

Unit 2

Transaction Flow and Message Flow

(applies to DB/DC and DCCTL environments)

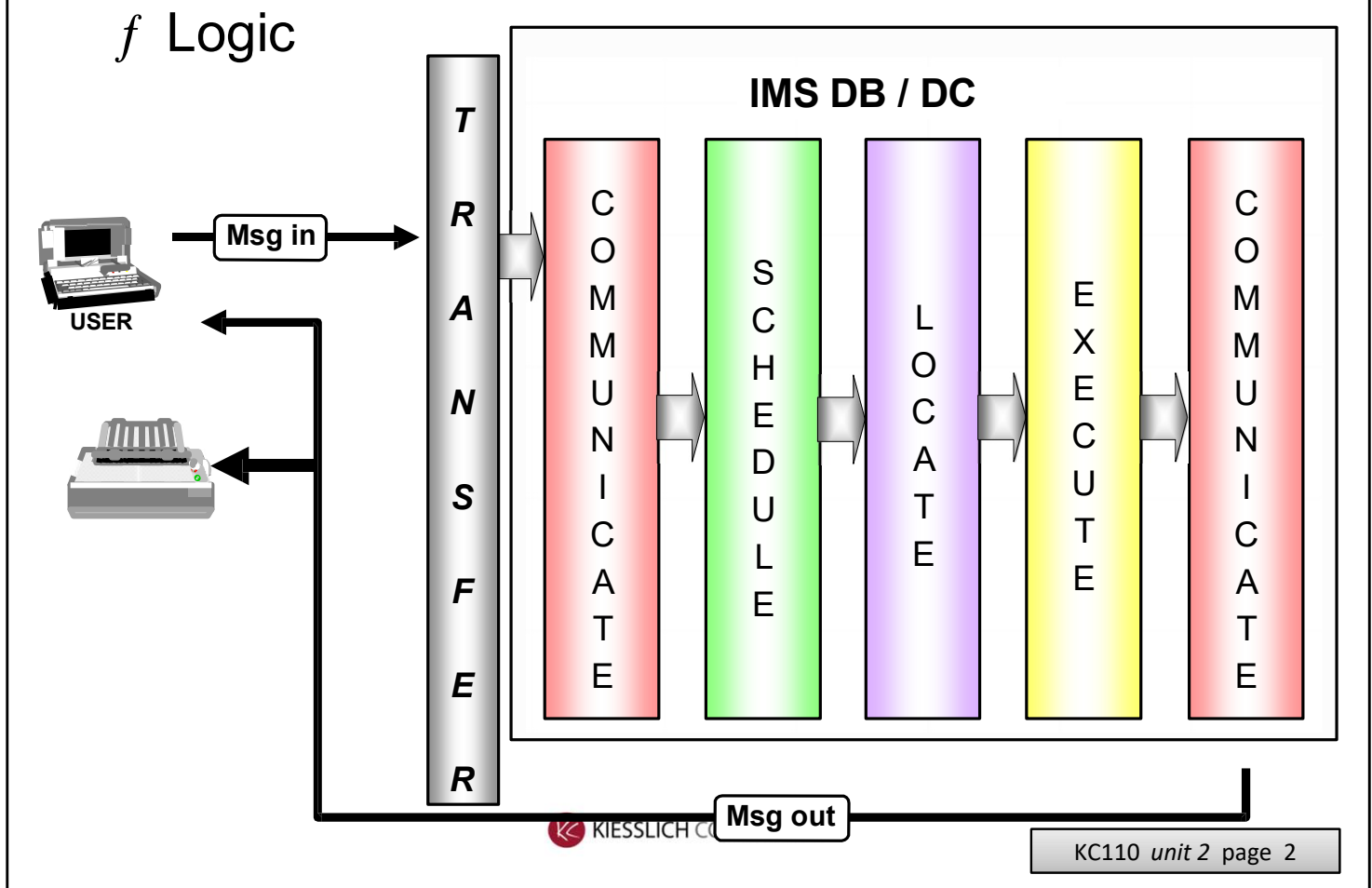
What this unit is about:

An important function of an IMS system is to receive input from a large number and variety of input devices. Once the input has been received, IMS must classify the input prior to performing additional processing that will eventually result in some response being returned to the source of the original input.

After completing this unit, you should be able to:

- Identify the major storage pools used by IMS TM that are related to input messages
- List the three different types of IMS input messages
- Identify the purpose of LTERM as well other terminal-related control blocks
- Describe how messages are organized and managed by the IMS Queue manager
- Understand the differences between the Short and Long Message Queue DS LRECLs
- Describe conversational transactions and how a transaction is designated as being conversational

Message processing overview



The Transfer function is performed by VTAM .. (COMM controller) ;
 or APPC / OTMA (via OTMA clients as DSNAIMS, MQ via OTMACON parm and "IMS Bridge", or IMS Connect),
 depending on the source of the input.

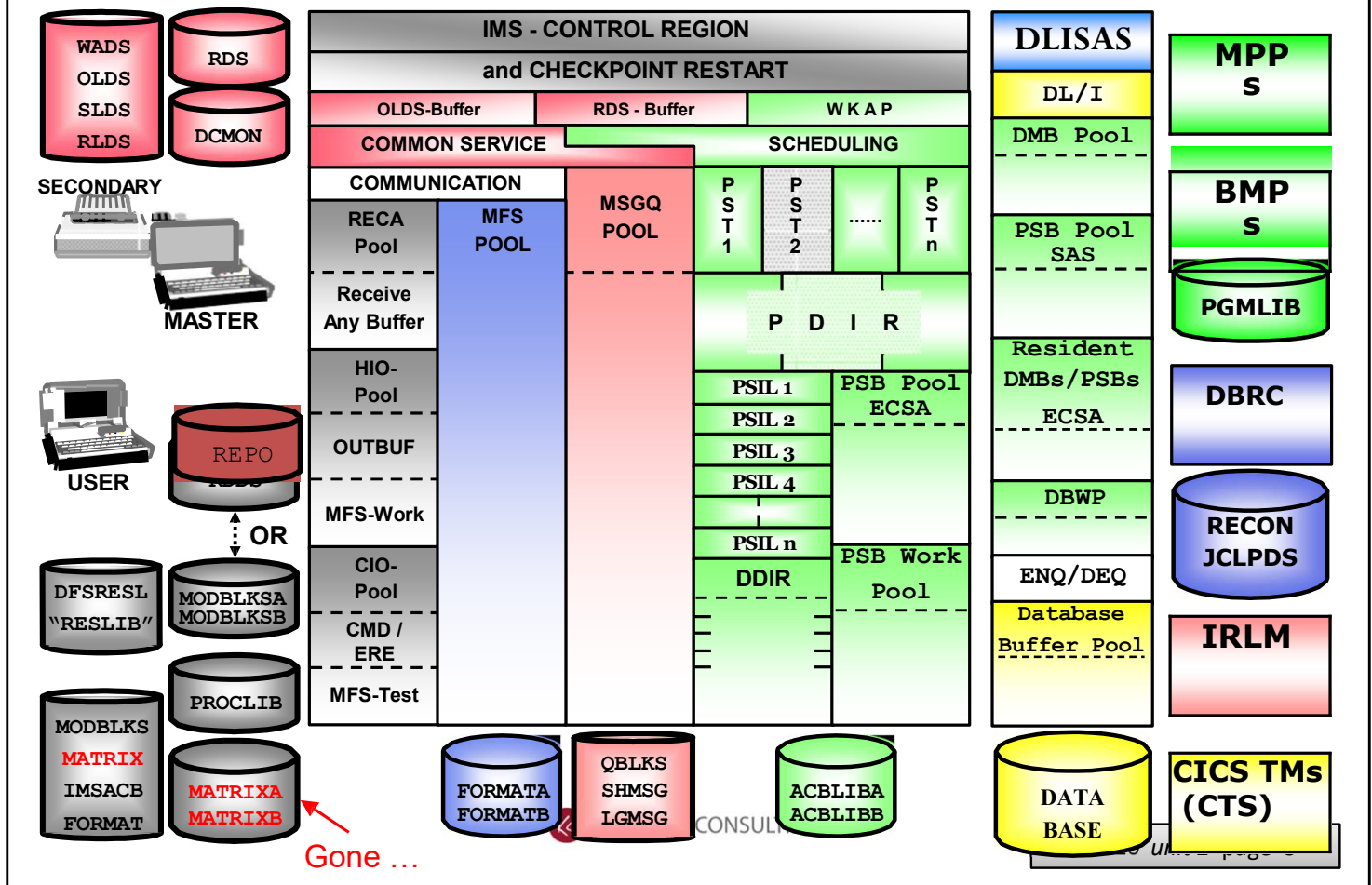
SCHEDULE : IMS scheduler (checking for region avail., doing "Blockmover" resp. pools ,
 PSB check , DB AUTH check a.s.o. , choosing Region by its own scheduler sequence
 queues ; 6 queue types)

LOCATE : ressources inside region / RTE (pgms)

EXECUTE : call analyzing / code

COMMUNICATE (2) : send response

IMS Transaction Flow overview



[The MATRIX data sets are only used in IMS V9 and earlier for IMS-based security; that was discontinued with IMS V10]

MODBLKS data sets are used by IMS when the IMS DRD feature is not in use (introduced by IMS V10) ;

DRD uses the RDDS (Resource Definition DataSet) or the REPOSITORY (since IMSV12).

IMS DB/DC systems can optionally communicate with IRLM and CICS.

For BLDS between IMS Subsystems (inlk. Batch vs. online) then IRLM is mandantory.

First : Starting IMS itself (1 of 2)

- /S ... PROC / ASID (as started task) - Or via JCL with JOB card
- Specify type of (Re)Start (IMS master terminal or z/OS console):
 - Input only permitted if AUTO=N parameter is used on start command
 - Otherwise, IMS will select either Warm or Emergency restart as appropriate
- ✓ COLD Start (/NRE CHKPT 0 [FORMAT ALL,...])
- ✓ WARM Start (/NRE [FORMAT ALL,...])
 - The *FORMAT* keyword parameter is rarely used on Warm starts
- ✓ EMERGENCY Restart (/ERE [FORMAT ALL,...])
 - The *FORMAT* parameter is one of numerous available for emergency restarts
 - See here:
- [/ERESTART command - IBM Documentation](#)

Different: JOB vs. STC ...as any JOB goes thru initiator (READER) , more TCBs, different handling task / abend handling

The IMS Startup parameter AUTO= is usually set to AUTO=Y. In this typical case, IMS will perform either a Warm start or an Emergency (also known as an ERE) restart as appropriate based on the last IMS shutdown/termination. In order to perform an IMS Cold Start AUTO=N must be specified then followed by the IMS Cold Start command. ERE: (<https://www.ibm.com/docs/en/ims/15.5.0?topic=commands-erestart-command>)

... Starting IMS (2 of 2)

... Then this will happen:

- IMS starts DLISAS and DBRC automatically based on PROC names (specified in parameters of the startup procedure in use)
- IMS initializes modules and storage (code, ctlblks and pools)
 - If Parameter RES=Y is specified, *RESident* PSBs and DMBs are also loaded
- Depending on MAXREGN= parameter in SYSGEN or PST= parameter in startup JCL (or parameter member) the blank PSTs are established
 - Up to MAXPST= parameter in execution JCL additional PSTs can be established dynamically and will be released at end of dependent region
- A first System Checkpoint is written to log
- IMS is *almost* ready to process work (input)
 - We still need to start dependent regions and connections to VTAM and IMS Connect (or other subsidiary ASIDs as ODBM,...)

