - IMS ignore the request and discards the late response message

When EXIT=YES is specified in the OTMA descriptor for the synchronous program switch and a late response message is created, DFSCMUX0 user exit will be called to take actions. The supported actions are either to dequeue the late response message or to reroute it. If reroute action is requested and the destination information is provided, the late response message can be rerouted to an LTERM or OTMA destination. If the reroute TMEMBER and TPIPE names are also specified in the OTMA descriptor (associated with the original synchronous program switch), they will be passed in the MSNB COPY for the DFSCMUX0 user exit. The new exit flag, MSNBDESC, can be set to reroute the late response message using the reroute TMEMBER and TPIPE information specified in the descriptor. If a non-supported action is specified for the late response message, IMS will ignore it and discard the late response messages.

### ***DFSYPRX0 (OTMA Pre-Routing Exit Routine)***

- Provides a new flag in the flag byte offset +24 of the input parameter list
  - Indicates that the OTMA ALTPCB output message was originally triggered by a synchronous program switch DL/I ICAL call
    - New description:

+24 = 1-BYTE FLAG

X'10' - IF SET, INDICATES THAT IT IS TRIGGERED BY A SYNCHRONOUS PROGRAM SWITCH ICAL CALL.

IF X'80' IS ALSO SET, IT MEANS AN OTMA TRANSACTION INITIATED THE ICAL CALL, AND THE LTERM/TPIPE NAME AND INPUT CLIENT MEMBER NAME IN THE PARAMETER LIST ARE FROM THE ORIGINAL OTMA TRANSACTION.

and ...

A new flag was added to the flag byte at offset +24 of the input parameter list of DFSYPRX0 (OTMAS Pre-Routing Exit Routine). This new flag indicates that this OTMA ALTPCB output message was originally triggered by a synchronous program switch DL/I ICAL call.

## DFSYPRX0 (OTMA Pre-Routing Exit Routine) ...

- Enhancements to the MCI and STATE DATA in the input parameter list
  - Indicate that the OTMA prefix may be generated by the system
    - Not from the original OTMA transaction which later invokes ICAL

+64 = ADDR(MESSAGE CONTROL INFORMATION),

AVAILABLE FROM INPUT OTMA MESSAGE PREFIX. THIS IS ENTRY PARAMETER ONLY.

IF IT IS FROM A SYNCHRONOUS PROGRAM SWITCH ICAL CALL AND THE ORIGINAL TRANSACTION IS FROM OTMA, THIS MESSAGE CONTROL INFORMATION PREFIX IS A SYSTEM CREATED MESSAGE PREFIX. IT IS NOT THE ORIGINAL MESSAGE PREFIX FROM OTMA CLIENT.

HOWEVER, THE LTERM/TPIPE NAME AND INPUT CLIENT MEMBER NAME IN THIS PARAMETER LIST ARE FROM THE ORIGINAL OTMA MESSAGE.

+68 = ADDR(STATE DATA),

AVAILABLE FROM INPUT OTMA MESSAGE PREFIX. IF SUPER MEMBER FEATURE IS USED, SUPER MEMBER NAME IS STORED AT +14 FROM THE BEGINNING OF STATE SEE TMAMSPNM FIELD THIS IS AN ENTRY PARAMETER ONLY.

@PK09946  
 OFFSET @PK09946  
 DATA @PK09946  
 IN DFSYMSG MACRO @PK09946  
 AN ENTRY PARAMETER ONLY.

CALL IF IT IS FROM A SYNCHRONOUS PROGRAM SWITCH ICAL AND THE ORIGINAL TRANSACTION IS FROM OTMA, THIS STATE DATA IS A SYSTEM GENERATED

STATE DATA PREFIX. IT IS NOT THE ORIGINAL OTMA CLIENT STATE DATA PREFIX FROM THE HOWEVER, THE CORRELATOR, TMAMHCOR, IN THE STATE DATA PREFIX IS OBTAINED FROM THE ORIGINAL OTMA STATE DATA. AND THE LTERM/TPIPE NAME AND THE INPUT CLIENT MEMBER NAME IN THIS PARAMETER LIST ARE FROM THE ORIGINAL OTMA MESSAGE.

Additionally, enhancements to the MCI and STATE DATA parameter of the input parameter list of the exit indicate that the OTMA prefix may have been generated by the system rather than reflect the original OTMA transaction which invoked the ICAL.

## ***DFSYDRU0 (OTMA Destination Resolution Exit Routine)***

- Provides a new flag in the flag byte offset +24 of the input parameter list
  - Indicates that the OTMA ALTPCB output message was originally triggered by a synchronous program switch DL/I ICAL call
    - New description:

+24 = 1-BYTE FLAG

X'10' - IF SET, INDICATES THAT IT IS TRIGGERED BY A SYNCHRONOUS PROGRAM SWITCH ICAL CALL.

IF X'80' IS ALSO SET, IT MEANS AN OTMA TRANSACTION INITIATED THE ICAL CALL, AND THE LTERM/TPIPE NAME AND INPUT CLIENT MEMBER NAME IN THE PARAMETER LIST ARE FROM THE ORIGINAL OTMA TRANSACTION.

and ...

DFSYDRU0 (OTMA Destination Resolution Exit Routine) has also been enhanced to provide a new flag at offset +24 of the input parameter list. This new flag indicates that the OTMA ALTPCB output message was originally triggered by a synchronous program switch DL/I ICAL call

## DFSYDRU0 (OTMA Destination Resolution Exit Routine)..

- Enhancements to the MCI and STATE DATA in the input parameter list
  - Indicate that the OTMA prefix may be generated by the system
    - Not from the original OTMA transaction which later invokes ICAL

### +80 = ADDR(MESSAGE CONTROL INFORMATION),

AVAILABLE FROM INPUT OTMA MESSAGE PREFIX. THIS IS AN ENTRY PARAMETER ONLY.

IF IT IS FROM A SYNCHRONOUS PROGRAM SWITCH ICAL CALL AND THE ORIGINAL TRANSACTION IS FROM OTMA, THIS MESSAGE CONTROL INFORMATION PREFIX IS A SYSTEM CREATED MESSAGE PREFIX. IT IS NOT THE ORIGINAL MESSAGE PREFIX FROM OTMA CLIENT.

HOWEVER, THE LTERM/TPIPE NAME AND INPUT CLIENT MEMBER NAME IN THIS PARAMETER LIST ARE FROM THE ORIGINAL OTMA MESSAGE.

### +84 = ADDR(STATE DATA),

AVAILABLE FROM INPUT OTMA MESSAGE PREFIX. IF SUPER MEMBER FEATURE IS USED, SUPER MEMBER NAME IS STORED AT +14 FROM THE BEGINNING OF STATE SEE TMAMSPNM FIELD THIS IS AN ENTRY PARAMETER ONLY.

IF IT IS FROM A SYNCHRONOUS PROGRAM SWITCH ICAL AND THE ORIGINAL TRANSACTION IS FROM OTMA, THIS STATE DATA IS A SYSTEM GENERATED STATE DATA PREFIX. IT IS NOT THE ORIGINAL OTMA CLIENT STATE DATA PREFIX. HOWEVER, THE CORRELATOR, TMAMHCR, IN THE STATE DATA PREFIX IS OBTAINED FROM THE ORIGINAL OTMA STATE DATA. AND THE LTERM/TPIPE NAME AND THE INPUT CLIENT MEMBER NAME IN THIS PARAMETER LIST ARE FROM THE ORIGINAL OTMA MESSAGE.

The MCI and STATE DATA parameters of the input parameter list of the exit indicate that OTMA prefix may have been generated by the system are not from the original OTMA transaction which invoked the ICAL.

### ***DFSYIOE0 (OTMA Input/Output Edit Exit Routine)***

- Not invoked for synchronous program switch request and reply messages
  
- **Can be invoked for late responses** that are routed to OTMA destinations
  - OTMA needs client specific information in the user data prefix (1024 bytes)
    - If the initial message that results in a synchronous program switch originated in OTMA
      - Client specific user data is propagated to the late response
        - **Can** be overridden by DFSYIOE0
      - If the initial message did not come from an OTMA client
        - User data prefix in the late response will be initialized to zeroes
          - DFSYIOE0 ***must*** provide the client specific information
    - A sample DFSYIOE exit routine is provided in IMS 13

DFSYIOE0 (OTMA Input/Output Edit Exit Routine) is never invoked for synchronous program switch request and reply messages but it **can be called for late responses** that are destined for an OTMA client.

For OTMA destinations, IMS relies on the information in the 1024-byte user data prefix of the late response.

•If the synchronous program switch call was initiated by an OTMA transaction from an OTMA client, then IMS will propagate the initial client user data to the user data prefix of the late response message. This user data prefix can remain as it is or can optionally be updated by the DFSYIOE0 user exit.

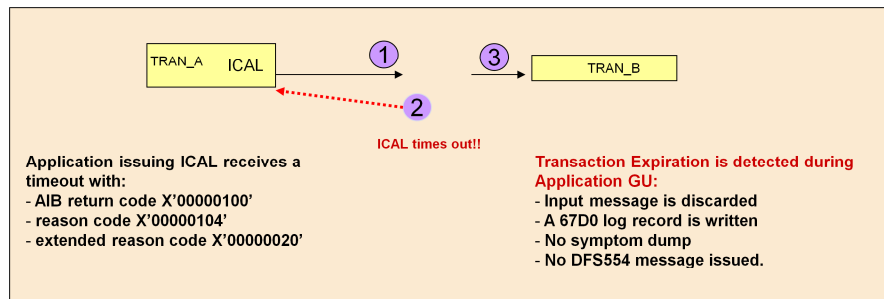
•On the other hand, if the program that initially issued the synchronous program switch call was not initiated by an OTMA client, such as IMS Connect, then this prefix will be initialized to zeroes and DFSYIOE0 must be used to build the client specific user data so that it can be correctly sent to the OTMA client.