PSTs - (Partition Specification Table); represents an MPP (or DB Thread)

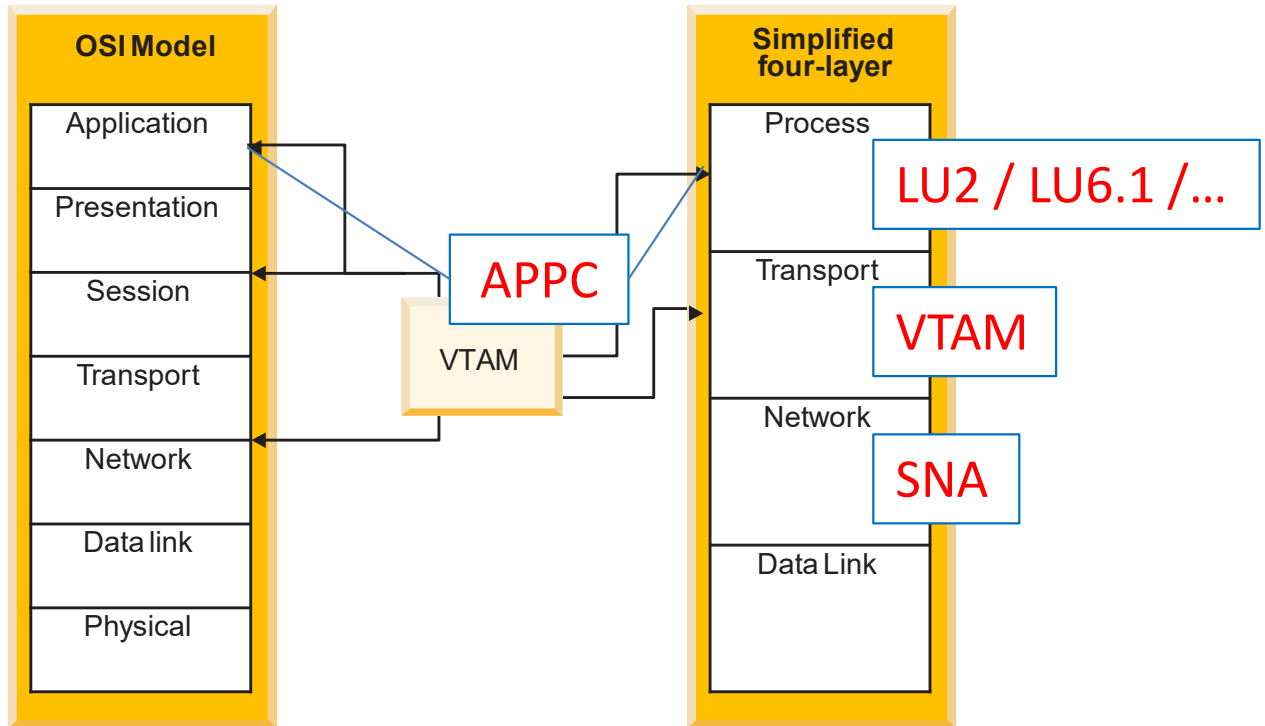
Discussion about APPL Code runtime env. ; if outside IMS Regions , then ... ? (DB2 StoredProcs via ODBA , CICS via DRA threads ... or ODBM , ...)

If distributed , then via IMS/CONNECT (the IP gateway for IMS)
and regarding ODBA : ODBM and CICS – only IMSDB (threads) ,

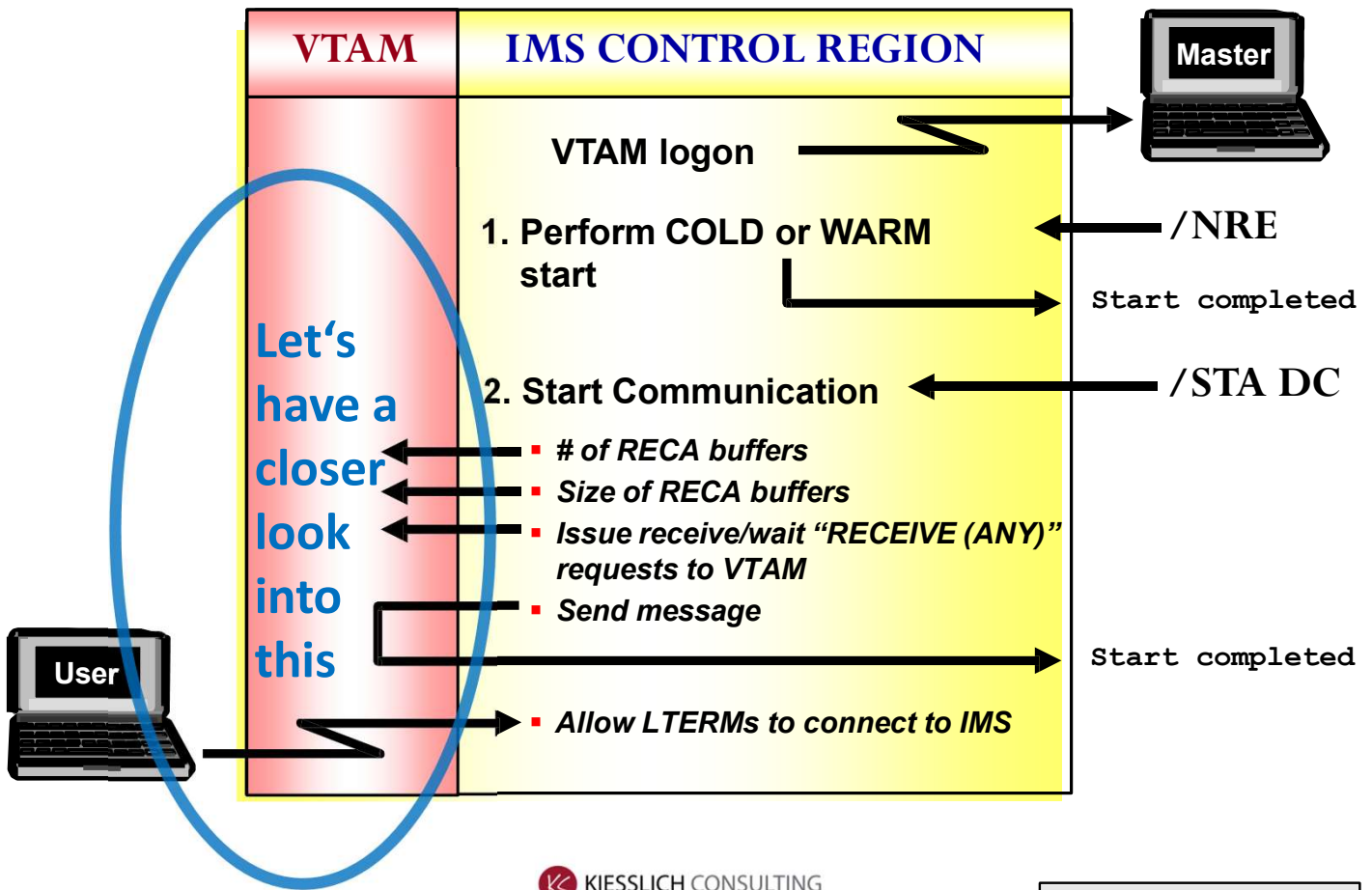
TRAN triggered (OTMA) : IMSDC and dep. Regions (MPPs) needed

IMS VTAM

correlated to OSI



Start of communication [VTAM]



After the "COLD / WARM / EMERGENCY (RE)START COMPLETED" message (DFS994I) the Data Communication component has to be started to enable the user to connect to IMS with VTAM [`/STA DC` , `/STA OTMA`]

(By contrast the `/START OTMA` cmd is only necessary if `OTMA=N` had been specified for the initialization.

Once OTMA is started then the communication between IMS and IMS Connect or other Clients (MQ, TMRA, WAS, RYO) is enabled.

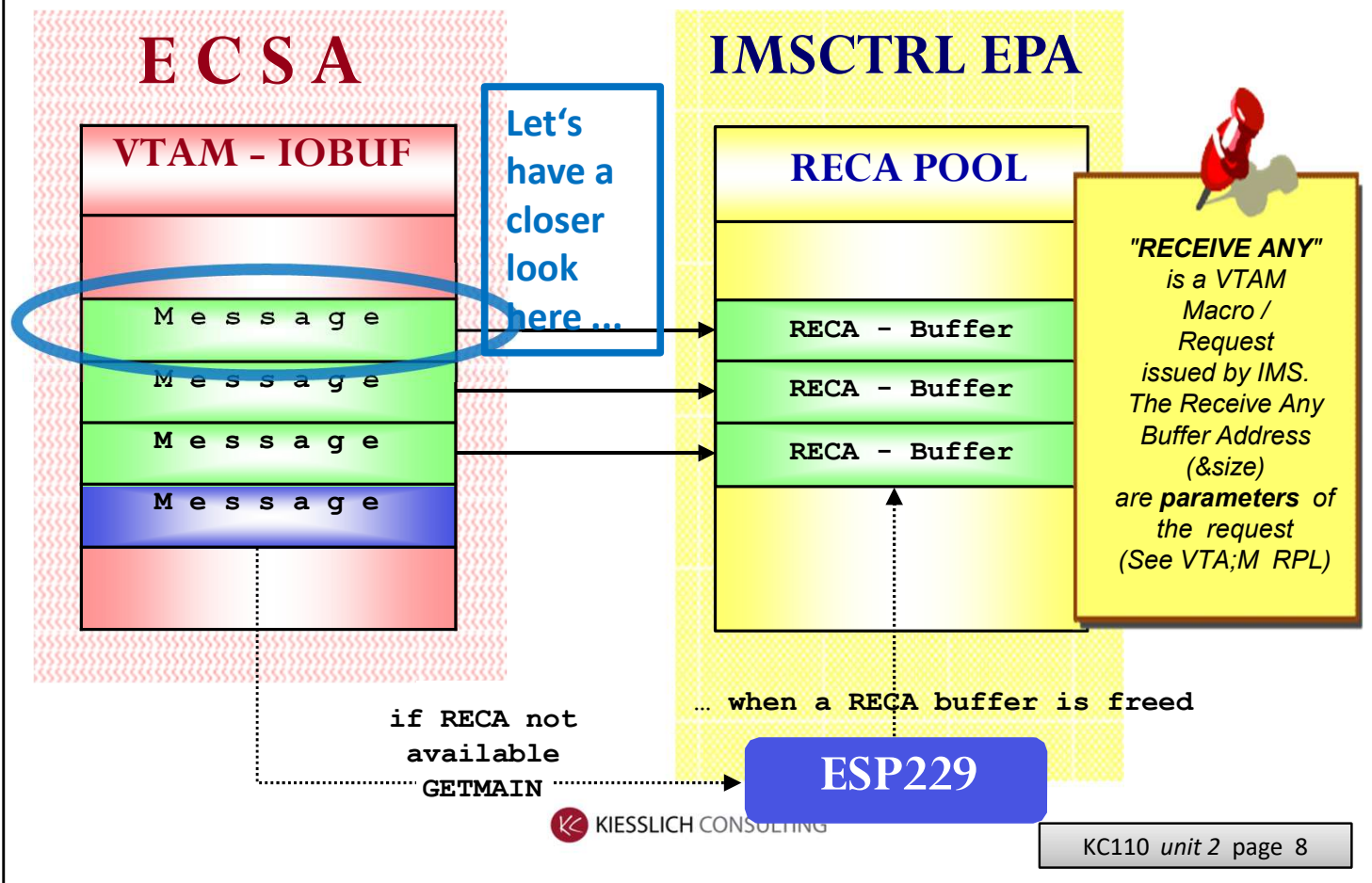
`/STA LINK / PLINK` a.o. -> MSC: no VTAM bfr used

`/STA USER/SUBPOOL xxx ID yyy` -> ISC : (LU 6.1) VTAM !!

`/STA APPC` (a.o. MVS cmds) -> LU6.2 : no VTAM bfr used (p-t-p)

Input Message handling [VTAM]

RecAny Buffer (1 of 2)



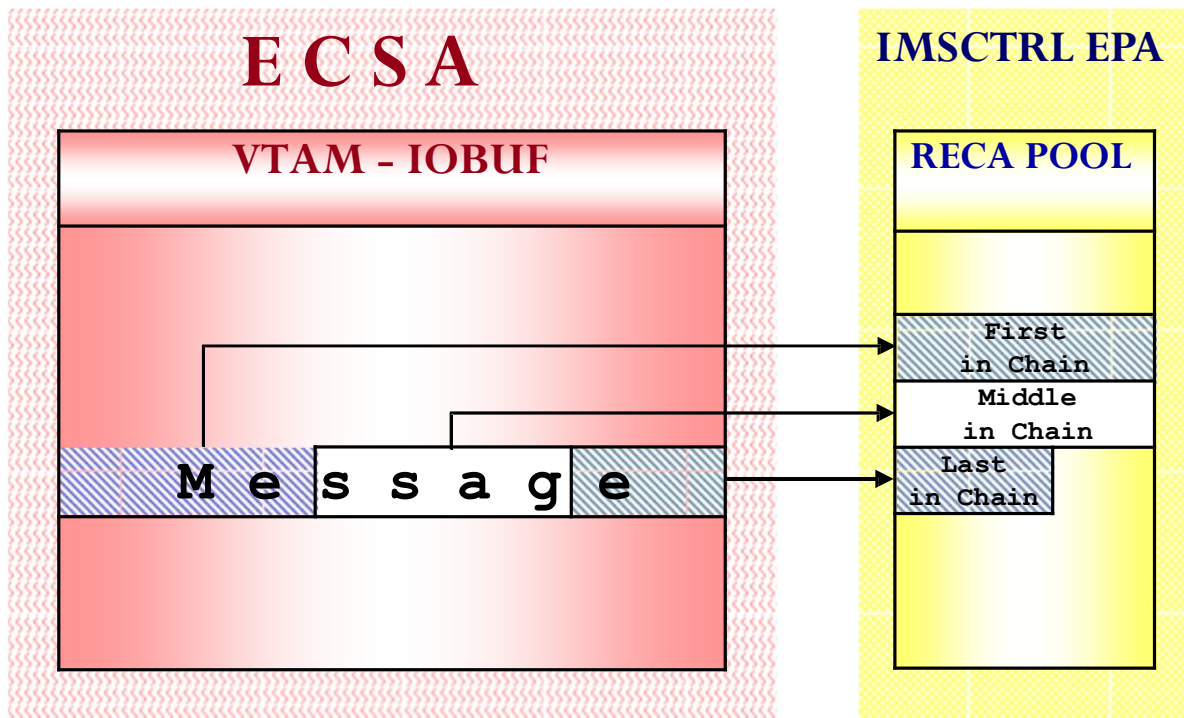
RecAny – “Receive any”

Notes:

Each receive any buffer is represented by an IMS 'RECEIVE ANY ITASK' which needs 1.5K of control blocks. The Receive Any buffer pool resides in EPA of the control region. If there are not enough Receive Any buffers specified in the IMS SYSGEN, VTAM allocates an ESP229 (Extended Subpool) in the IMS CTRL region and loads the message into it to free the VTAM - IOBUF. 10 to 100 RecAny bfrs are common in IMS systems. [1...500]

Input Message handling [VTAM]*

RecAny Buffer (2 of 2)



* OTMA Input Msg handling follows later



KC110 unit 2 page 9

If the Receive Any buffer is too small, chaining takes place. Chaining is a VTAM function. The size of the RECANY Buffer must be equal to or greater than the 'SECONDARY LU - Size' defined in the Request Unit Size (RU Size) of the VTAM MODETAB. (see z/OS VTAM INTERFACE).

For some good recommendations and how to figure out your appropriate specification please see under RECANY= description (size) here:

[COMM macro - IBM Documentation](https://www.ibm.com/docs/en/ims/15.5.0?topic=environments-comm-macro)

(<https://www.ibm.com/docs/en/ims/15.5.0?topic=environments-comm-macro>)

To get all (3270) messages into *one* RECAAny Buffer as 'Only In Chain', you should specify a size of 3892 bytes (control characters included , in terms of 48 x 80 screens plus control chars). Like this : RECANY=(10,3892)

IMS Message types

- The IMS **"Communication Analyzer"** classifies messages as:

TRANSACTIONS:

TRANcode	text
1	8
9	
	n

- **Originate from terminal .. Or origin source (TPIPE, OTMA YTIB, APPC TIB) - most IMS/TM activity**
- **Can be originated from another application via *program-switch (PTP)***

MESSAGE-SWITCHES:

LTERMname	text
1	8 9 n

- **Allows terminals to communicate with each other**

COMMANDS:

/	COMMAND-VERB	text
1	2	n

- **Mostly entered by Automation or *Master Terminal***
- **Commands can be entered from user terminals**
- **Also from (authorized) APPLICATIONs (AOI)**



„Destinations“ .. IMS destinations are always 8 Bytes in length ;

For IMS (Type1) Commands , pls. Check LOGTYPE x'02' (could be suppressed)

IMS Message type examples

- Transactions:
 - Example 1:
 - User input
 - SALARY 123456
 - IMS Transaction Output
 - "Salary for John Doe (123456) is \$xxx.xx"
 - Example 2:
 - User input
 - ADDINV PART 34567
 - IMS Transaction Output
 - "Inventory for red wagons = 1492 units"
- Message Switch:
 - User input
 - SALSTERM SALES DEPARTMENT MEETING AT 3:30 PM
 - IMS TERMINAL Output at terminal ACTGTERM (NODE with LTERM called SALSTERM by name)
 - SALES DEPARTMENT MEETING AT 3:30 PM
- Command examples:
 - /CHE FREEZE
 - /STA REGION IMSMSGx

IMS Message destinations

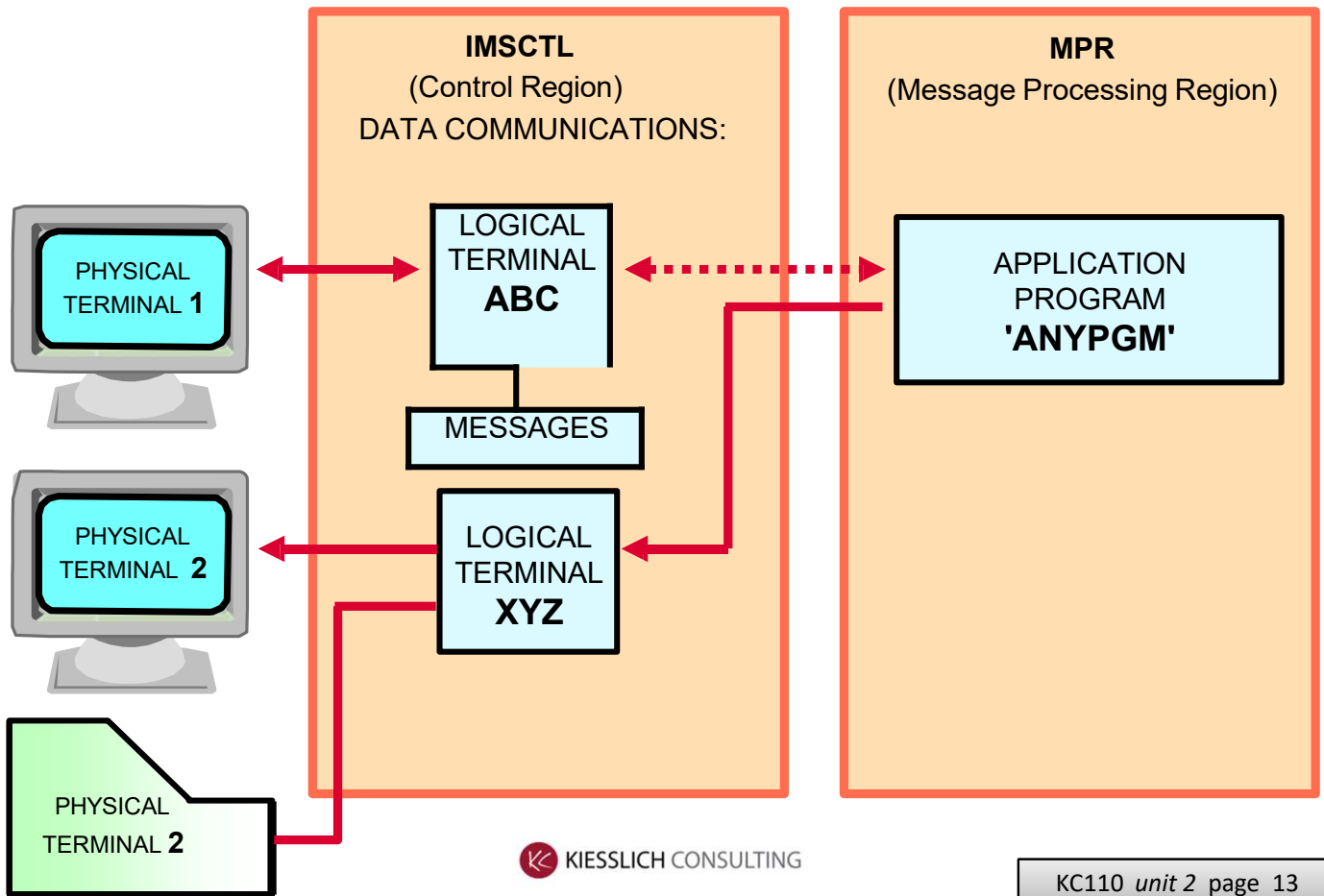
- Transaction-type and Message Switch-type transactions:
 - Processed by the IMS Queue Manager (component of IMS) and placed onto the IMS message queue
 - Messages and therefore IMS message queues have two types of destinations:
 - **SMB** (Scheduler Message Block) destinations are the Queue Manager's representation of messages associated with a transaction code
 - **LTERM** (Logical TERMinal) destinations - more details later
 - IMS SYSGEN coding will determine whether a destination is an LTERM or an SMB
 - For example, "PAYROLL" could have been defined as an LTERM in IMSA and a Transaction (SMB) in IMSB
- Output messages from Application processing programs are also queued to LTERM destinations

[No IMS commands are placed into the message queues; commands are not considered as IMS Msgs → special command handler task]

Some commands require storage space in the WKAP as part being processed

PAYROLL example: IMSA and IMSB are not connected or running in one IMSPLEX !

IMS Terminal concept [VTAM]



MSG Q:

LTERMs as input source , destinations are the queue elements (TRAN for input, LTERM for response / output)

In case of PGM-2-PGM switch, the output destination is the next TRAN

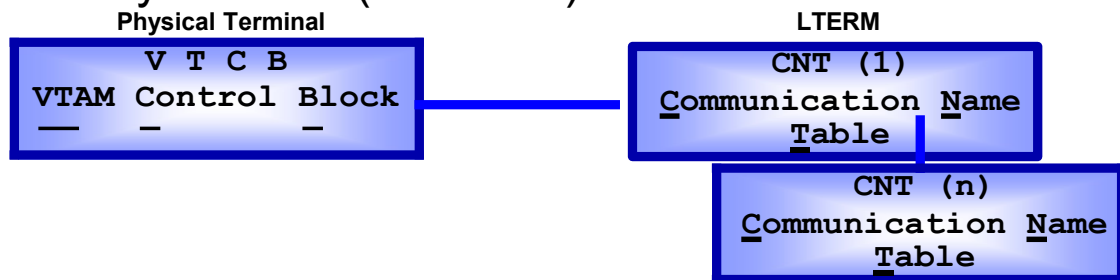
APL PGM:

- works with this always first PCB from PSB - the special „IOPCB“ representing input (LTERM is a „subparm“) and response

Logical Terminal: could be also TPIPE (from MQ Bridge Adapter via OTMA a.o.)