## *Application Design Considerations*

- *Transaction Expiration*
- *Late Responses*
- *LTERM Override*
- *ALTPCB destinations*
- *Examples*
  - *Multiple ICALs*
  - *Recursive requests*
  - *DFSDDLTO*

### Transaction expiration for the target transaction

- Transaction expiration for targets of synchronous program switches (new)
  - Uses the timeout value of the synchronous program switch
    - ICAL timeout value is carried in the OTMA header
  - Invoked during application GU time



07- IMS Application: 36

- (1) When a transaction issues an ICAL, e.g., TRAN\_A, the OTMA message header that is built for the synchronous program switch message carries the ICAL timeout value which is used for the target transaction expiration process.
- (2) If the ICAL times out before a response is received, then TRAN\_A receives an AIB return code/ reason code indicating the condition.
- (3) When TRAN\_B is finally scheduled and the application issues a GU IOPCB, transaction expiration is detected. Since the OTMA message header contains the transaction expiration information with TRAN\_A's ICAL timeout value, the transaction expiration process discards the input message and a 67D0 log record is written. No DFS message is sent back to the ICAL call since it is no longer waiting, and no symptom dump is produced.



***Transaction expiration for the target transaction ...***

- **Transactions expiration will NOT occur**
  - For regular program-to-program switch destinations performed by the target
    - Only the initial target transaction can be expired
  - In a remote IMS through MSC
    - After ICAL times out, the response message from the remote MSC system will be processed as a late message
  - For fastpath transactions
  - If TMEMBER, TPIPE and/or EXIT=YES are specified
    - These specifications designate what to do with Late Reply messages
  
- **NOTE: If the ICAL request times out and transaction expiration is not performed, late response messages are either:**
  - Dequeued
  - Rerouted

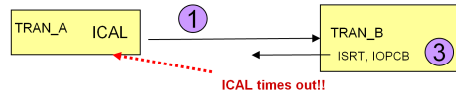
Note that the transaction expiration process does not occur:

- When the target of the ICAL also issues program-to-program switches. Only the initial called program can be expired.
- For ICAL messages sent across an MSC link. If the calling program issuing the ICAL times out, any response message from the remote MSC system is processed as a late response message.
- For Fastpath transactions.
- If the descriptor defines parameters that can deal with a late response.

## Late Response Messages?

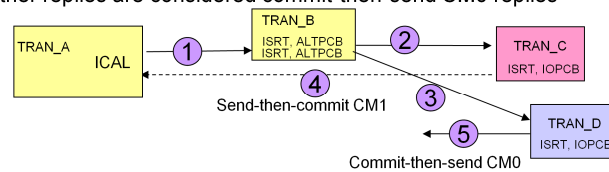
- A reply is late when:

- It is returned to the synchronous program switch ICAL call which has already timed out



- It is a subsequent message(s) returned to the ICAL which has already received and processed a previous response

- If multiple target transactions are involved that reply to the IOPCB
  - The **first** one to process an application GU,IOPCB has the responsibility for the send-then-commit CM1 response
  - The other replies are considered commit-then-send CM0 replies



07- IMS Application: 38

An IOPCB reply is a late response message when IMS attempts to return to a calling synchronous program switch program (issuer of the ICAL) after that program has either: timed out, or already received a previous response message.

The bottom picture on the visual gives an example of the latter. Note that in this scenario, the target of the synchronous program switch TRAN\_B does not reply to the IOPCB but rather issues two program-to-program switches. TRAN\_C and TRAN\_D which both respond to the IOPCB can be ultimately responsible for responding to TRAN\_A's ICAL request. The first one to process an application GU,IOPCB gets the responsibility of sending the CM1 response. The other transaction's reply becomes a CM0 reply and is considered a late response.

### ***Late Response Messages...***

- The default action for processing the late response message is to ***dequeue***
- Optionally, it can be ***rerouted*** to an LTERM or OTMA destination
  - OTMA TMEMBER and TPIPE destinations can be specified in the descriptor
    - Destination can be IMS Connect and WebSphere MQ
      - Optional: supermember (SMEM) and synchronized TPIPE (SYNCTP)
  - If EXIT=YES is specified in the descriptor
    - The DFSCMUX0 user exit can identify an LTERM or OTMA destination
  - When TMEMBER, TPIPE, and EXIT=YES are all specified
    - TMEMBER and TPIPE are passed to DFSCMUX0 for final routing decision
      - A new output flag can be set by the user exit to inform IMS to use the TMEMBER and TPIPE in the descriptor for routing the late messages
  - NOTE: DFSYICAL and DFSTPIPE are invalid destinations for reroute

By default, IMS logs and dequeues late response messages if detected. However, customers can save the late response messages by informing IMS to route the late response messages to a LTERM, LU62, or OTMA queue.

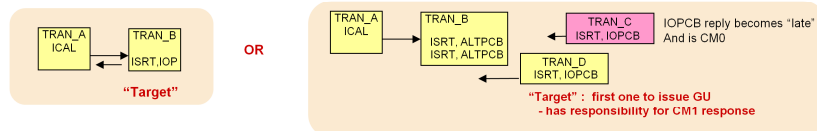
The TMEMBER= and TPIPE= parameters in the descriptor of the synchronous program switch can specify an OTMA TPIPE queue so that the late response can be retrieved by an OTMA client, such as IMS Connect or WebSphere MQSeries. Optionally, the SMEM= parameter for super member and SYNCTP= parameter for WebSphere MQ synchronous TPIPEs can be used to further specify special TPIPE queues for the message retrieval request. When the TMEMBER= and TPIPE= parameters are specified without EXIT=YES, all of the late messages are queued to the specified TPIPE and TMEMBER

Specifying the EXIT parameter with YES allows DFSCMUX0 user exit (described earlier) to determine the final fate of the late response message even without TMEMBER and TPIPE specification.

If the TMEMBER, TPIPE, and EXIT=YES are all specified, then the DFSCMUX0 user exit is invoked with the member-name and tpipe-name as defined in the descriptor.

## Late Response Messages ...

- REPLYCHK parameter (YES |NO) in the IMSTRAN descriptor
  - Specifies whether or not IMS should check that the “target” IMS transaction responds to the IOPCB
  - **“Target”** transaction has responsibility for the send-then-commit CM1 reply



- Controls the actions IMS performs
  - Including what to do with the **DFS2082** message
    - Existing OTMA mechanism that releases a waiting request when the target transaction for a send-then-commit CM1 response does not reply to the IOPCB
    - Note:
      - ICAL will never get a DFS2082 for this function
      - DFS2082 messages are not considered late messages and are not rerouted

The REPLYCHK parameter can optionally be used when multiple response messages are competing to be sent back to the ICAL call and specifies whether or not IMS should check that a “target” transaction responds to the IOPCB. To better understand this setting (detail in the following visuals) some concepts need to be clarified.

A “target” transaction is the transaction that has the responsibility of replying to the outstanding ICAL request. It could be the only transaction in the ICAL path (e.g., TRAN\_B in picture on the left) or it could be the first one that is switched to that issues a GU IOPCB (e.g., TRAN\_D in the picture on the right). The assumption is that the “target” will issue an ISRT IOPCB to send a reply.

Additionally, the REPLYCHK specification influences the process that IMS uses for the DFS2082 message. Historically, the DFS2082 message has been used by OTMA to inform the OTMA client that the target transaction does not insert back to IOPCB and does not issue the program to program switches. This is needed so that the client does not need to: wait for an IMS response that will not come, or wait until a timeout occurs.

For synchronous program switches, the caller of the ICAL call will never receive a DFS2082 message but it may receive AIB return/reason codes instead.

There is no reroute or late message processing for the DFS2082 message for ICAL calls.

**REPLYCHK = YES**

- **Before an ICAL timeout**
  - The “target” transaction is expected to create the IOPCB CM1 response
    - If there is no reply
      - ICAL receives AIB return code X'00000100', reason code X'00000110', and extended reason code X'00000061' instead of a DFS2082 message
  - For multiple responses
    - CM1 response takes precedence over any other reply messages
      - All other IOPCB replies are CM0 late responses (dequeued or rerouted)
- **After an ICAL timeout**
  - All replies (CM1 and CM0) are considered late (dequeued or rerouted)
  - Transaction expiration considerations:
    - If the “target” transaction expires at the application GU time, it is discarded
      - Any other transaction IOPCB replies are CM0 late responses
    - If the “target” transaction does not expire at the application GU time
      - Subsequent program switches are performed without transaction expiration

**REPLYCHK=YES**

Before an ICAL timeout:

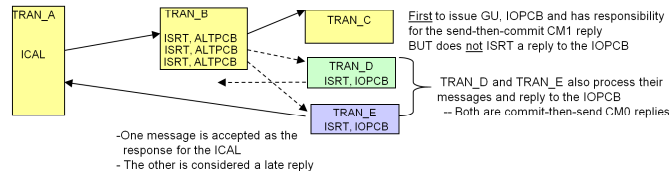
- The target transaction (previous visual) is responsible for sending a reply. If there is no IOPCB reply then DFS2082 processing occurs but instead of sending the message the ICAL receives AIB error codes to clear the outstanding wait.
- On the other hand, if there are multiple responses that could be returned to the ICAL, the CM1 message takes precedence over any CM0 replies.

After an ICAL timeout:

- All replies are considered late and are either dequeued or rerouted.
- Additionally, there could be some transaction expiration considerations. If the initial target transaction of the synchronous program switch expires at application GU time, this target transaction is simply discarded. If the initial target transaction issues the application GU before ICAL timeout and then issues subsequent program-to-program switches, these will be performed without transaction expiration even if the ICAL timer then expires. Multiple late response messages can be generated for the ICAL. These late response messages by default will be discarded if there is no reroute option specified.

**REPLYCHK=NO**

- Before an ICAL timeout
  - First reply message, either send-then-commit CM1 or commit-then-send CM0, is accepted as the response message for the synchronous ICAL
    - Other response messages are late replies (dequeued or rerouted)



- After an ICAL timeout
  - Same considerations as *REPLYCHK=YES* on previous visual
- CM1 DFS2082 messages are ignored

**REPLYCHK=NO**

In the case of REAPLYCHK=NO in the descriptor, IMS expects that there could be multiple responses for the ICAL call. Since the DFS2082 message is not a real response, it will be ignored and not be sent back to ICAL.

Before an ICAL timeout:

If there are multiple response messages for the ICAL, the first response message received, either a send-then-commit output or commit-then-send output, will be accepted as the response message. The rest of the response messages are all late response messages. In the example on the visual, TRAN\_A issues an ICAL to TRAN\_B. TRAN\_B does not reply to the IOPCB but instead issues 3 program-to-program switches. TRAN\_C is the first to issue a GU IOPCB and would ordinarily have the responsibility to send the CM1 IOPCB reply to release the ICAL wait. In this example TRAN\_C terminates without a reply. Meanwhile the other two transactions, TRAN\_D and TRAN\_E have processed and replied to the IOPCB. Both the messages are CM0 replies. With the specification of REPLYCHK=NO, IMS chooses one of the messages to be the ICAL response. The other is considered a late response.