IMS Terminal related CBs – Static Defined [VTAM]

Statically Defined (IMSGEN) Devices



- There is one VTCB for each VTAM Terminal (*NODE*):
 - The VTCB is actually a collection of control blocks as : CLB, CTB, CRB, CCB and CIB
 - The name is specified in the IMSGEN *TERMINAL* macro
- The CNT control block represents an IMS LTERM:
 - The name of an LTERM/CNT is specified through the IMSGEN *NAME* macro
 - Multiple CNTs can be associated with each VTCB (CLB)
- The relationship between Physical terminal and LTERM can be changed by the /ASSIGN IMS Operator command

Control blocks

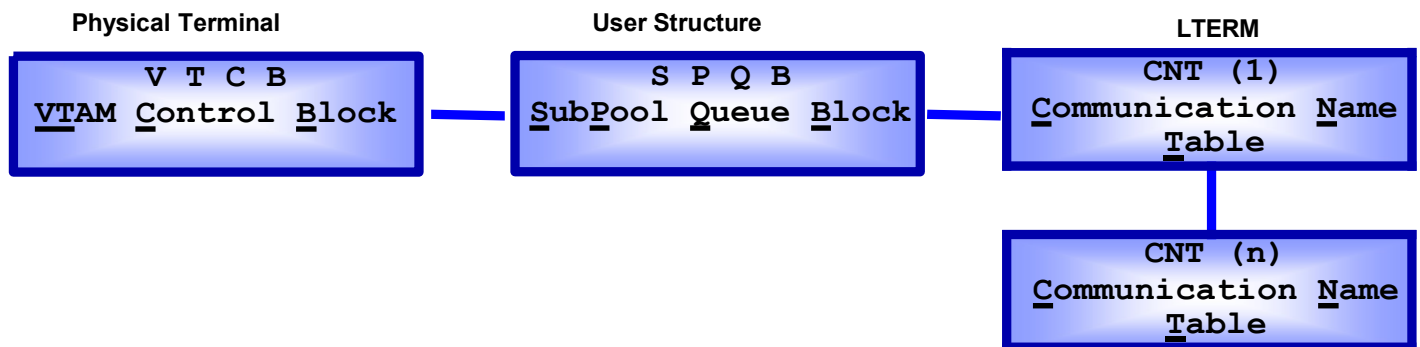
See also IMS Debugging class CM660x , KC660G

See the IMS Diagnostic Guide and Reference manual for the "Table of Control Block Definitions" to learn control block definitions and mapping macros.

Note, that VTCB spans all CBs belonging to the represented device (VTAM LU)

IMS Terminal related CBs: Dynamic (1 of 2) [VTAM]

Dynamically Defined (ETO) Devices



- The purpose of dynamic VTCBs and LTERMs is the same as for static devices
- The *User Structure* (SPQB) is an added control block type that only exists for ETO devices
 - The SPQB enables additional function such as the ability to associate multiple LTERMs to a signed-on user

USER CB vs. SPQB – same CB structure (by layout) !!

Not „only“ for ETO Devices ... SPQB needed also for LU 6.1 / ISC

IMS Terminal related CBs: Dynamic (2 of 2) [VTAM]

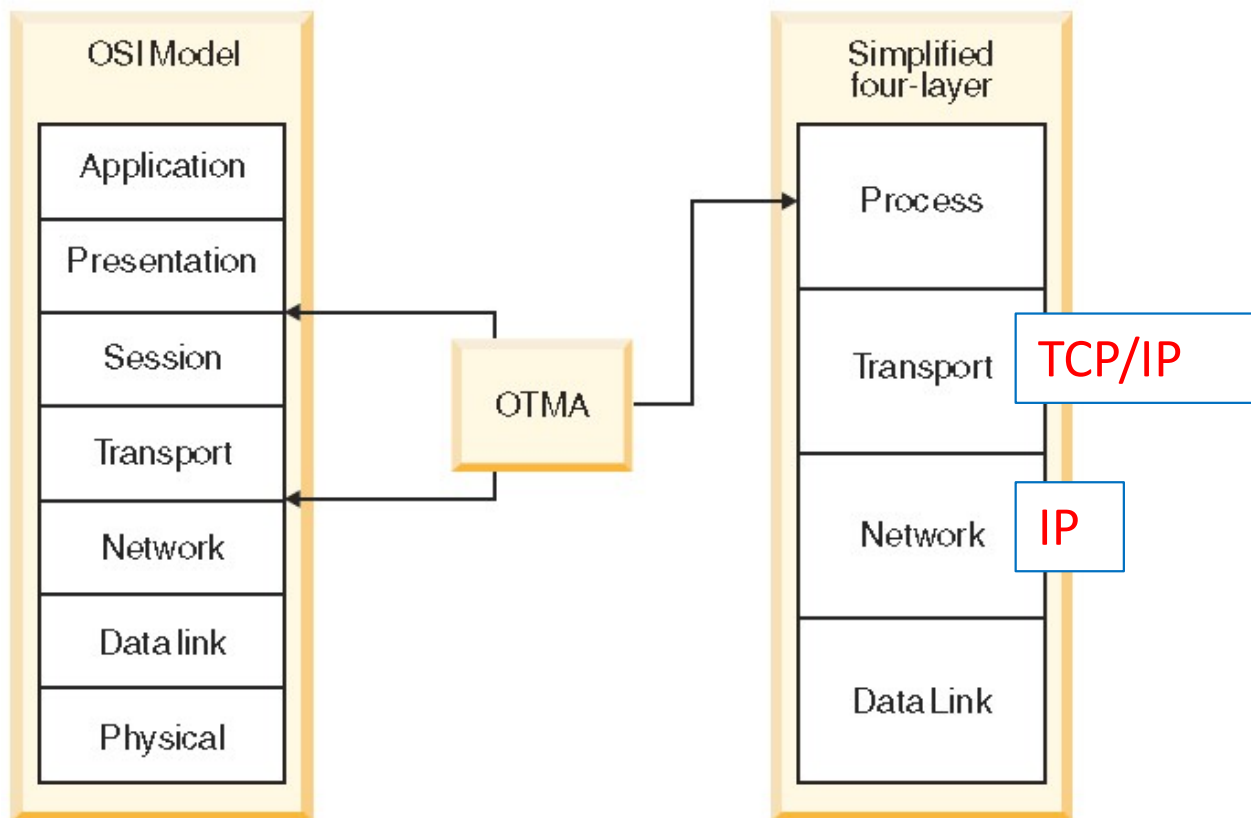
- When ETO : These three types of structures are created only when needed:
 - At user logon or signon time
 - When application output is created and queued
 - When these control blocks were deleted before (deleted when no longer needed -> timeout)
- The names used for these structures depend on installation rules defined through the various types of *ETO Descriptors* (that is, logon, user, device, and MSC) *models* and user exits

Some similar techniques for OTMA terminals, f.i. defining descriptors (, EXITs, ...)

Timeout: see ALOT / ASOT

IMS/TM specifics: LOGON + SIGNON !!

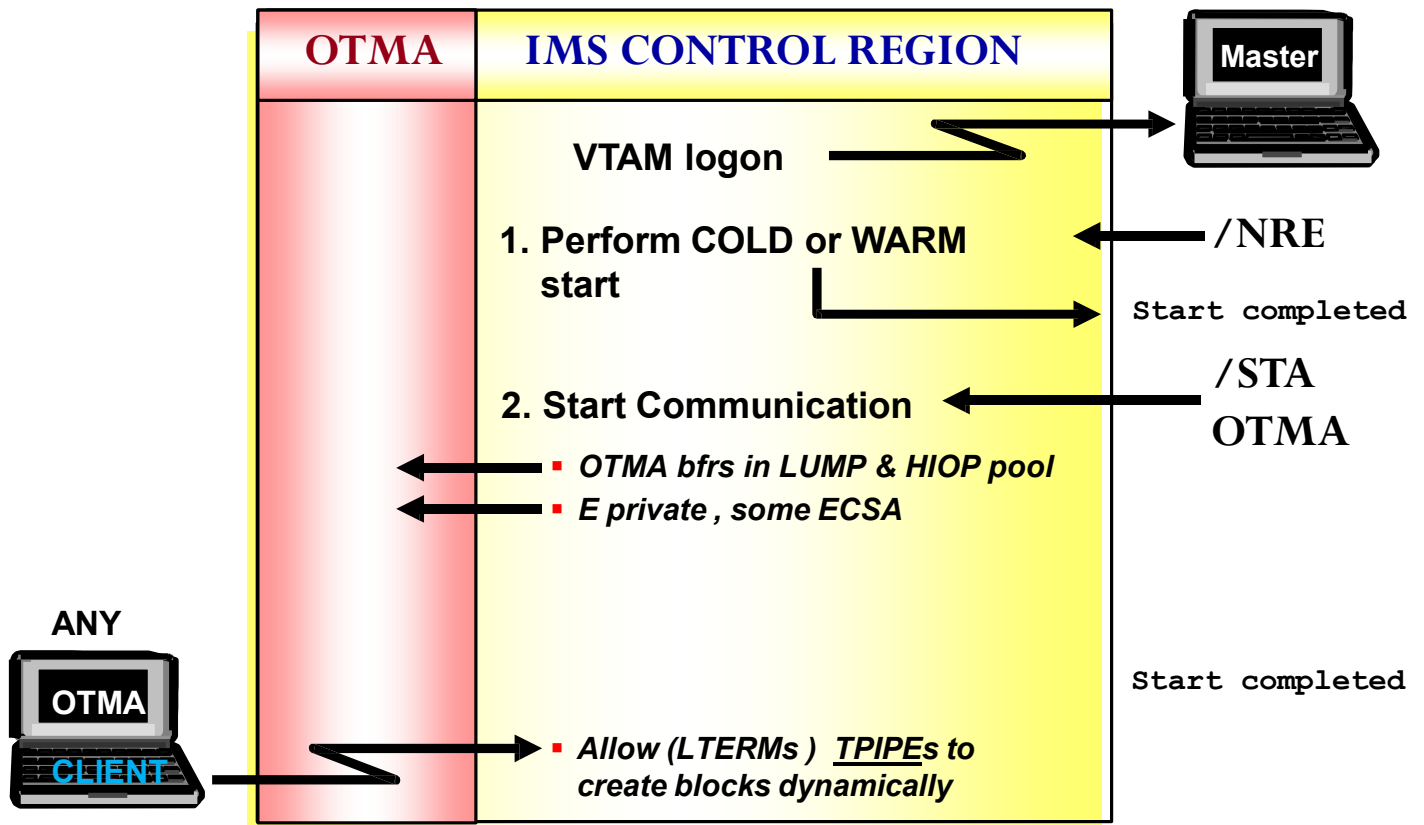
IMS OTMA correlated to OSI



OTMA - OPEN TRANSACTION MANAGMENT ACCESS

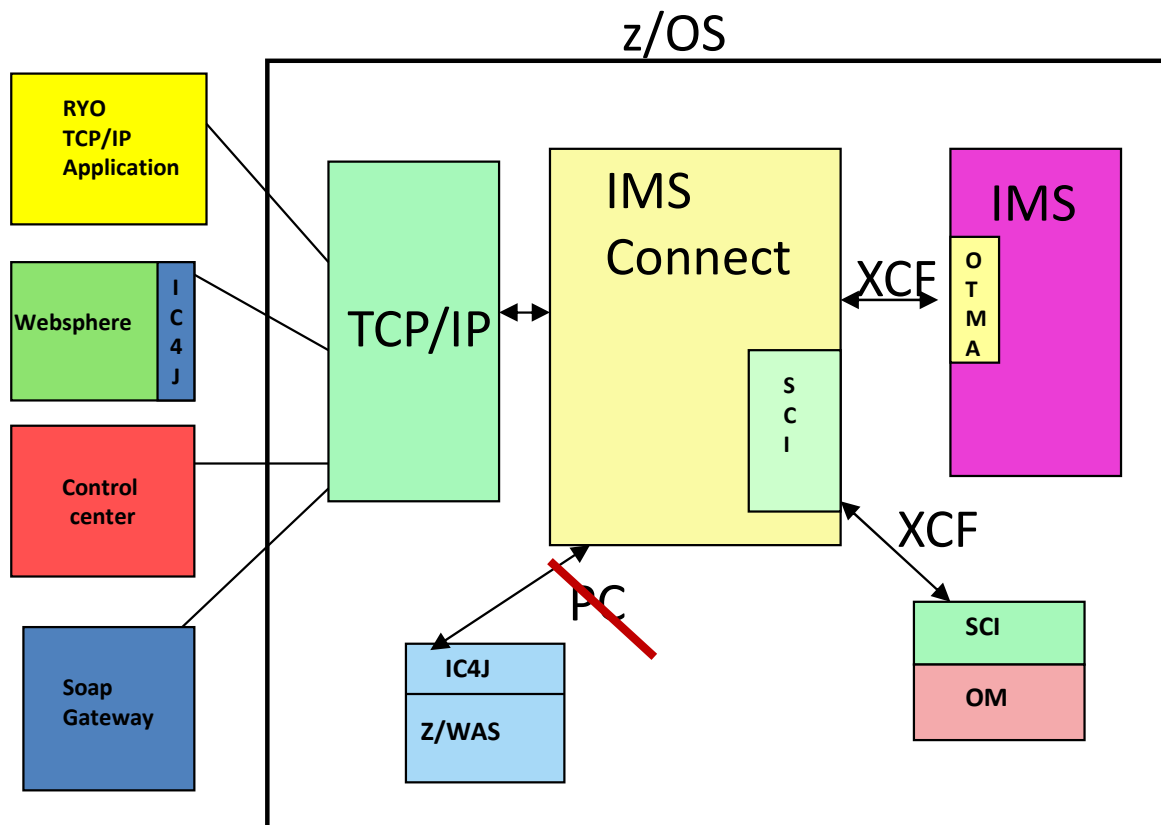
OSI – Open Systems Interconnection

Start of communication [OTMA]



OTMA=Y (or /STA OTMA) enables communication between IMS and:
 IMS Connect, the IMS specific IP – Gateway managing socket connections under IP
 Port(s)
 or other Clients as W/MQ, or RYO OTMA clients (OTMA C/I is documented)

IMS Connect – An OTMA CLIENT



PC: means cross memory is used (program call) ! - requires running on same LPAR

IMS/Connect as IP Gateway (compare to CICS TS and its CTG)

z/WAS .. Or distrib. WAS :

TMRA :

OTMA Message Structure

← PREFIX →

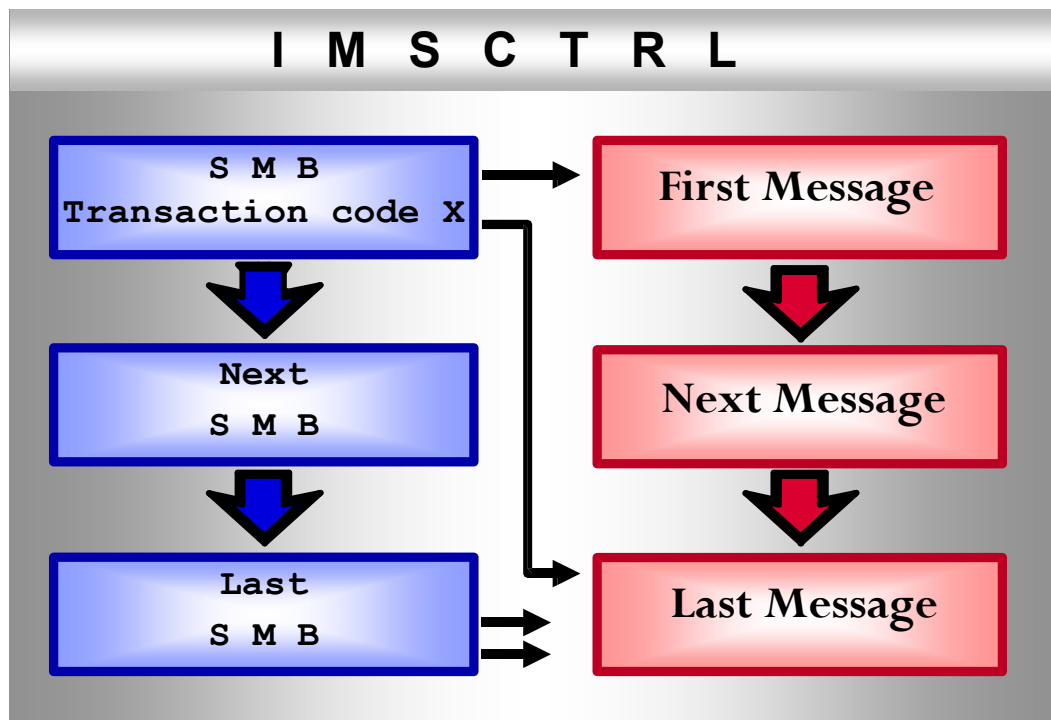
Message Control Information	State Data	Security Data	User Data	Application Data
-----------------------------	------------	---------------	-----------	------------------

Tpipe names
Message type
Sequence Numbers
Processing Flag
Response Indicator
Chaining Indicator

Destination
Override
Map name
Sync Flags
Sync_Level
Commit Mode
Tokens
Server State

User ID
Utoken
Security Flags

MSGQ: Input Message queuing (local)



All messages destined for a given transaction code are queued in a serial chain generally based on the time of receipt.

The first and last messages for a transaction are pointed to by that transaction's control block (SMB = SCHEDULER MESSAGE BLOCK), the IMS Scheduler's representation of transaction codes.

Input Queue: For each class specified during IMS definition (MAXCLASS), a TCT (Transaction Class Table) is generated. (255 since Rel.4.1)

The TCTs are grouped together in the TCT Pool (Part of the TAB – Transaction Anchor Block). When there is a message on the Input Q for this class, the appropriate priority slot will point to an SMB. There are 16 priority slots in each TCT, numbered from 0-15. Prty 0 is for batch type messages, Prty 15 is for messages that have been put back to the Input-Q following deadlock situation.

The TCT also holds a pointer to the "Highest Priority SMB" which is taken when the scheduler selects a message from this class to be scheduled. During IMS definition a specified Transaction Code will generate an SMB. The SMB points, if it has messages to the Input Q, to the first and last message. The first is the one to be dequeued when the scheduler wants a message, the last pointer is used to enqueue a new, incoming message. Since these pointers point into one of the two Message Q Datasets, they are DRRN pointers. "D" stand for device, i.e. 'D' value selects whether a message is on the short, long or QBLKS dataset.

"RRN" identifies a relative record number on that dataset.

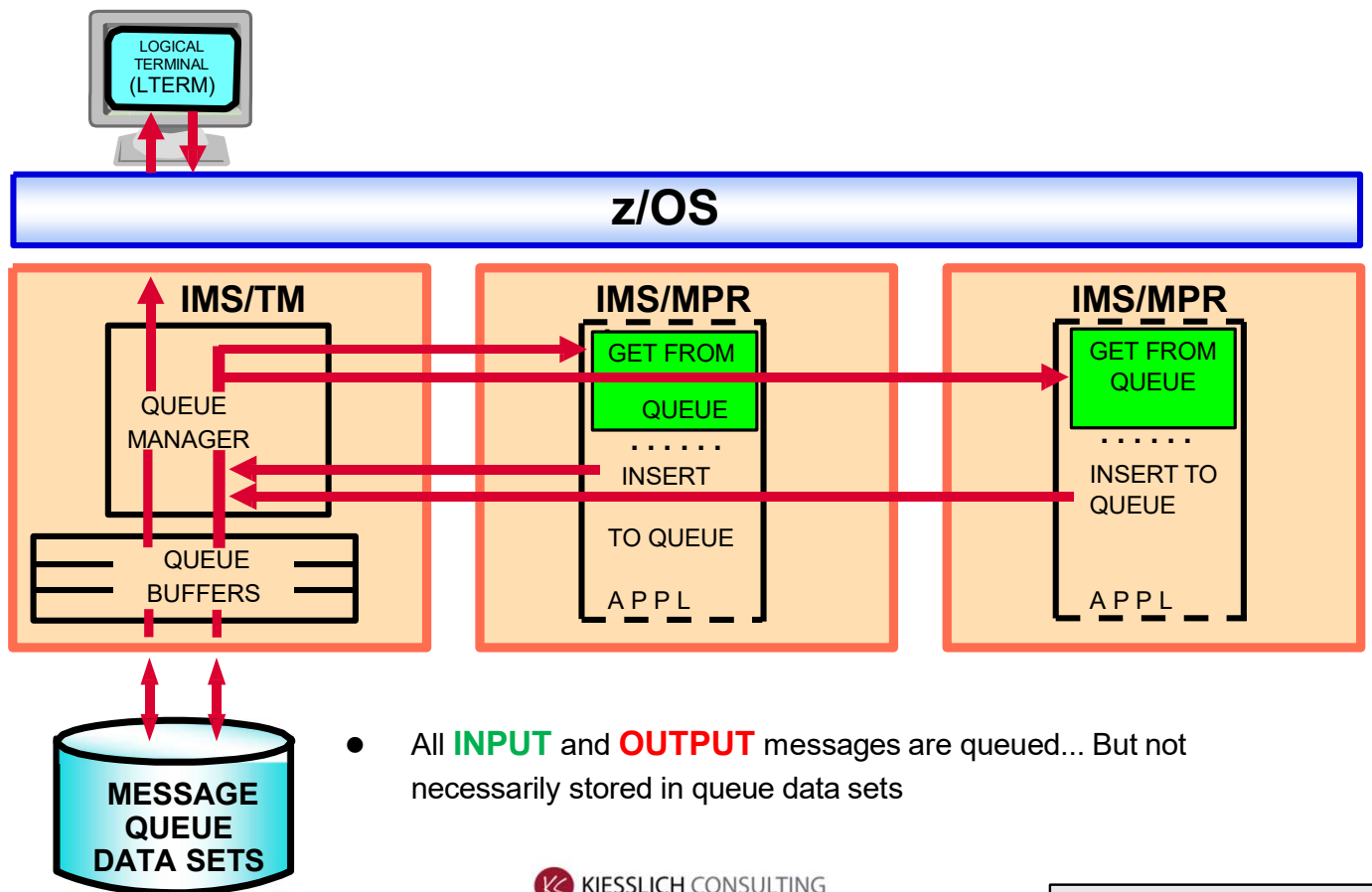
When the scheduler wants to select a message from the queue (Q Datasets), it takes that DRRN-pointer and checks whether the message is in the Q buffer or not. If not, Message Q Management will fetch it from the appropriate dataset.