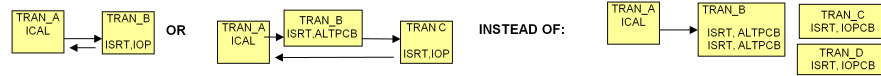
After an ICAL timeout:

Processing is the same as for REPLYCHK=YES (described on the previous visual) with the transaction expiration considerations.

DFS2082 messages are ignored.

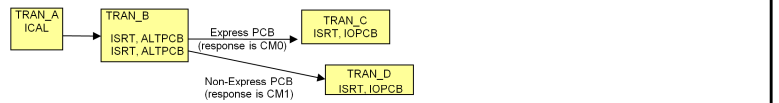
## REPLYCHK Considerations

- Use the simplest design where possible
  - Avoid spawning transactions



- Use of Express PCBs and non-Express PCBs

- REPLYCHK=NO would allow Express PCB response to be sent back as the reply

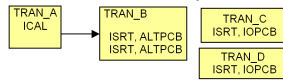


- REPLYCHK=yes would wait for non-Express PCB response to be sent back as the reply

## **REPLYCHK Considerations ...**

### ■ OTMAASY

- Pre-IMS 13 parameter that addresses synchronous (CM1) requests where multiple spawned transactions can respond to the request

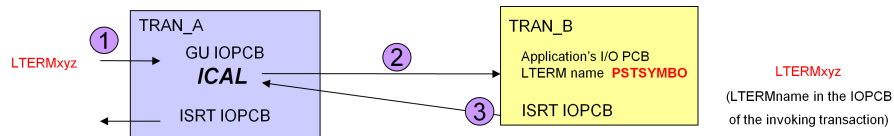


- Allows transaction definition, e.g., Response Mode versus Non-Response mode, to determine which of the spawned requests should be the CM1 response
- For Synchronous Program Switches (ICAL)
  - REPLYCHK=YES
    - OTMAASY can be useful in determining which of the spawned transactions is the target for the CM1 reply
  - REPLYCHK=NO
    - Even with OTMAASY, the first response (CM1 or CM0) is selected as the response to the ICAL



## LTERM Override

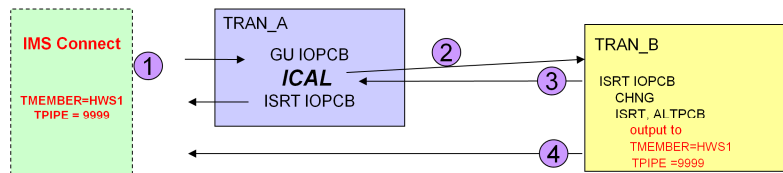
- The default LTERM name in the IMS application program's I/O PCB is the IMS application terminal symbolic (PSTSYMBO)
  - It can be overridden by the following 2 methods:
    - AIBRSNM2 in AIB block of the ICAL
    - LTERMOVR value in the OTMA descriptor for the ICAL call
  - If both AIB and descriptor has the LTERM override name, the name in the AIB will be used.



The default LTERM name in the target transaction's IO PCB is the symbolic PSTSYMBO. This value can be overridden in one of two ways: either by specifying a value in AIBRSNM2 of the AIB block used in the ICAL, or by specifying a value in the LTERMOVR parameter of the associated OTMA descriptor. If a value is specified using both methods then the name in the AIB is used.

## ALTPCB Output Messages

- Default routing destination for ALTPCB output is based on the originator of the message that invoked the ICAL transaction
  - Carried in the OTMA header
    - LTERM or OTMA information
- DFSYPRX0 and DFSYDRU0 exit routines can specify alternate routing
  - A new input flag indicates that “this ALTPCB output message is triggered by a synchronous program switch ICAL call.”
    - OTMA incoming TMEMBER/TPIPE names are passed to the exits
      - DFSYICAL and DFSTPIPE cannot be used as the valid destination name.



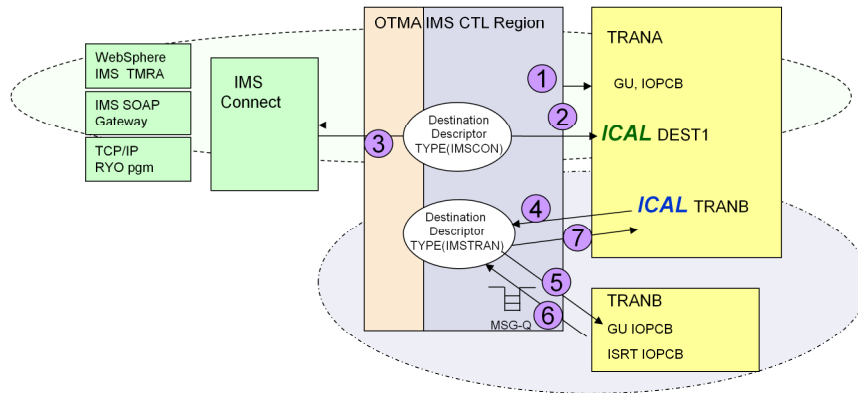
07- IMS Application: 46

When an IMS application program processing the target transaction of the synchronous program switch issues an ISRT call to an alternate PCB, or issues CHNG call, to generate an ALTPCB output message, IMS checks the originator of the ICAL message to determine the default routing destination. For example, If a terminal initiates a synchronous program switch then the legacy destination by default will be used to process the ALTPCB output. If an OTMA client, such as IMS Connect, initiates a synchronous program switch, the ALTPCB output by default will be delivered back to the incoming TPIPE.

The OTMA Pre-Routing and OTMA Destination Resolution exit routines can specify alternate routing destinations. As a reminder, these exit routines have flags that indicate that the ALTPCB message invoking the routine was originally triggered by a synchronous program switch DL/I ICAL call.

## Application Examples

- Applications can issue multiple ICALs to different destination TYPEs
  - Synchronous callout
  - Synchronous program switch

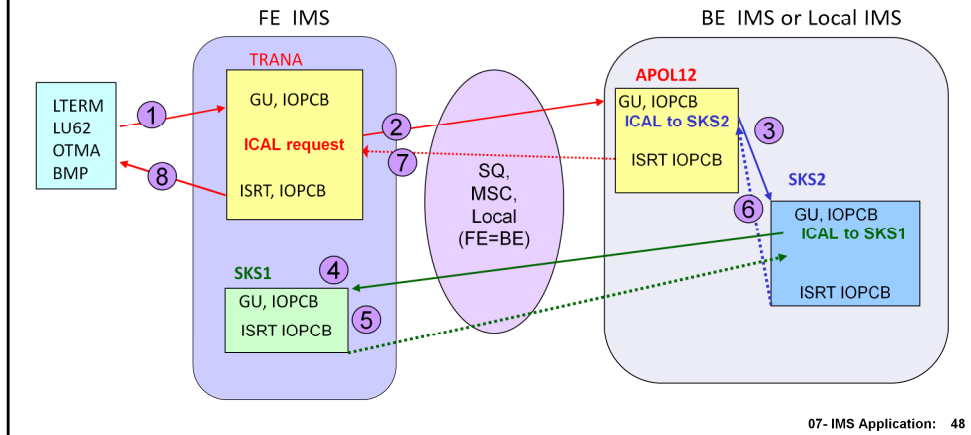


07- IMS Application: 47

An IMS application can issue ICALs to remote destinations such as web services and also to other IMS applications.

### Application Examples...

- The IMS application environment supports recursive requests
  - ICAL to ICAL
    - Across a single or multiple IMS systems



07- IMS Application: 48

The ICAL for synchronous program switching can be invoked in a recursive manner. There is no limit for the number of recursions allowed. The timeout values of the ICAL(s) should be considered when a recursion is used.

# DFSDDLT0

- Supports ICAL
  - Can be used for testing your setup

### APOL12

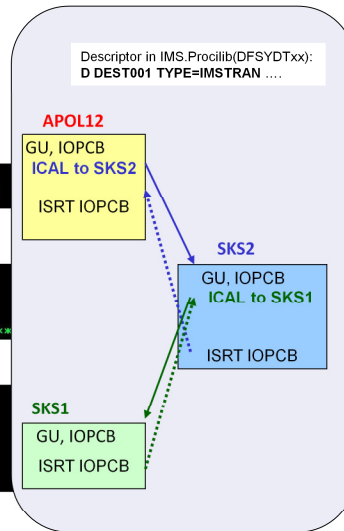
```
S1111 1 1 1 1 IOPCB AIB
L ICAL SENDRECV DEST001 999999 00100 00100
L DATA -> SKS2 gc is sent to client
```

### SKS2

```
S1111 1 1 1 1 IOPCB AIB
L ICAL SENDRECV DEST001 999999 00100 00100
L DATA -> SKS1 gc is sent to client
WTO PROGRAM DDLT02A ENDED
U*****ISSUE A RESPONSE BACK TO THE I/O PCB*****
```

### SKS1

```
WTO MFPID PROGRAM STARTED
S11 1 1 1 1 TP 1
L GU
L E OK
L ISRT
L Z0055 DATA RESPONSE FROM SKS1
L E OK
L PURG
```



DFSDDLT0 can be used to test the synchronous program switch.

### ***Implementation Considerations***

- Consider taking advantage of more dependent regions
  - 4095 MAXPST in IMS 13 (Systems Enhancements section)
  - If needed, leverage the capability of assigning classes to transactions and regions
- Determine if late replies are to be purged or rerouted
- Understand the REPLYCHK capabilities
- Determine an appropriate timeout value for SYNTIMER
- Remember database locks can be held while waiting for ICAL completion
  - If applicable use the DL/I RLSE call to release locks for unmodified data owned by the calling program before issuing ICAL

When implement the Synchronous Program Switch capability, care should be taken to understand the environment and design accordingly. For example:

- Understand the implication of using synchronous program switches to the existing IMS environment. More dependent regions might need to be defined to address possible elongation of the dependent regions that contain programs issuing ICALs. As described in the Systems Enhancements section, IMS 13 increases the MAXPT value to 4095.
- If needed, IMS provides the ability to control where work is process through its ability to define classes to transactions and dependent regions.
- The default action for late replies is to purge them. If rerouting is to occur instead then this decision has to be built into the design.
- Review the REPLYCHK capabilities and decide which set of actions is more appropriate to your environment.
- Determine an appropriate value for the SYNTIMER parameter of the TYPE=IMSTRAN descriptor so that the calling IMS program does not wait for too long a time.
- Remember that database locks can be held by the application issuing the ICAL. This is not a new consideration. If applicable, leverage the use of the DL/I RLSE call which releases locks currently held for unmodified data.