The pointer with DRRNs has one big advantage. It allows to point to a message without knowing whether it is in a message-Q buffer or out on the message-Q dataset yet. If there are more messages for an SMB on the Input Q, one message chains to the other via its Message Prefix. The message prefix has the layout of a 01 type Log-Record (for Input) or a 03 type Log-Record (for Output). Pointing from one message to the other is again done via a DRRN-Pointer. When a message consists of multiple segments, the first segment's prefix points, with a DRRN-Pointer, to the second segment. The second and all subsequent segments have a so called short prefix (See DSECT of log record 01, 03).

About Scheduler and its queues , see chapter 3.

Within Shared Queues environment: slightly different then !

Message queue: **Input** message (1 of 2)



Message queue: **Input** message (2 of 2)

1. **INPUT TRANSACTION Messages** are QUEUED by **SMB** or if message switch, LTERM (not shown)

TRANX

MSG1 (LTERMA) TRANX

MSG3 (LTERMX) TRANX

MSG4 (LTERMQ) TRANX

MSG7 (LTERMX) TRANX

TRANY

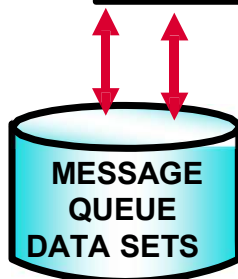
MSG2 (LTERMB) TRANY

MSG5 (LTERMZ) TRANY

MSG6 (LTERMA) TRANY

MESSAGE-QUEUE BUFFERS (Control Region)

MSG1	RPL3	RPL2	MSG5	MSG7
RPL1	MSG6			
	MSG2		MSG4	MSG3



2. OUTPUT MESSAGES are QUEUED by **LTERM**

LTERMA

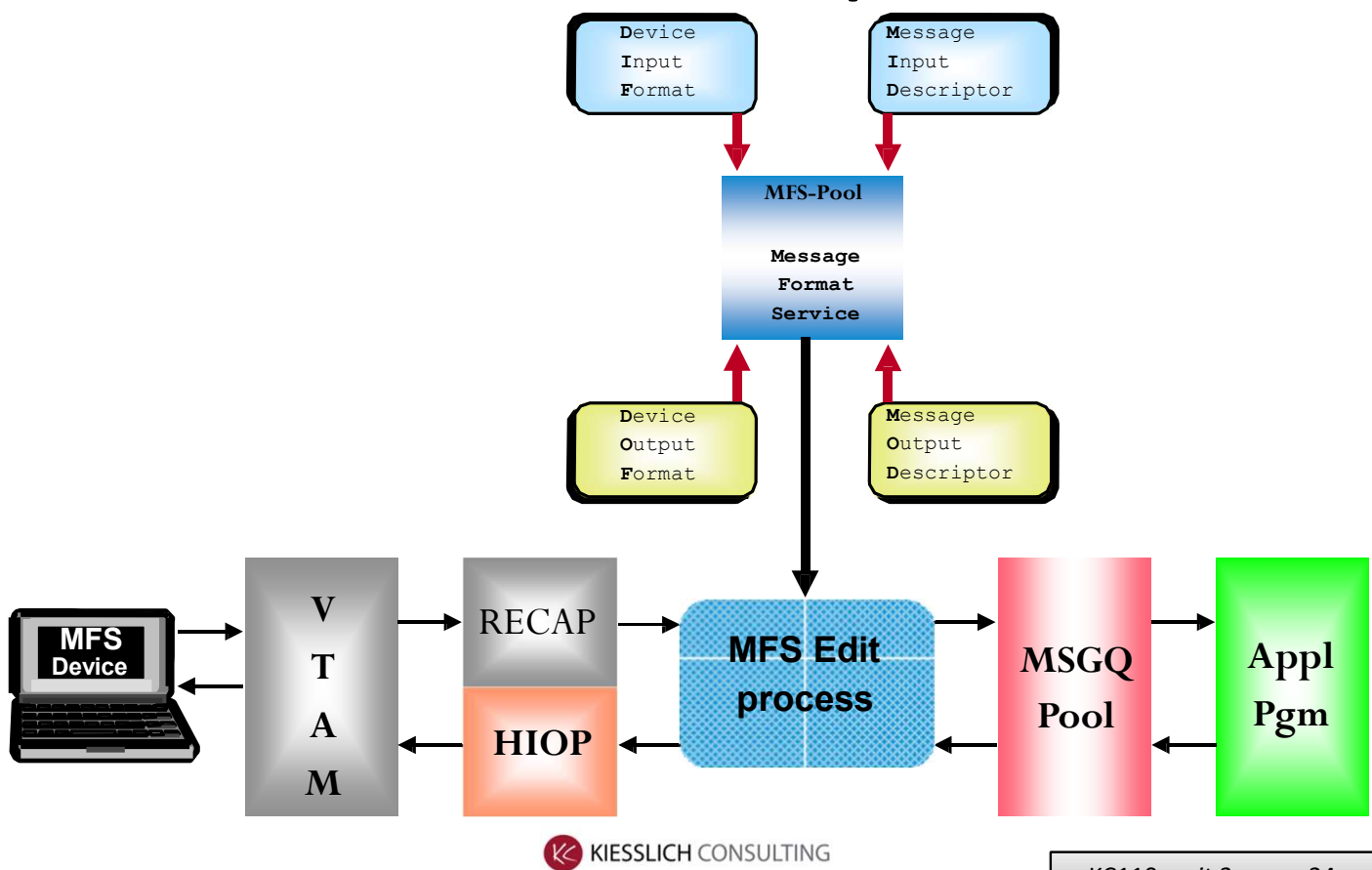
RPL1 (TRANX)

RPL3 (TRANX)

LTERMB

RPL2 (TRANY)

Message Format Service (MFS) block relationship



KISSLICH CONSULTING

KC110 unit 2 page 24

MFS importance is shrinking a bit compared to more input thru OTMA resp. TCP/IP traffic growth,

MFS Device means : VTAM connected ..translation possible to distrib. Client presentation layer !

OTHER Notes:

MFS translates data streams to segments and vice versa, by reference to format blocks. These describe:

The device format, that is, the positions of fields on the screen

The message format, that is, the positions of fields within segments as expected by the application program.

MFS formatting is only applicable to messages from traditional green screen devices or PC's running 3270 emulators:

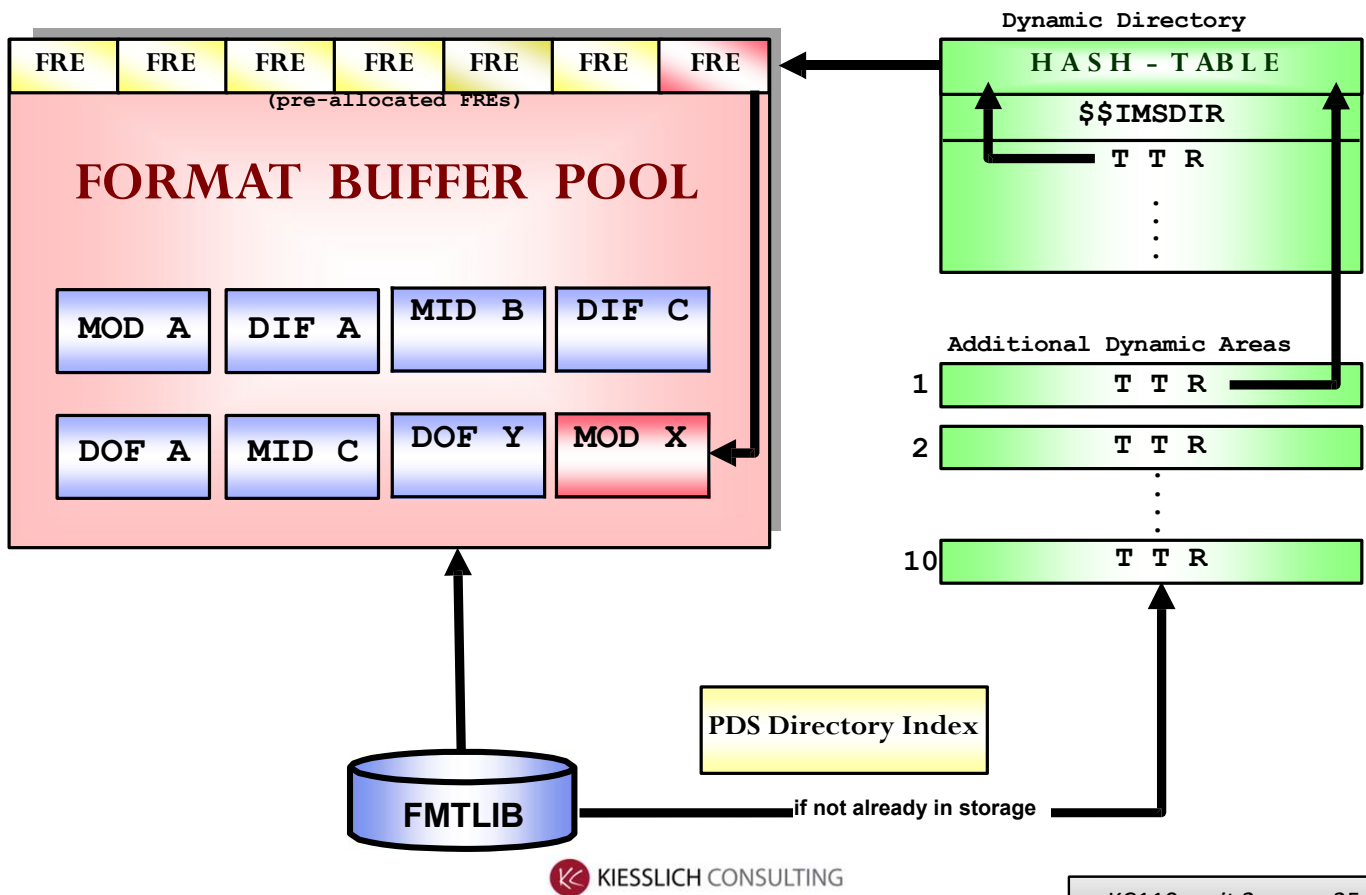
MFS formatting is ***NOT*** applicable (not performed) for messages received from (or sent to) APPC (LU 6.2) or OTMA devices

MFS POOLS in debugging class only !

Next foils off !!!

Similar function to BMS in CICS

Message Format Pool (1 of 2)



Notes:

The Messages Format Pool (also called the MFS Pool) is used by MFS to store MFS blocks while they are being used as templates during MFS editing of input and output messages. FORMATA and FORMATB are PDS data sets that contain MFS blocks; like all PDS data sets, access to a member usually requires multiple I/O operations: one to the directory portion and (at least) another to the data portion to retrieve the required block.

As we will see over the next few pages, IMS maintains an ever-growing Dynamic Directory ... that is intended to eliminate repeated access (I/Os) to the FORMAT data set's directory.

Message Format Pool (2 of 2)

- Standard Format Fetch:
 - At IMS initialization time, IMS loads the FORMAT library directory block addresses and the name of the last format block in each directory block
 - If a format block is requested and not already in storage, IMS has only to search the corresponding directory block to get the address TTR (Track-Track-Record) of the format block
 - The TTRs are stored in the DYNAMIC AREAS in 10% increments
- Format Fetch without directory search:
 - To avoid directory block searches IMS can use the '\$\$IMSDIR' member which contains the TTRs of the format blocks on Disk
 - This member is produced by running the \$\$IMSDIR Utility
 - The \$\$IMSDIR member will be loaded at IMS initialization into the DYNAMIC DIRECTORY
 - Its use avoids an additional I/O operation for the first access of a block
 - If a format block is requested and not already in storage, IMS loads the format block without additional Directory block searches

For the duration of an IMS execution (or to the next FMTLIB Online Change), MFS will require at most 1 I/O operation to the PDS directory for each block:

At first use, a BLDL is issued and the block's TTR information is added to the Dynamic Directory.

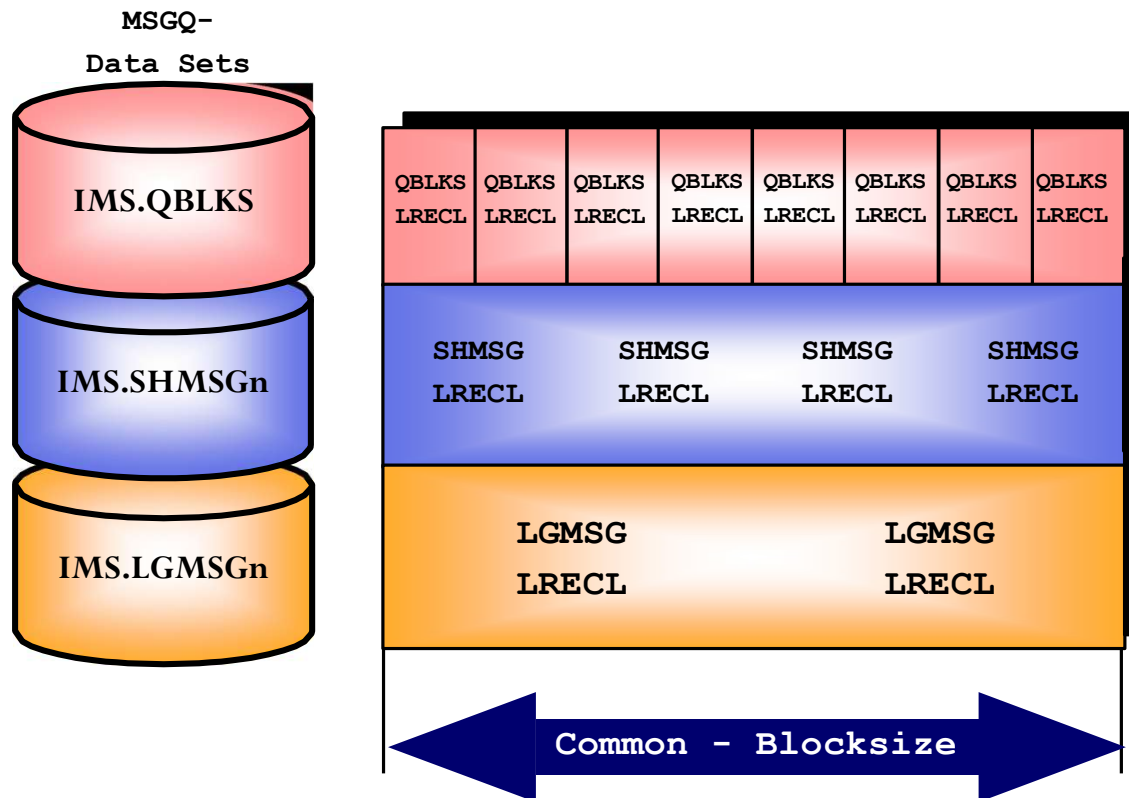
Subsequent access to that block uses Dynamic Directory information and not require additional BLDL I/O.

The use of the \$\$IMSDIR process avoids the need for the BLDL for even the first access to the block:

BLDL information is loaded directly into the Dynamic Directory (from the \$\$IMSDIR Member) at IMS initialization.

The trade-off is the maintenance effort for the support of the \$\$IMSDIR member.

Message Queue datasets and LRECL sizes



Message Queue pool resides in the IMS CTRL EPA and contains:

- Input message segments for scheduling
- SPA segment to be passed, if conversational
- Output segments:
 - To reply to terminal (IOPCB/ALTPCB)
 - To reply to printer (ALTPCB)
 - To another application program (TX Message)
- Output sub-queuing reflected by QBLKS contents

MSGQ Pool Structure and Management (1 of 2)

- Message Queue data sets are used to back-up queue content
 - With Shared Queues, Coupling Facility Structures are used to store contents of IMS Message Queues
- Always formatted at Cold Start - Verified at other (Warm or Emergency) start up:
 - Can optionally be individually formatted during warm or /ERE restarts
 - This formatting is when the DCB information (described below) is established
- Message Queue data sets are organized as three OSAM data sets:
 - QBLKS LRECL is 56 bytes
 - The QBLKS data set is associated with output messages
 - SHMSG LRECL(size1) must be an even multiple of the QBLKS LRECL
 - LGMSG LRECL(size2) must be an even multiple of the SHMSG LRECL
 - LGMSG LRECL must be large enough to contain the largest output segment inserted by any application program (+ at least 64 bytes prefix), or truncation of message segment occurs

Even though messages are almost never actually written to any of the queue data sets, the Queue Manager assigns a specific dataset, track and record to each message segment as part of the Enqueue process.

For APPL. DEVs: your input/output msg max size should be known to the Sysprogs regarding settings for msg buffers...

MSGQ Pool structure and management (2 of 2)

- Common blocksize (MSGQ buffer size) is a multiple of the LGMSG LRECL
 - There is a slight performance benefit to having same value specified for the MSGQ buffer size = LGMSG LRECL (size=size2 below)
- Definition in MSGQUEUE macro of IMS Gen:

`BUFFERS=(nbr,size), RECLNG=(size1,size2)`

 - Startup override parameter values can be specified as well
 - Shared Queue parms only in Startup specifications – NOT in IMS Gen
- Up to 10 SHMSG and LGMSG data sets can be allocated:
 - *Queue Manager Concurrent I/O* added as performance enhancement in order to enables parallel I/O operations between multiple Message Queue Data sets
 - LRECL and BLKSIZE will be the same for all the data sets of a single queue data set type (that is, all SHMSG data sets have same LRECL)
 - Same SPACE allocation **should** be specified for the same data set type

Queue Manager Concurrent I/O

- Up to 10 data sets can be specified for the IMS Short and Long Message data sets:
 - The goal is to reduce or eliminate I/O bottlenecks by spreading message assignment between additional data sets
 - This is only a benefit if systems are experiencing significant I/O delays to Message Queue data sets
 - Option is not available for QBLKS data set
- The Queue Manager assigns the messages in a *Round-Robin* manner to all data sets of the queue data set type:
 - The Queue Manager uses simple modulo/remainder division based on the message number of the message to assign it to a data set
 - The remainder is the data set number (0 - 9), to which the message is assigned.
 - Example: 3 SHMSG data sets, 5 LGMSG data sets – new message number 127 is to be assigned to a LGMSG data set
 - Result: $127/5 = 25$ Remainder 2
 - Message is therefore assigned to 3rd data set (0, 1, 2, 3, 4)



Notes:

This option was introduced at the time of IMS V4, but is no longer as useful as before due to the ability to specify larger queue buffer pools, and thereby reduce or eliminate the need to perform any I/O to message queue data sets.

Message Queue data sets out of space

- All space in a Message Queue data set could be depleted by a looping program (especially a BMP) issuing Insert calls
 - There is an IMS *shutdown* limit parm in the IMS *MSGQUEUE* macro
- If this limit is reached, IMS attempts an orderly shutdown:
 - IMS will issue one message DFS206, DFS207 or DFS208 depending on whether the shutdown limit was reached for the QBLKS, SHMSG or LGMSG message queue data set respectively
 - IMS will then issue an internal '/CHECKPOINT DUMPQ' command and attempt an orderly shutdown
 - IMS might terminate *normally* or might ABEND with a U0758 with the contents of register 14 indicating the full queue data set
- Whether or not an orderly shutdown was successful, the same recovery action is required:
 - Delete then re-allocated the failing Message Queue data set with a larger size - then
 - Restart IMS with command: '/NRE BUILDQ FORMAT XX'
 - Where 'XX' (either QC, SM, or LM) depends on the message queue data set involved.
 - <https://www.ibm.com/docs/en/ims/15.5.0?topic=commands-nrestart-command>

While it is not mandatory that the failing queue dataset be deleted and enlarged prior to restarting IMS, IMS is likely to fail again after restart.

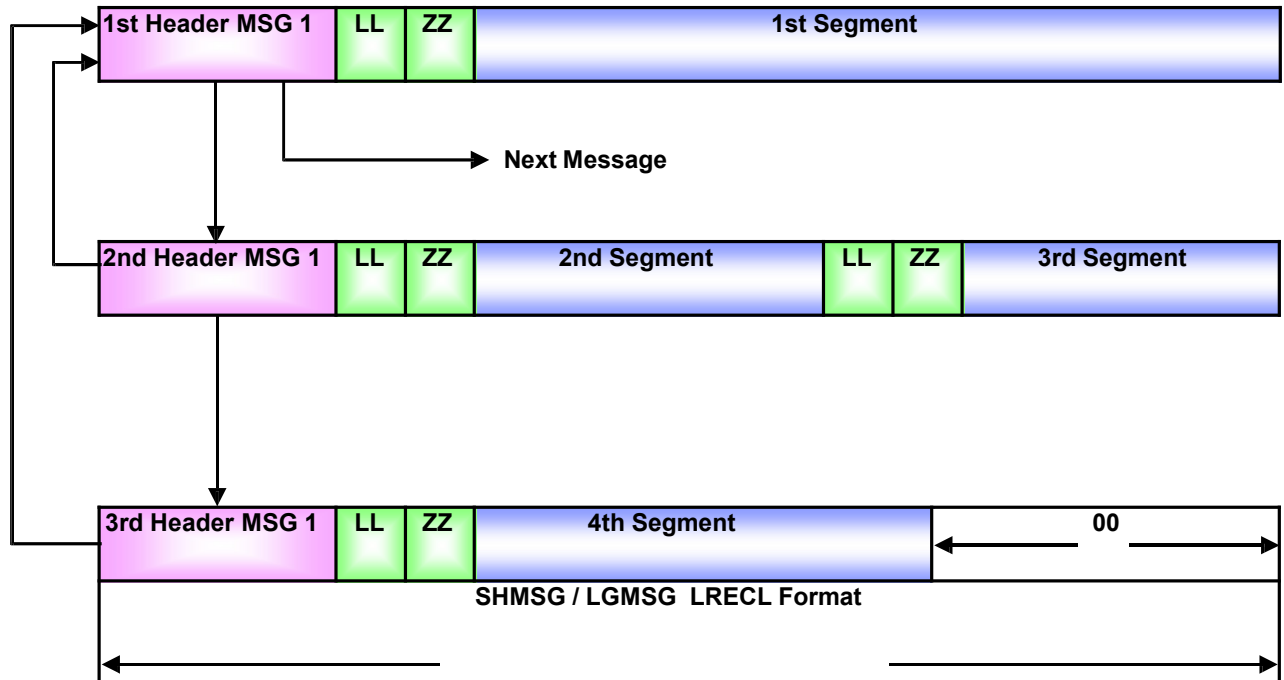
Although not described on this page, an effort should also be made to identify the source of the additional messages that exceeded the capacity of the IMS queue dataset.