RLSE call reminder:

For fast path databases, the RLSE call releases all locks held for unmodified data that are owned by an application. For full function databases, the RLSE call releases the locks held by the DB PCB that is referenced in the call. There are considerations, however, to the use of the RLSE. After the RLSE call, all database position information is lost even though there may be no locks to release. Also, a RLSE call between ISRTs will lose the insert position so if no new position is established before inserting a dependent segment then it could possibly go under the wrong root. Note that locks protecting a resource that has been updated will not be released.

## ***Migration/Setup Considerations***

- **For Non-shared queues**
  - Synchronous program switch is a base function of IMS 13
- **For Shared Queues**
  - DBRC minimum version (MINVERS) value of 13.1 is required to enable the synchronous program switch function
    - Even if it is a single-system Shared Queues environment
- **The MSC remote IMS does not need to be an IMS 13 system**
- **New OTMA trace table entries**
  - Documented in the IMS Diagnosis Guide under “OTMA Diagnostic Aids”

## IMS 12 SPE Enhancement SSA Qualify By Position and Length

SSA qualification with the position and length of the target data instead of a DBD-defined field name.



## **SSA Enhancement - Qualify by Position**

- **IMS 12 APAR PM65139 / PTF UK81837 & UK81838**
  - New SSA command code “O”
  - Enhanced database SSA processing with ability to search for data in a segment by specifying a field position and length instead of a field name
  - Contains core IMS database code
  
- **IMS 12 APAR PM69378 / PTF UK81917**
  - Enhanced IMS Universal Drivers to allow SQL predicates containing ‘columns’ not defined in the DBD by internally converting ‘columns’ to position and length for SSA qualification
  - Contains IMS universal driver code

IMS SSA processing has been enhanced. Instead of requiring a field name in a qualification statement, it will be possible to provide the position and length of the field instead. This will allow non-DBD defined fields to be included in segment search arguments.

IMS V12 PTF UK81837 & UK81838 provide maintenance which enables a new SSA Command Code ‘O’, allowing IMS database calls to use a position and length instead of a field name for the SSA qualification.

IMS V12 PTF UK81917 provides maintenance which enables the IMS Universal Drivers to allow SQL calls to use ‘columns’ not defined in the DBD. The ‘column’ will be converted internally to position and length.

Users can now search for any field in a segment by specifying a position in the segment and a length. This feature allows non-DBD defined fields to be included in Segment Search Arguments and continues IBM’s IMS database modernization efforts. This enhances the IMS Simplification story by removing any required special affinity between IMS and JDBC tooling products that generate queries. Qualify by position will greatly enhance IMS support of JDBC tools. JDBC tools such as Data Source Explorer and Cognos generate SQL queries and do not have a concept of searchable and non-searchable fields. Currently when a non-searchable field is placed in the WHERE(SSA) clause the search will fail. This enhancement provides a method of generating searchable SSAs for non-searchable fields.

When the IMS catalog is enabled, the non-DBD fields can be defined using the EXTERNALNAME within the DBD providing support for SQL predicates using the non-searchable fields.

## SSA Command Code "O": Qualify by Position

- New SSA command code "O"
  - Non-key field definitions not required in the DBD
    - Allows fields defined in database metadata to be used
  - Search by position and length vs. field name
    - 4 byte position, hex value, relative to 1
    - 4 byte length, hex value
  - Valid for HDAM, HIDAM, PHDAM, PHIDAM and DEBD databases
  - Valid for GU, GHU, GN, GNP, GHNP, ISRT calls
  - Plays well with other command codes
  - "GE" status code returned if field not found
  - Support for DFSDDLTO and IMS REXX
  - Performance will be the same as a non-key field search

IMS 12  
APAR PM65139  
PTF UK81837 &  
PTF UK81838

SSA processing is enhanced. Instead of requiring a field name in a qualification statement, it will instead be possible to provide the position and length of the field. This will allow non-DBD defined fields to be included in segment search arguments when accessing HDAM, HIDAM, PHDAM, PHIDAM, DEBD. Command Code "O" is not supported for FP Secondary Index DBs and will return an "SD" status code if used.

To qualify an SSA, you must specify the field position. The field position is either a field, the sequence field of a virtual child or a position and a length. A qualified SSA describes the segment occurrence that you want to access. This description is called a qualification statement and has three parts. The following table shows the structure of a qualified SSA.

<u>SSA component</u>	<u>Field length</u>
Segment name	8
*	1
Command code	Variable
(	1
Field position	8
Relational operator	2
Field value	variable
)	1

The field position and the field value are connected by a relational operator which tells IMS how you want the two compared. The field value contains the data that you want IMS to use as the comparative value. When the field position specifies a field name then the field value must be the same length as the field specified by field name. When command code "O" is specified the field name can be replaced with a position and length. When the field position specifies a position and then the field value must be the same length as the length specified in the qualification. The position and length apply to the physical segment layout.

## SSA Command Code "O": Qualify by Position

IMS12  
APAR PM69378  
PTF UK81917

- Support for IMS Universal Drivers
  - Allows users to issue queries qualified on non-searchable fields and have a field's position and length generated automatically in the qualified SSA
  - SQL
    - Universal Drivers will detect a non-searchable field in the where clause based on database metadata and will internally convert the SSAList qualification
  - DLI
    - Universal Drivers will detect a non-searchable field in the SSAList based on database metadata and will internally convert the SSAList qualification

07-IMS Application: 55

IMS can only search on fields that are defined in the DBD – sequence/key or search fields. As a result, the Universal Drivers cannot support an SQL predicate containing columns that are not defined in a DBD.

The following is an example exception message users may receive upon issuing a SQL query containing columns not defined by a DBD: `java.sql.SQLException: com.ibm.ims.drda.base.DrdaException: com.ibm.ims.dli.SSAConversionException: An error occurred converting the SSA for segment <segmentName>: com.ibm.ims.dli.SSAQualificationConversionException: The field <fieldName> in database segment <segmentName> is not a searchable field.`

IMS has been enhanced to allow a search based on offset and length. This removes the requirement that fields need to be defined in the DBD. The Universal drivers have been enhanced to allow SQL predicates containing columns that are not defined in the DBD by internally converting to an offset and length.

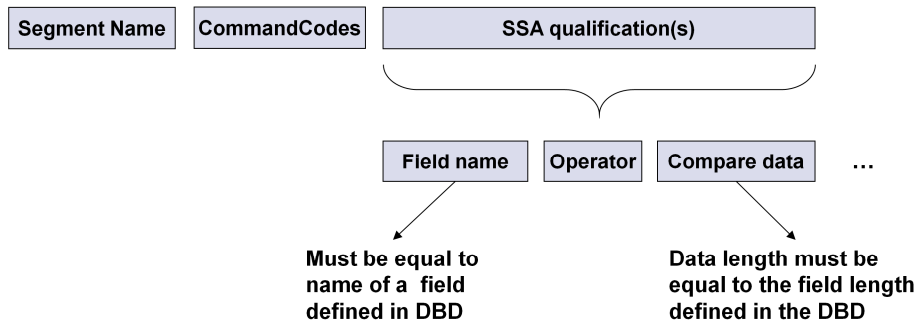
This removes any required special affinity between IMS and JDBC tooling products that generate queries. Qualify by position will greatly enhance IMS support of JDBC tools. JDBC tools such as Data Source Explorer and Cognos generate SQL queries and do not have a concept of searchable and non-searchable fields. Currently when a non-searchable field is placed in the WHERE(SSA) clause the search will fail. This enhancement provides a method of generating searchable SSAs for non-searchable fields.

If segment is in a logical relationship -> position & length apply to only the 1<sup>st</sup> physical segment, not the combined segment

If PCB uses SENFLD statements -> position applies to physical segment, and fields not available to the PCB cannot be searched

## SSA Enhancement - Qualify on Field Name

- Existing SSA with field names



## SSA Enhancement - Qualify on Field Name

- Existing SSA with field names

DBD

Field	Offset	Len
Labname	1	5
Street	10	20
State	30	2

Database

0	1	2	3
<u>12345678901234567901235678901</u>			
SVL	DEV	555 BAILEY AVE	CA
ARC	RSC	650 HARRY RD	CA

COBOL Copybook

Field	Offset	Len
Labname	1	5
Type	6	3
Street	10	20
State	30	2

```

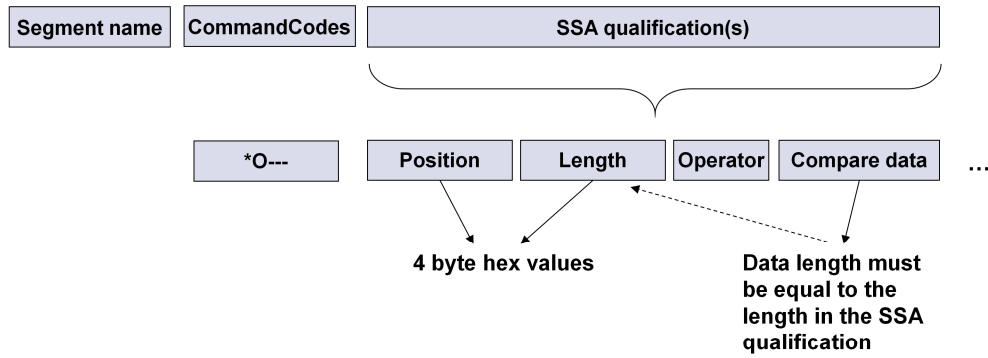
GU IBMLABS *-(LABNAME EQSVL )
GU IBMLABS *-(LABNAME EQARC )
GU IBMLABS *-(STATE EQCA)
GU IBMLABS *-(TYPE EQDEV) INVALID
    
```



**'AK' Status Code: SSA contains a field name not defined in the DBD**

## SSA Enhancement - Qualify by Position

- New SSA using "O" command code with position/length



## SSA Enhancement - Qualify by Position

- New SSA with "O" command code, position and length

DBD

Field	Offset	Len
Labname	1	5
Street	10	20
State	30	2

Database

0	1	2	3
12345678901234567901235678901			
SVL	DEV	555 BAILEY AVE	CA
ARC	RSC	650 HARRY RD	CA

COBOL Copybook

Field	Offset	Len
Labname	1	5
Type	6	3
Street	10	20
State	30	2

```

GU IBMLABS *O(0000000100000005EQSVL )
GU IBMLABS *O(0000000100000005EQARC )
GU IBMLABS *O(0000001E00000002EQCA)
GU IBMLABS *O(0000000600000003EQDEV)
    
```

Position    Length



'bb' Status Code: all segments returned successfully

## SSA Enhancement - New "SD" Status Code

### Explanation

- For call-level programs:
  - When command code "O" is specified with a segment search argument, the SSA is not allowed to contain position and length values for a segment of a Fast Path secondary index database
- For command-level programs:
  - When command code "O" is specified, the SSA format with position and length values is not allowed against a segment of a Fast Path secondary index database

### Programmer response

- Correct the SSA

Response from IMS

FINEST: [ibm][ims][drda] [t4][thread:1][tracepoint:2][Reply.fill]

[ibm][ims][drda][t4]	RECEIVE BUFFER: OPNQRYM	(ASCII)	(EBCDIC)
[ibm][ims][drda][t4]	0 1 2 3 4 5 6 7 8 9 A B C D E F	0123456789ABCDEF	0123456789ABCDEF
[ibm][ims][drda][t4] 0000	0010D0520001000A 2205000611490000	...R...."....I..	..}. .....
[ibm][ims][drda][t4]			
[ibm][ims][drda][t4]	RECEIVE BUFFER: QRYDSC	(ASCII)	(EBCDIC)
[ibm][ims][drda][t4] 0000	001CD05300010016 241A0676D0260000	...S....\$.v.&..	..}. .....
[ibm][ims][drda][t4] 0010	0671E0D000010671 F0E00000	.q....q....	..}\....0\..
[ibm][ims][drda][t4]			
[ibm][ims][drda][t4]	RECEIVE BUFFER: QRYDTA	(ASCII)	(EBCDIC)
[ibm][ims][drda][t4] 0000	011AD05300010114 241B000000038400	...S....\$. .....	..}. .....
[ibm][ims][drda][t4] 0010	0000000000000000 0000000000C4C5C4	.....	.....DED
[ibm][ims][drda][t4] 0020	C2D1D5F2F1F0F140 40C8D6E2D7C9E3C1	.....@@.....	BJN2101 HOSPITA
[ibm][ims][drda][t4] 0030	D3000000000CD9F1 F2F1F0F0F1F0F0F0	.....	L.....R121001000
[ibm][ims][drda][t4] 0040	F0C1D9F1F2F1F0F0 F1F0F0F0F0C10000	.....	0AR1210010000A..
<i>(lines removed to save space)</i>			
[ibm][ims][drda][t4] 00C0	F2F1F0F0F3F0F0F0 F0C1D9F1F2F1F0F0	.....	210030000AR12100
[ibm][ims][drda][t4] 00D0	F3F0F0F0F0C10000 0003840000000000	.....	30000A....d....
[ibm][ims][drda][t4] 00E0	0000000000000000 00C4C5C4C2D1D5F2	.....	.....DEDBJN2
[ibm][ims][drda][t4] 00F0	F1F0F14040C8D6E2 D7C9E3C1D3000000	...@@.....	101 HOSPITAL...
[ibm][ims][drda][t4] 0100	000CD9F1F2F1F0F0 F4F0F0F0F0C1D9F1	.....	..R1210040000AR1
[ibm][ims][drda][t4] 0110	F2F1F0F0F4F0F0F0 F0C1	.....	210040000A
[ibm][ims][drda][t4]			
[ibm][ims][drda][t4]	RECEIVE BUFFER: ENDQRYM	(ASCII)	(EBCDIC)
[ibm][ims][drda][t4] 0000	003CD00200010036 220B000611490004	.<....6"....I..	..}. .....
[ibm][ims][drda][t4] 0010	002CCC0200000000 0000000900000000	,.....	.....
[ibm][ims][drda][t4] 0020	00000000000000C4 C5C4C2D1D5F2F1F0	.....	.....DEDBJN210
[ibm][ims][drda][t4] 0030	F0C7C24040404040 404040FF	...@@@@@.	0GB



### ***SSA Enhancement - Performance Consideration***

- Performance will be similar to a search on a non-key field
- IMS will scan the database looking for field match(es)
- Qualification of the root key will help reduce the impact
- If business need requires searching on a non-key field
  - Consider defining the non-key field as a searchable field in the DBD

Performance should be no different than doing a call using a non-key search field.

It is possible that the call will scan the entire database and not find a match based on the position, length and qualifier, only to return a “GE” status code – just as it would today if you searched on a non-key field and did not find a match.

## ***SSA Enhancement - Qualify by Position***

- **Benefit**
  - Allows non-DBD defined fields to be included in SSAs
  - Allows a search on any field in a segment by specifying a position within the segment and the field length
  - Provides a method of generating searchable SSAs for non-searchable fields
  - Continuation of the IMS database modernization roadmap
    - Enhances and simplifies IMS support of JDBC tools
      - Discovery tools - Data Source Explorer
      - Reporting tools – Optim Data Studio
      - Analytics tools – COGNOS BI V10.2
      - Any JDBC tool that generates SQL calls !

07-IMS Application: 62

Users can now search for any field in a segment by specifying a position in the segment and a length. This feature allows non-DBD defined fields to be included in Segment Search Arguments and continues IBM's IMS database modernization efforts.

This enhances the IMS Simplification story by removing any required special affinity between IMS and JDBC tooling products that generates queries. Qualify by position will greatly enhance IMS support of JDBC tools. JDBC tools such as Data Source Explorer and Cognos generate SQL queries and do not have a concept of searchable and non-searchable fields. Currently when a non-searchable field is placed in the WHERE(SSA) clause the search will fail. This enhancement provides a method of generating searchable SSAs for non-searchable fields.

IMS 12 SPE Enhancement  
Qualify By Position  
*Reference Slides*

## SSA Enhancement - SQL to DLI translation

- **Connection.nativeSQL(String)** method will display the DLI equivalent of an SQL query
- `SELECT * FROM LARGE2.GBO02 WHERE GBO01.FIRSTNAME='KIN'`

```
GHU HOSPITAL
  GBO01  *O((x'1F',x'A')EQKIN      )
  GBO02

[LOOP]

GHN HOSPITAL
  GBO01  *O((x'1F',x'A')EQKIN      )
  GBO02
```

07-IMS Application: 64

For troubleshooting and diagnostic purposes, the `Connection.nativeSQL(String)` method will come in handy.

By using it, one can see the DLI call generated by IMS in place of the JDBC SQL call.

## SSA Enhancement - SSAs in the DRDA log

- The Universal driver logs will show the exact SSA that is used in the IMS DLI Call
- `SELECT * FROM LARGE2.GBO02 WHERE GBO01.FIRSTNAME='KIN'`

SEND BUFFER: SSALIST	(ASCII)	(EBCDIC)
004ED00300010048 CC060006C9050003	.N.....H.....	.+}.....I...
000DC906C8D6E2D7 C9E3C1D3400024C9	.....@.\$.	..I.HOSPITAL ..I
06C7C2D6F0F14040 405CD64D0000001F	.....@@@\..M....	.GBO01 *O(.....
0000000AC5D8D2C9 D540404040404040	.....@@@@@@@	....EQKIN
5D000DC906C7C2D6 F0F240404040	].....@@@	)..I.GBO02

Tracing must be enabled for the universal drivers.  
The DRDA log can be sent to a file, the console or the joblog.

Tracing can be enabled for the universal drivers. The DRDA log can be sent to a file, console or joblog.

[http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/topic/com.ibm.ims12.doc.apg/ims\\_odbdl4jtracing.htm](http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/topic/com.ibm.ims12.doc.apg/ims_odbdl4jtracing.htm)

### Sample SQL and Trace output:

SQL Query: `SELECT * FROM PCB01.HOSPITAL`

FINEST: [ibm][ims][drda] [t4][thread:1][tracepoint:1][Request.flush]

[ibm][ims][drda][t4]	SEND BUFFER: OPNQRY	(ASCII)	(EBCDIC)
[ibm][ims][drda][t4]	0 1 2 3 4 5 6 7 8 9 A B C D E F	0123456789ABCDEF	0123456789ABCDEF
[ibm][ims][drda][t4] 0000	0029D05100010023 200C00062141001F	.)Q...# ...!A..	..}.....
[ibm][ims][drda][t4] 0010	0009C907D7C3C2F0 F100082114000080	.....!....	..I.PCB01.....
[ibm][ims][drda][t4] 0020	0000082156000003 69	...!V...i	.....
[ibm][ims][drda][t4]	SEND BUFFER: DLIFUNC	(ASCII)	(EBCDIC)
[ibm][ims][drda][t4] 0000	0012D0530001000C CC05D9C5E3D9C9C5	...S.....	..}.....RETRIE
[ibm][ims][drda][t4] 0010	E5C5	..	VE
[ibm][ims][drda][t4]	SEND BUFFER: AIB	(ASCII)	(EBCDIC)
[ibm][ims][drda][t4] 0000	001BD05300010015 CC010009C901D7C3	...S.....	..}.....I.PC
[ibm][ims][drda][t4] 0010	C2F0F10008C90400 000384	.....	B01..I....d
[ibm][ims][drda][t4]	SEND BUFFER: RTRVFLD	(ASCII)	(EBCDIC)
[ibm][ims][drda][t4] 0000	0012D0530001000C CC04000000020000	...S.....	..}.....
[ibm][ims][drda][t4] 0010	000C	..	..
[ibm][ims][drda][t4]	SEND BUFFER: SSALIST	(ASCII)	(EBCDIC)
[ibm][ims][drda][t4] 0000	001DD00300010017 CC060006C9050001	.....	..}.....I...
[ibm][ims][drda][t4] 0010	000DC906C8D6E2D7 C9E3C1D340	.....@	..I.HOSPITAL

(Trace output for IMS Response will follow)

### SSA Enhancement - DFSDDLT0 Examples

GU calls with CmdCode "O" searching at position x'01' for x'10' bytes

\*\*\*\*\*

IMS 1210 TEST PROGRAM OUTPUT \*\* BEGIN TEST \*\*\*\*TIME= 16.22.28.84 DATE= 12.128

WTOR CALL: WTOR Start of GBO BATCH1  
STATUS INPUT: S11 1 1 1 1 01  
0002 OF 0005 PCB SELECTED = DBD SELECTED = DBOHIDK5  
COMMENTS GU K1