[/NRESTART command - IBM Documentation](#)

(<https://www.ibm.com/docs/en/ims/15.5.0?topic=commands-nrestart-command>)

MSGQpool: Multi segment/Spanned Messages

Multisegment Messages



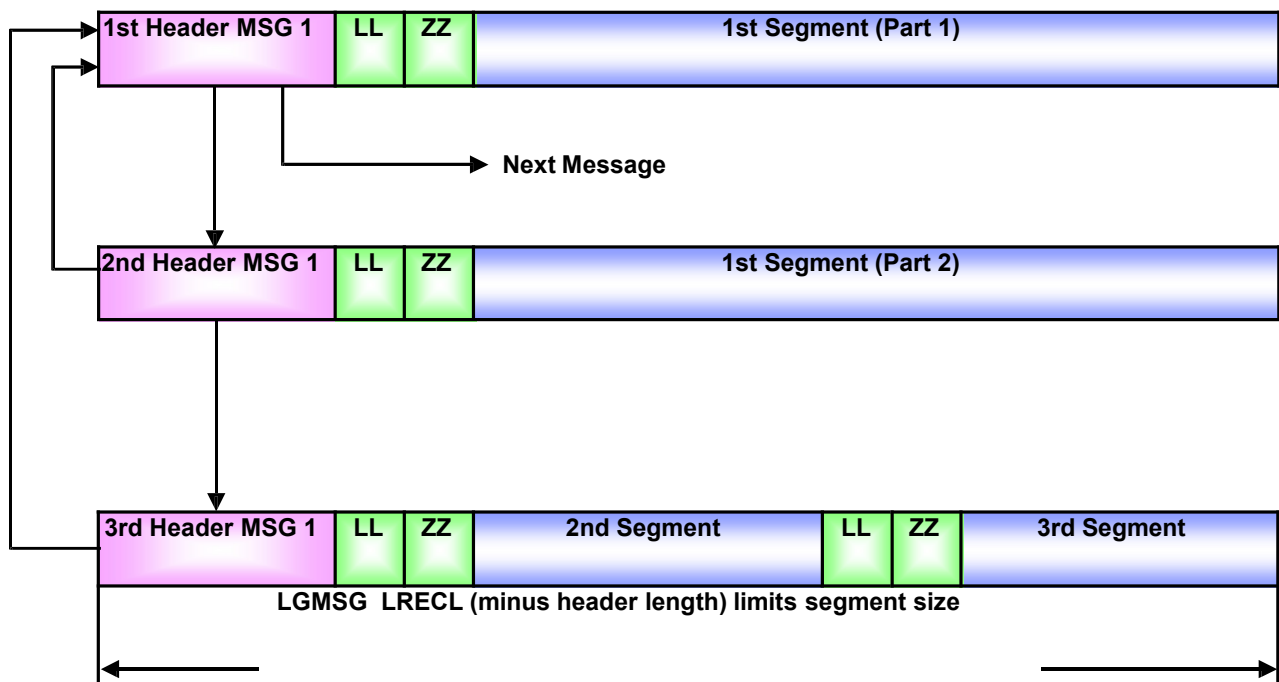
Notes:

A SHMSG/LGMSG LRECL can contain multiple message segments, but only from the same message.

MSGQpool: Spanned Message Segments

Message with Spanned Segment

- Spanned Segment Normally **Not** Allowed



Exceptions (where segment spanning **IS** allowed):

- 1) Non-MFS **output** message segments
- 2) The **SPA** Segment of a Conversation Transaction

A SHMSG/LGMSG LRECL can contain multiple message segments, but only from the same message.

Spanning of message segments beyond the length of the LGMSG LRECL is normally not allowed.

NON-MFS output message segments and Conversational SPAs are the exceptions to this rule.

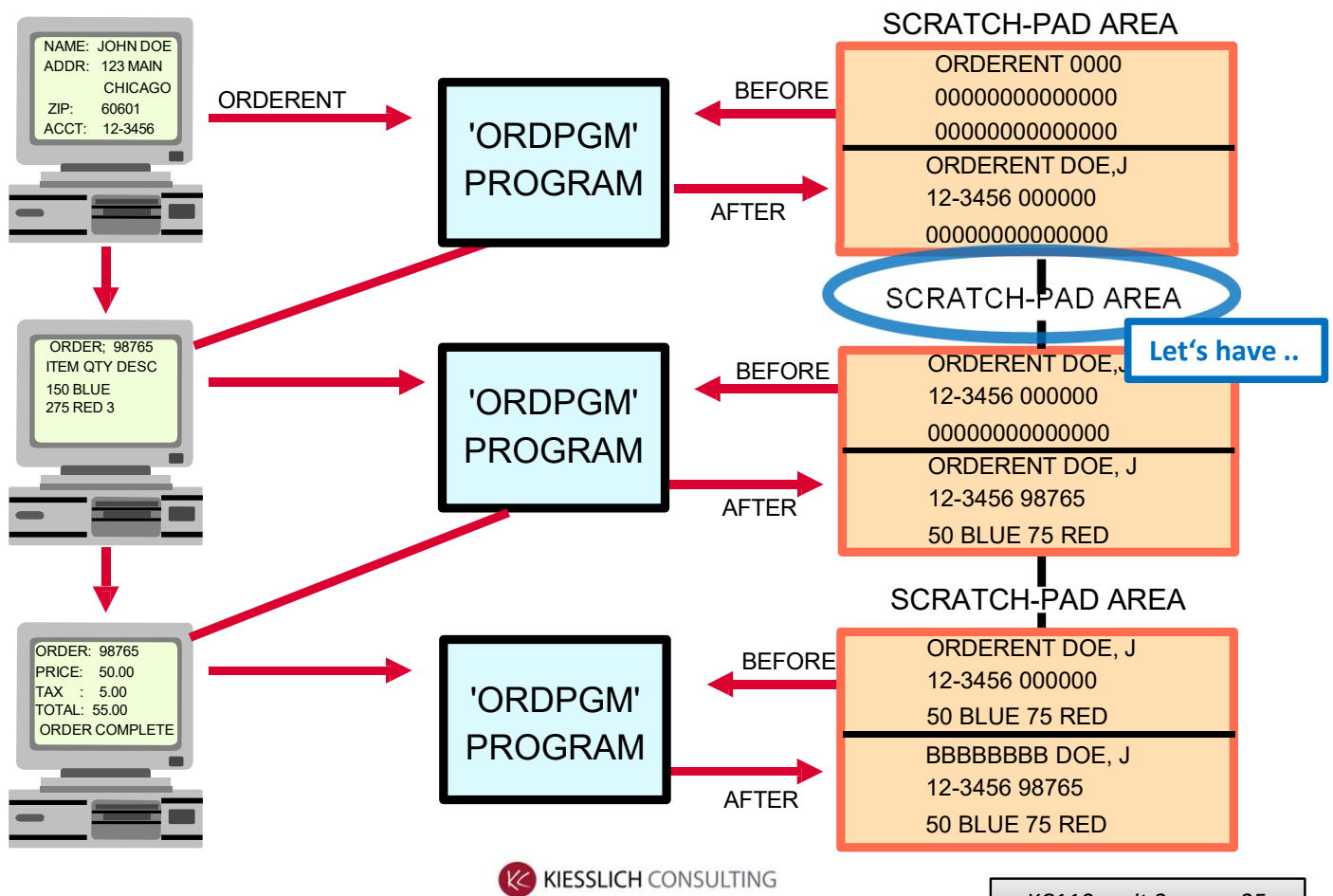
(It's obvious , they don't need the MFS as presentation layer)

Message queue

Logical Record assignment rules

- Multiple segments from the same message **can** be stored in the same logical record
- Segments from different messages **cannot** be stored in the same logical record but can share the same block
- Except for SPA segments, an input segment **cannot span** logical records
- Only SPA segments and non-MFS output Segments **can span** logical records
- Multi-segment messages can use logical records from both the long and short message data sets
- Segments from individual messages are chained together across logical records but not within a logical record

Transaction processing: Conversational



Types :

- Send only
- Respond mode
- Conversational

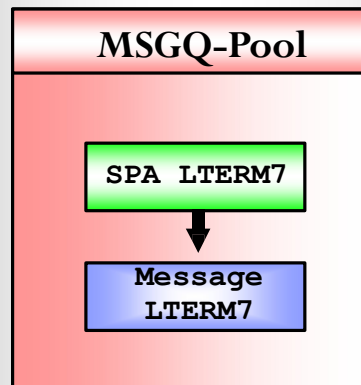
Notes:

IMS maintains a Scratch-Pad Area (SPA) between iterations of conversational programs. TRAN definition with a SPA sized to maximum of 7 x 32k forces this underlying program to be "conversational", also sets the terminal entering this TRAN code into that special session mode (no other tran or cmd can be entered now).

Scratch Pad Area overview

... a
closer
look
into
SPA

I M S C T R L E x t P A



- Any IMS transaction with a SPA is a *Conversational Transaction*
- SPAs are created for IMS transactions that have the optional "SPA=" parameter specified on their TRANSACT macro or has the IMS Dynamic Resource Definition attribute of "CONV(Y)" specified

 NESSLICH CONSULTING

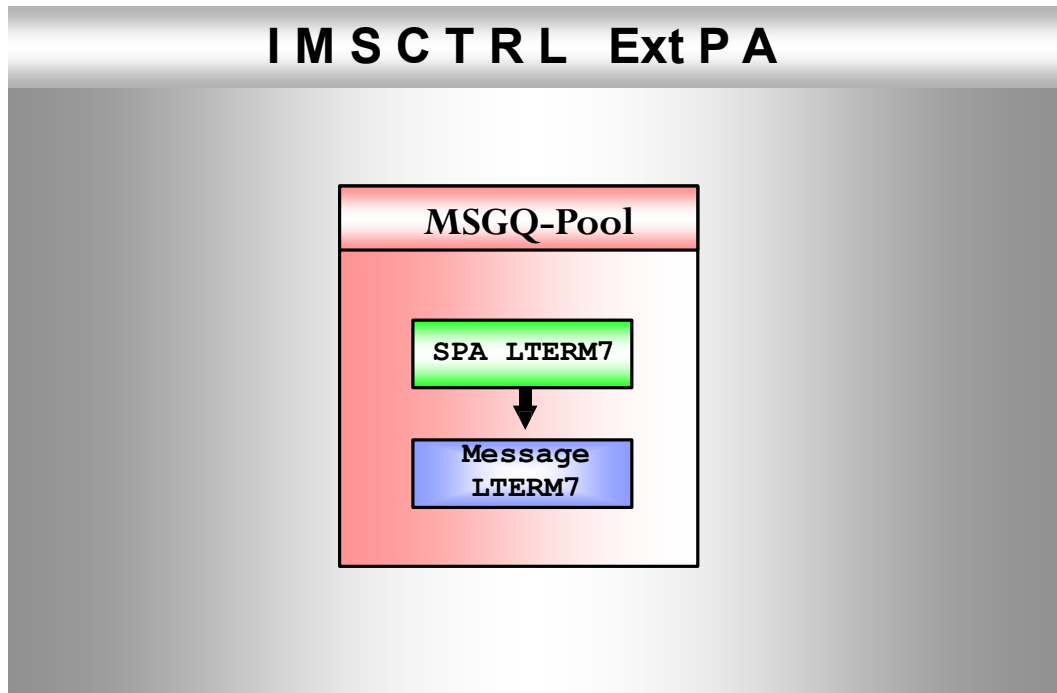
KC110 unit 2 page 36

The SPAs are user (ETO) or terminal (static) related work areas. IMS creates the SPA when a CONVERSATION is started. Every conversational dialog step gets the SPA via GU IOPCB SPA.

Program requested size specified in SPA= parameter of TRANSACT macro or the DRD SPA size attribute of a transaction definition.

Scratch Pad Area ...

IMS CTRL Ext P A



MSGQUEUE Pool Tuning discussion (incl. SPA):

- [Message queue pool tuning - IBM Documentation](https://www.ibm.com/docs/en/ims/15.5.0?topic=tuning-message-queue-pool)
(<https://www.ibm.com/docs/en/ims/15.5.0?topic=tuning-message-queue-pool>)

The SPAs are user (ETO) or terminal (static) related work areas. IMS creates the SPA when a CONVERSATION is started. Every conversational dialog step gets the SPA via GU IOPCB SPA.

Program requested size specified in SPA= parameter of TRANSACT macro or the DRD SPA size attribute of a transaction definition.

<https://www.ibm.com/docs/en/ims/15.5.0?topic=tuning-message-queue-pool>