WTO CALL: WTO K1 \*O(00010010EQ0000000000000000)  
CALL=GU SEG=K1 COMND=O  
00000001  
FIELD=00010000 OPER== VALUE=0000000000000000  
SEGMENT =(00000000000000000000000000000000@STARTING@POINT@)  
DBPCB LEV=01 SEG=K1 RET CODE= K FDB LEN=0020 KEY FDB=(000000000000000000000000)

WTO CALL: WTO K1 \*O(00010010GTAAAAAAAAAAAAAAAA)  
CALL=GU SEG=K1 COMND=O  
00000001  
FIELD=00010000 OPER-> VALUE=AAAAAAAAAAAAAAAA  
SEGMENT =(00000000000000000000000000000000@STARTING@POINT@)  
DBPCB LEV=01 SEG=K1 RET CODE= K FDB LEN=0020 KEY FDB=(000000000000000000000000)

Sample DFSDDLT0 control cards and output

## SSA Enhancement - DFSDDLT0 Examples

### GU call using key field vs. GU call using CmdCode "O" & position

```

*****
IMS 1210 TEST PROGRAM OUTPUT ** BEGIN TEST ****TIME= 15.08.19.61 DATE= 12.292

WTOR CALL:      WTOR  Start of batch app
WTOR REPLY:
STATUS INPUT: S 1 1 1 1 1  DBOHIDK5
0002 OF 0010 PCB SELECTED =          DBD SELECTED = DBOHIDK5

WTOR CALL:      WTOR  (K1 (K1 EQOKEY K1 1000)
-----
CALL=GU         SEG=K1  (FIELD=K1 (OPER=EQ (VALUE=OKEY K1 1000)
-----
SEGMENT        =(OKEY K1 1000@@STARTING@@POINT@@          )

DBPCB  LEV=01 SEG=K1  (RET CODE= (KFDB LEN=0020 KEY FDB=(OKEY K1 1000)

WTOR CALL:      WTOR  (K1 *(00020013xEQKEY K1 1000)
-----
CALL=GU         SEG=K1  (COMND=O
                   00000001
                   FIELD=00020003 (OPER=EQ (VALUE=KEY K1 1000)
-----
SEGMENT        =(OKEY K1 1000@@STARTING@@POINT@@          )

DBPCB  LEV=01 SEG=K1  (RET CODE= (KFDB LEN=0020 KEY FDB=(OKEY K1 1000)

```

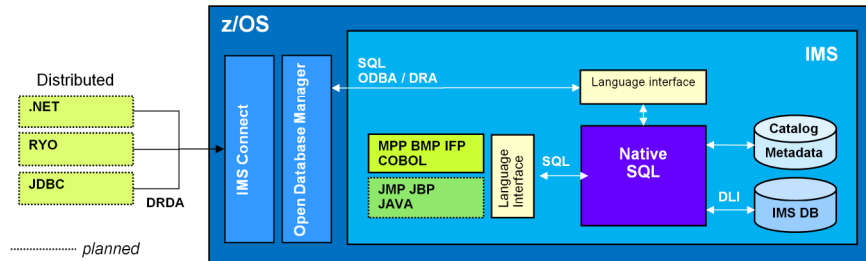
Sample DFSDDLT0 control cards and output

# IMS Native SQL Support for COBOL



## IMS 13 SQL Support

- Native SQL COBOL and distributed applications (.NET/JDBC)
- Provides standard SQL keywords to easily access IMS data
  - ✓ SELECT, INSERT, UPDATE, DELETE
  - ✓ Uses Dynamic SQL programming model
  - ✓ Converts SQL statements to DL/I calls
  - ✓ Supports a subset of SQL keywords that are currently supported by IMS Universal JDBC driver
- Uses database metadata in IMS Catalog
  - ✓ No need to generate metadata for use in applications



07- IMS Application: 69

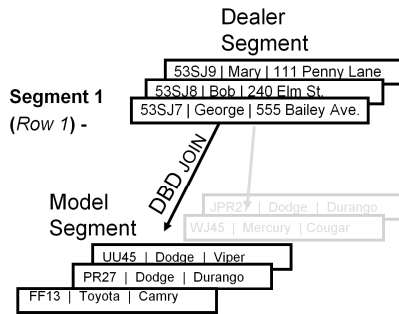
IMS 13 delivers an SQL engine that is capable of evaluating SQL statements (a subset of the SQL keywords supported by the IMS 13 Universal JDBC Driver) and converts them to DL/I calls.

The IMS Native SQL Engine is used to support COBOL and DRDA access.

Note The IBM IMS Enterprise Suite Data Provider for Microsoft .NET and IMS 13 TYPE-4 Universal Drivers will provide the DRDA support via the service process.

## Hierarchical to Relational Terminology Mapping

### Hierarchical Design



**Note:** Segment Names ~ Table Names  
 Segment Instances ~ Table Rows  
 Segment Field Names ~ Column Names  
 Segment unique key ~ Table primary key  
 IMS foreign key field ~ Table foreign key  
 PCB ~ Schema

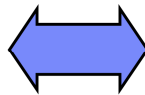
### Relational Design

**Dealer Table**

	DealerID	DealerName	DealerAddress
Row 1 -	0	53SJ7	George   555 Bailey Ave.
	1	53SJ8	Bob   240 Elm St.
	2	53SJ9	Mary   111 Penny Ln.
Row N -	...	...	...

**Model Table**

	ID	Make	Model	Dealer	
Row 1 -	UU45	Dodge	Viper	53SJ7	0
	PR27	Dodge	Durango	53SJ7	0
	FF13	Toyota	Camry	53SJ7	0
	JR27	Dodge	Durango	53SJ8	1
	WJ45	Mercury	Cougar	53SJ8	1
Row N -	...	...	...	...	...



Relational JOIN

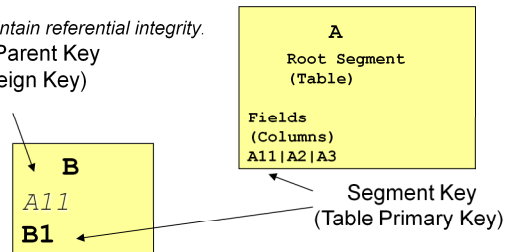
This slide provides a mapping of IMS hierarchical database concepts and relational database concepts.

IMS Native SQL Processor views a Segment as a Table, A field as column and segment instance as a row.

## Solution highlights - IMS foreign keys Referential constraint

- IMS cannot insert a dependent segment unless the parent segment exists
  - IMS has built-in foreign keys in each segment which are comprised of keys of each parent segment
    - Exist in the key feedback area not physically stored in the IMS database
      - For INSERT operations the Foreign Keys are used to establish the correct position in the hierarchy
        - Values aren't actually inserted as they already exist in the database

IMS Foreign Key - *maintain referential integrity.*  
(Segment Parent Key  
Table Foreign Key)

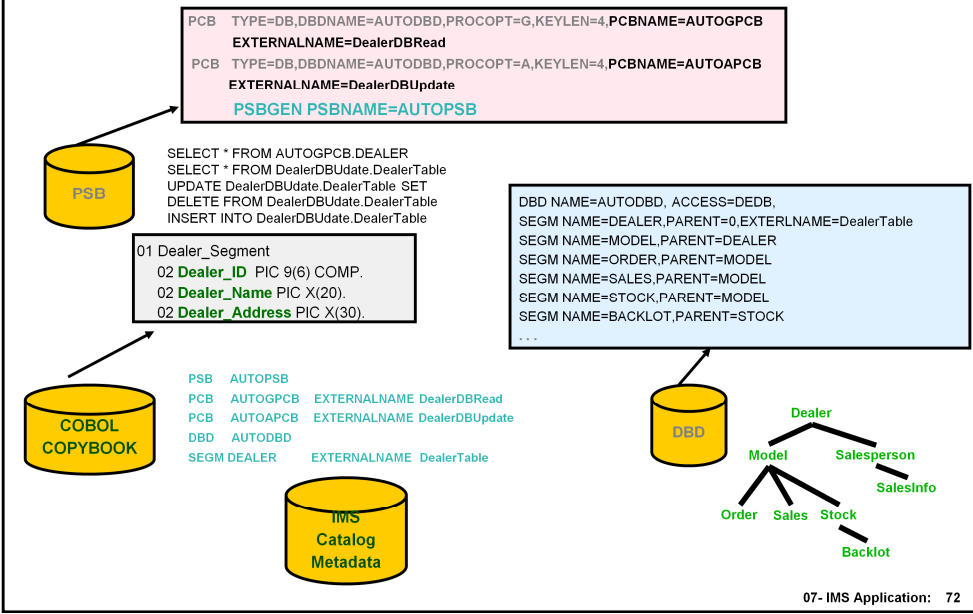


07-IMS Application: 71

This slide provides a mapping of IMS hierarchical database concepts and relational database concepts. IMS SQL views Segment as a Table, A field as column and segment instance as a row.

The purpose of the IMS foreign key fields is to maintain referential integrity, similar to foreign keys in relational databases. This allows SQL SELECT, INSERT, UPDATE, and DELETE queries to be written against specific tables and columns located in a hierarchic path.

# IMS Catalog Metadata and SQL



This slide shows how information from PSB, DBD and COBOL copybooks is used to populate IMS Catalog and how SQL statements can be coded with External Names to access IMS databases

### ***IMS 13 SQL support for COBOL Solution Highlights***

- **SQL support for COBOL**
  - Use Dynamic SQL as a query language for COBOL programs to access IMS database
  - EXEC SQLIMS is the interface to execute IMS SQL calls
  
- **Native SQL in IMS**
  - Process SQL calls natively by the IMS subsystem
  - Still perform DL/I database call processing to IMS DB
  - Provide a consolidated way for SQL processing
  - Uses database metadata in IMS Catalog
  
- **Support IMS TM/DB (MPP, IFP, BMP) and DBCTL BMP**

IMS COBOL application programs running in MPP,IFP,BMP for IMS DB/DC and DBCTL BMP .

- DL/I Batch, CICS, and DB2 Stored Procedures are not supported
- SQL to GSAM is not supported.

Static SQL refers to the type of SQL statement that is embedded inside an application and its source form is known before runtime. This is not supported by IMS SQL for COBOL.

Dynamic SQL refers to the type of SQL statement that the actual content of the statement is constructed and prepared at run time. Segment and field names associated with the SQL calls are defined in the program source and are pre-processed when the program is being compiled.

### ***Solution Details – Dynamic SQL***

- **Enable SQL statement to be constructed at runtime**
  - No need to hard code SQL statement in the application
  - Segment and field names associated with the SQL calls are not known at compile time
  - SQL accepts input in the form of a character string
- **Use Prepare call to process SQL statement**
  - Parse SQL statement for syntax and semantics validation at runtime. No bind process.
  - Convert SQL artifacts to DLI
  - Statement can be prepared once and then execute many times

Programs that contain embedded *dynamic* SQL statements must be precompiled like those that contain static SQL, but unlike static SQL, the dynamic statements are constructed and prepared at run time.

The source form of a dynamic statement is a character string that is passed to IMS by the program using the SQL PREPARE statement. A statement that is prepared using the PREPARE statement can be referenced in a DECLARE CURSOR, DESCRIBE, or EXECUTE statement.

In IMS 13, only dynamic SQL is supported for COBOL.

## SQL Statements

### ▪ Data Access

- SELECT... FROM... to retrieve data
- INSERT INTO... VALUES... to insert data
- UPDATE... SET... to update data
- DELETE FROM... to delete data
- WHERE... AND... OR... to perform conditional selection of data

### ▪ Pre-compiler directives

- DECLARE CURSOR, STATEMENT... to declare cursor, statement
- INCLUDE SQLIMSCA, SQLIMSDA... to generate SQLIMSCA and SQLIMSDA structures
- WHENEVER... to handle errors and warnings

SQL statements let you retrieve, insert, update, or delete data in IMS databases. When you write an SQL statement, you specify what you want done, not how to do it. To access data, for example, you need only to name the segment and fields that contain the data. You do not need to describe how to get to the data.

In accordance with the relational model of data:

- The database is perceived as a set of tables.
- Relationships are represented by values in tables.
- Data is retrieved by using SQL to specify a *result table* that can be derived from one or more tables.

IMS transforms each SQL statement, that is, the specification of a result table, into a sequence of operations for data retrieval or modifications. All executable SQL statements must be prepared before they can run.

### ***Solution Details – Key application elements***

- Delimit SQL statement using EXEC SQLIMS ... END-EXEC
- Dynamic SQL programming model
  - Must call PREPARE to process SQL statement
- Host variables
  - Use for both send and receive data processed by IMS
- SQL communication area (SQLIMSCA)
  - Structure used by IMS to provide status feedback
  - SQLIMSCODE (error code), SQLIMSSTATE (state), SQLIMSERRM (error message)
- SQL description area (SQLIMSDA)
  - DESCRIBE statement IMS provides information to an application program about a prepared statement
  - FETCH statement application program describes a host variable or buffer that is to be used to contain an output value from a row of the result.

If your program includes any of the following statements, you must include an SQLIMSDA in your program:

- DESCRIBE *statement-name* INTO *descriptor-name*
- FETCH ... INTO DESCRIPTOR *descriptor-name*

An SQLIMSDA is a collection of variables that is required for execution of the SQLIMS DESCRIBE statement, and can be optionally used by the FETCH statements. An SQLIMSDA can be used in a DESCRIBE statement, modified with the addresses of host variables, and then reused in a FETCH statement. The meaning of the information in an SQLIMSDA depends on the context in which it is used. For DESCRIBE, IMS™ sets the fields in the SQLIMSDA to provide information to the application program. For FETCH, the application program sets the fields in the SQLIMSDA to provide IMS with information:

#### **DESCRIBE *statement-name***

With the exception of SQLIMSN, IMS sets fields of the SQLIMSDA to provide information to an application program about a prepared statement. Each SQLIMSVAR occurrence describes a column of the result table.

#### **FETCH**

The application program sets fields of the SQLIMSDA to provide information about host variables or output buffers in the application program to IMS. Each SQLIMSVAR occurrence describes a host variable or output buffer. For FETCH, each SQLIMSVAR occurrence describes a host variable or buffer in the application program that is to be used to contain an output value from a row of the result.



## Handling errors

- **SQL communication area (SQLIMSCA)**
  - Structure used by IMS to provide status feedback
  - The SQL INCLUDE statement is used in the COBOL application to provide the declaration of the SQLIMSCA

```
EXEC SQLIMS INCLUDE SQLIMSCA
```

- **The main elements in the SQLIMSCA are:**
  - SQLIMSCODE – A return code represents a successful or failed SQL operation
  - SQLIMSSTATE – Common codes for error conditions which conform to the SQL standard
  - SQLIMSERRM – Error message text

An SQLIMSCA is a structure or collection of variables that is updated after each SQL statement executes. An application program that contains executable SQL statements must provide exactly one SQLIMSCA.

In COBOL, the INCLUDE statement can be used to provide the declaration of the SQLIMSCA.

- When IMS processes an SQL statement, it places return codes that indicate the success or failure of the statement execution in SQLIMSCODE and SQLIMSSTATE.
- When IMS processes a FETCH statement, and the FETCH is successful, the contents of SQLIMSERRD(3) in the SQLIMSCA is set to the number of returned rows.
- When IMS processes a FETCH statement, the contents of SQLIMSCODE is set to +100 if the last row in the segment has been returned with the set of rows.
- When IMS processes an UPDATE, INSERT, or DELETE statement, and the statement execution is successful, the contents of SQLIMSERRD(3) in the SQLIMSCA is set to the number of rows that are updated, inserted, or deleted.

### **SQL descriptor area (SQLIMSDA)**

- SQLIMSDA stores information about prepared SQL statements or host variables.
  - SQLIMSDA header
  - SQLIMSVAR entry
    - each column or host variable is described

```
EXEC SQLIMS INCLUDE SQLIMSDA
```

- Can be read by IMS or the application program
  - Read by application program after a DESCRIBE statement
  - Read by IMS for the host variables set by the application program

An SQLIMSDA consists of four variables, a header, and an arbitrary number of occurrences of a sequence of variables collectively named SQLIMSVAR. It is a collection of variables that is required for execution of the SQLIMS DESCRIBE statement, and can be optionally used by the FETCH statements. An SQLIMSDA can be used in a DESCRIBE statement, modified with the addresses of host variables, and then reused in a FETCH statement.

The meaning of the information in an SQLIMSDA depends on the context in which it is used. For DESCRIBE, IMS™ sets the fields in the SQLIMSDA to provide information to the application program. For FETCH, the application program sets the fields in the SQLIMSDA to provide IMS with information:

#### **DESCRIBE *statement-name***

With the exception of SQLIMSN, IMS sets fields of the SQLIMSDA to provide information to an application program about a prepared statement. Each SQLIMSVAR occurrence describes a column of the result table.

#### **FETCH**

The application program sets fields of the SQLIMSDA to provide information about host variables or output buffers in the application program to IMS. Each SQLIMSVAR occurrence describes a host variable or output buffer.

For FETCH, each SQLIMSVAR occurrence describes a host variable or buffer in the application program that is to be used to contain an output value from a row of the result.

## ***IMS Native SQL Support for COBOL solution***

- Compile IMS program using COBOL compiler with the SQL(IMS) option
  - Create an executable program to be run in IMS.
  - IMS co-processor knows when a particular SQL statement begins and ends by the following delimits for SQL statements:

```
– EXEC SQLIMS  
   SQL-STATEMENT  
– END-EXEC .
```

- Translate SQL statement to a COBOL CALL statement

```
*EXEC SQLIMS FETCH . . .  
CALL SQLTDLI USING SQL-PARMLIST
```

- **SQLTDLI**
  - non-language-specific interface added to DFSLI000

Any segment and field names associated with the SQL calls are defined in the program source and are pre-processed when the program is being pre-compiled. There is no need to use a separate precompile step.

Each executable SQL statement calls IMS through the SQLTDLI language interface with a list of parameters that is generated by the IMS co-processor function. The parameter list contains a collection of addresses for the input and output host variables, the SQL statement string, the SQL call type, the execution parameters and other information. IMS uses this information to determine how the call should be processed

## Sample COBOL SQL

```
WORKING-STORAGE SECTION.  
* Declare SQLIMSCA  
* SQL communications area (SQLIMSCA) that for your COBOL program can use to  
  check whether an SQL statement executed successfully.  
EXEC SQLIMS INCLUDE SQLIMSCA END-EXEC.  
* Declare HOST variables for SQL statement and result data  
01 SQL-STATEMENT  
   49 SQL-STATEMENT-LEN   PIC S9(4) COMP.  
   49 SQL-STATEMENT-TEXT PIC X(100).  
01 HOSPITAL-RESULT-ROW  
   05 HOSPLL   PIC S9(3) BINARY.  
   05 HOSPCODE PIC X(12).  
   05 HOSFNAME PIC X(17).
```

Define an SQL communications area (SQLIMSCA) for your COBOL program which is used to check whether an SQL statement executed successfully.

## Sample COBOL SQL

```
PROCEDURE DIVISION.  
* Declare Cursor for the Prepared Statement  
* use cursors to select a set of rows and to process one row at a time  
EXEC SQLIMS  
    DECLARE CURSOR cursor-name for prepared-statement-name  
END-EXEC.  
* Load SQL statement in the COBOL variable  
MOVE "SELECT * FROM PCB01.HOSPITAL" TO SELECT-STATEMENT-TXT.  
* Prepared SQL statement string for processing  
* Only one prepared statement at a time is allowed for  
  database access  
EXEC SQLIMS  
    PREPARE prepared-statement-name FROM :SQL-STATEMENT  
END-EXEC.
```

Use cursors to select a set of rows and then process one row at a time. Only one prepared statement at a time is allowed for database access and a statement cannot be prepared if a cursor is open for another statement.

### Sample COBOL SQL (Cont'd)

```
* Open Cursor
* only support 1 PCB, 1 cursor
EXEC SQLIMS
    OPEN cursor-name
END-EXEC.
* Execute SQL statement
* Fetch data from IMS into host variable until no more data is found
PERFORM FETCH-PROC
UNTIL SQLCODE EQUAL 100.
:
FETCH-PROC.
    EXEC SQLIMS
        FETCH cursor-name INTO :HOSPITAL-RESULT-ROW
    END-EXEC.
:
* Close Cursor
EXEC SQLIMS
    CLOSE cursor-name
END-EXEC.
```

Only one prepared statement at a time is allowed for database access and a statement cannot be prepared if a cursor is open for another statement. Only support 1 PCB, 1 cursor.

***IMS coprocessor***

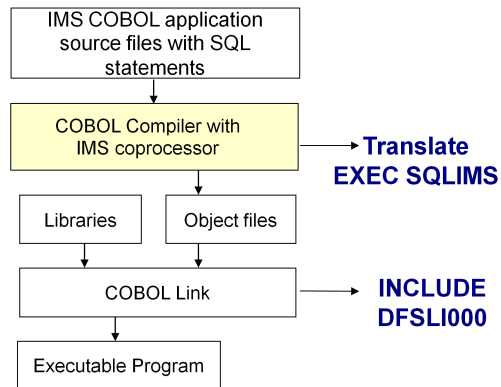
- Compile IMS SQL COBOL application with IMS coprocessor
- Pre-process EXEC SQLIMS statements in COBOL source
- Integrated with Enterprise COBOL V5.1
- Specify 'SQLIMS' compiler option to compile COBOL program with IMS SQL calls

### Sample JCL for compile

```
//*****  
//* COMPILING IMS COBOL SQL PROGRAM  
//*****  
//COBOL1 EXEC PGM=IGYCRCTL,  
// PARM='LIST,XREF,CP(37),SQLIMS("APOSTSQL"),DUMP,LIB,DYNAM'  
//STEPLIB DD DSN=IGYV5R10.TRIAL.SIGYCOMP,DISP=SHR  
// DD DSN=IGYV5R10.TRIAL.SIGYMAC,DISP=SHR  
// DD DSN=IMSBLD.IMSTS%.CRESLIB,DISP=SHR  
//SYSLIB DD DSN=IGYV5R10.TRIAL.CEEZ1D0.SCEERUN,DISP=SHR  
// DD DSN=USER.PRIVATE.PROCLIB,DISP=SHR  
//SYSPRINT DD SYSOUT=A  
//SYSLIN DD DSN=&&LOADSET,DISP=(MOD,PASS),  
// UNIT=SYSDA,SPACE=(80,(500,200))
```



## IMS COBOL SQL application compiled and linked

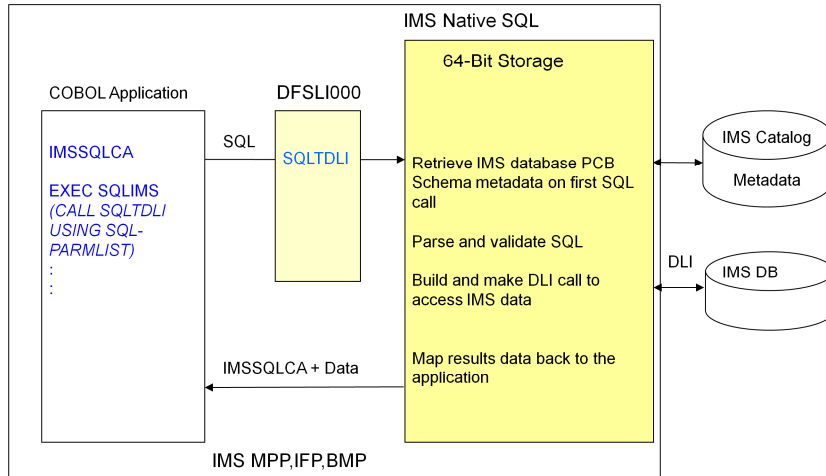


07- IMS Application: 85

During compilation, the IMS co-processor takes an SQL statement enclosed between the EXEC SQLDLI and END-EXEC keywords and processes it as follows:

- Translate an executable SQL statement to a COBOL CALL statement to invoke the IMS language interface
- Initialize and map host variables for input and output data
- Include and add data structures for SQL processing

## IMS SQL Call Request Handler



07- IMS Application: 86

IMS SQL is an SQL-to-DLI processor or translator that intercepts a SQL calls and translates them into DLI calls for execution. This diagram shows an example on how a SQL call from a COBOL application is processed by IMS:

1. The COBOL program is first pre-compiled by the IMS co-processor which converts the EXEC SQLDLI statements to COBOL call statements to the new IMS language interface (SQLTDLI).

This interface is used to pass the SQL statement with the host variable information to IMS for processing.

2. Once the COBOL program runs, the SQLTDLI interface uses the native SQL processor.

3. The native SQL processor retrieves the corresponding PSB/DBD metadata from catalog based on the SQL call.

4. The SQL statement is parsed and validated by the native SQL processor based on the IMS SQL syntax rules and metadata information.

5. The native SQL processor builds and executes the appropriate DLI call(s) based on the SQL statement to access IMS DB.

6. The native SQL processor processes the result data from IMS DB using 64 bit storage and performs column / aggregation functions if needed.

7. The native SQL processor returns and puts the data back to the host variables for the COBOL application

## SQL considerations and restrictions for COBOL

- A subset of SQL keywords is supported.
  - Aggregate functions and XML are not supported by COBOL SQL in SELECT statements.
  - SQL COMMIT and ROLLBACK keywords are not supported.
    - use IMS DB system services call to commit or roll back your database changes
- Batch and DB Batch are not supported.
- IBM® CICS® Transaction Server for z/OS® and DB2® for z/OS stored procedures to IMS are not supported..
- The IMS catalog must be enabled to use SQL support for COBOL..
- Specify at least 12M for your IMS dependent region size for running a COBOL SQL application.
- Only one cursor and SQL statement can be active at a time in the application.
- For IMS database services, GSAM, IMS TM, and message processing services, continue to use DL/I API.
- Dynamic SQL statement is supported. Static SQL is not supported
- Only EBCDIC CCSID 37 and 1140 codepages for the COBOL CODEPAGE option are supported.
- Note The IMS Universal Database resource adapter and IMS Universal JDBC driver internally manage the LL field on behalf of the application
  - For SQL support for COBOL, COBOL applications are responsible for managing the LL field

Dynamic SQL refers to the type of SQL statement where the actual content of the statement is constructed and prepared at run time. For an example, a dynamic SQL statement can be passed to the application as an input message, or the statement is dynamically constructed based on other values at runtime. For Dynamic SQL, the statement type, table or column names are not known when the program is pre-compiled. Since the source of the SQL statement is not known during pre-compile time, the SQL statement is parsed and validated by the database during runtime each time the SQL statement is executed

### Using the LL field with COBOL

The LL field is treated as a normal column in the standard SQL result set for all operations. You can read, insert, or update the LL field data directly. Deleting the LL field data also deletes the rest of the associated database record. To set a field to the null state, set the length of the segment (the value of the LL field column) to be smaller than the offset of the field within the segment.

The LL field is 2 bytes long and must be handled as BINARY, SHORT, or USHORT data.

## ***Performance***

### ▪ Recommendations

- Fully qualify all tables (segments) and columns (fields) in SQL statements
  - Specify the schema (PCB) name
- Always use PREPARE call for SQL statement that is going to be executed multiple times
- Consider using FETCH or cursors to select a set of rows and then process the set either one row at a time or one rowset at a time

## ***Documentation***

- **Application Programming Guide**
  - Programming for IMS -> Application programming -> Application Programming for SQL
  
- **Application Programming Reference**
  - Programming for IMS -> Application programming APIs -> SQL programming reference
  
- **Messages and Codes**
  - Troubleshooting for IMS -> IMS component codes -> SQL codes

## ***Prerequisites***

- **Software requirements**
  - IMS 13 + PTF UK98028
  - IMS Catalog function is required
  - COBOL compiler with IMS co-processor function
    - Enterprise COBOL Developer Trial z/OS V5.1 or
    - Enterprise COBOL for z/OS V5.1 + APAR PM92523
  - Note: COBOL V5.1 requires z/OS 1.13 and above
  
- **Hardware requirements**
  - Same as IMS 13 and COBOL V5.1

# IMS Universal Driver Enhancements

### ***IMS Catalog Access***

- For retrieving database and application metadata
- Provides greater application scalability
- Supports complex datatypes (arrays and structures)
- Alternate mapping of fields within a segment
- SSA qualifier based on an offset and length instead of a field name
  - search fields do not need to be defined within the DBD source.

The IMS Universal Drivers are enhanced to support the IMS catalog for retrieving database and application metadata. This support allows for greater application scalability and support for complex datatypes (arrays and structures), and segment maps, which are different cases (sets of fields) within a segment where each case is only valid for a unique value of the map's control field. The drivers have also been enhanced with the ability to search on a qualifier based on an offset and length within a segment instance instead of a field name. This enhancement allows for greater search capabilities as search fields do not need to be defined within the DBD source.



## **IMS 13 APAR PM90041: All users of the IMS V13 Universal Drivers**

- This APAR contains various performance enhancements
- Connection Properties:
  - Property Name: signedCompare
    - Behavior:
      - Set to true ranged queries over signed data types.
      - Set to false, standard binary comparisons are performed based on the binary representation of the data type value.
      - Setting the value to false can increase performance but might result in incorrect results.
    - The signedCompare property applies to all environments.
  - PropertyName: t2OutputBufferSize
    - Behavior:
      - Sets the size of the output buffer in bytes for the results from a SELECT operation
    - The t2OutputBufferSize property applies only to Type-2 connections

The two new Connection Properties listed below are added to the list of topics in "IMS Version 13 Application Programming" (SC19-3646-00):

Under the following high level path:

Java application development for IMS > Programming with the IMS Universal drivers > Programming using the IMS Universal DB resource adapter > Connecting to IMS with the IMS Universal DB resource adapter  
> Connecting using the IMS Universal JCA/JDBC driver in a managed environment.  
> Connecting using the IMS Universal DB resource adapter in a managed environment

Under the following high level path:

Java application development for IMS > Programming with the IMS Universal drivers > Programming with the IMS Universal JDBC driver > Connecting to IMS using the IMS Universal JDBC driver:  
> Connecting to an IMS database using the JDBC DataSource interface  
> Connecting to an IMS database by using the JDBC DriverManager interface

The following two Connection Properties that have been added are:

- signedCompare: When this property is set to true, special SSAs are generated to support ranged queries over signed data types. If the property is set to false, standard binary comparisons are performed based on the binary representation of the data type value. Setting the value to false can increase performance but might result in incorrect results.

- t2OutputBufferSize: The size of the output buffer in bytes for the results from a SELECT operation for a Type-2 connection.

(this page intentionally left blank)