Telcordia Technologies

Performance from Experience

IMS Technical Conference

IMS Connect Experiences at Telcordia

Session C04

Steve Nathan
stephen.nathan@telcordia.com

Disclaimer

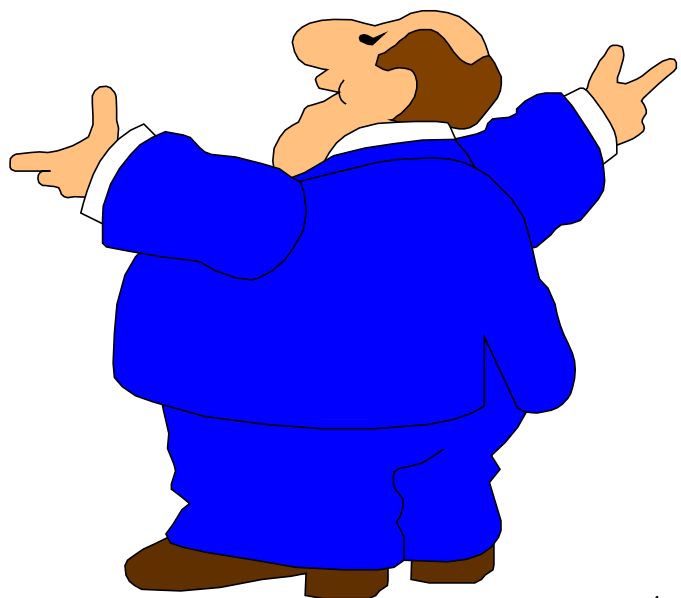
- The purpose of this presentation is to provide a technical perspective of Telcordia's experience using IMS and IMS Connect.
- Although this document addresses certain IBM products, no endorsement of IBM or its products is expressed, and none should be inferred.
- Telcordia also makes no recommendation regarding the use or purchase of IMS or IMS Connect products, any other IBM products, or any similar or comparable products.
- Telcordia does not endorse any products or suppliers. Telcordia does not recommend the purchase or use of any products by the participants.
- Each participant should evaluate this material and the products himself/herself.

Acknowledgements

- Special thanks to:
  - Gerald Hughes and his team from IBM IMS Connect development for all their hard work, support and enhancements
  - Jack Yuan and his team from IBM IMS OTMA development for all their support and enhancements
  - Alan Ho, Virgil Aguilar, and the rest of IBM IMS Level 2 for all their patience and support

Presentation Outline

- Introduction
- IMS OTMA
- IMS Connect
- IMS Connect Applications



Telcordia Technologies

Performance from Experience

Introduction

- Telcordia has given several presentations about its experiences using OTMA and IMS Connect to put Web browser front-ends on to existing IMS applications
- There have been many changes and enhancements in the last year
- We have implemented a major new IMS Connect application using new IMS Connect features and finding new IMS Connect and OTMA enhancement opportunities
- This presentation will concentrate on what is new and exciting

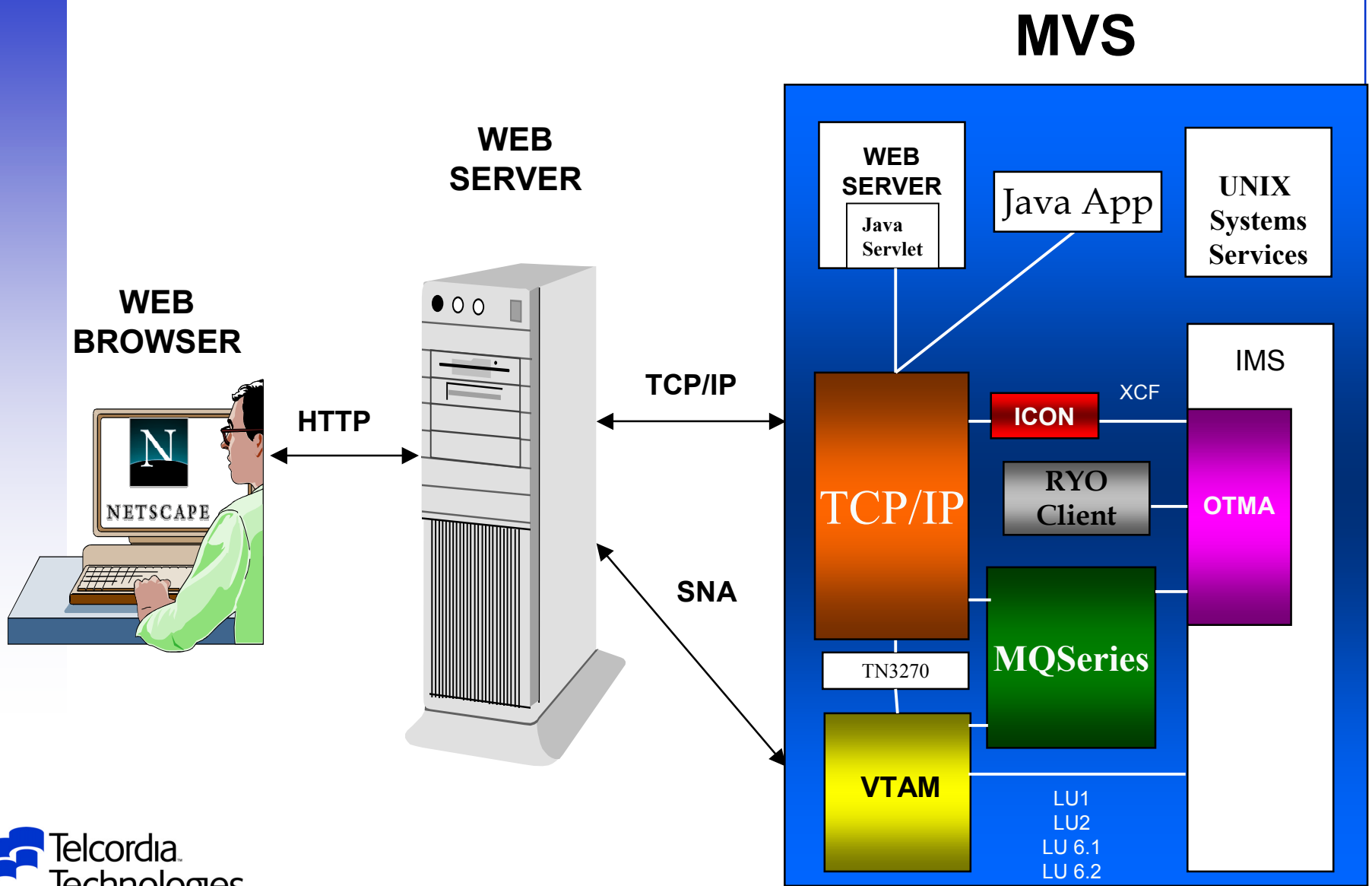
Introduction

- IMS/ESA 5.1 introduced the OTMA (Open Transaction Manager Access) feature
  - There have been MANY enhancements since then
- This feature uses the MVS cross-coupling facility (XCF) to send data to IMS from other MVS applications (OTMA clients)
  - No VTAM or TCP/IP is involved

Introduction

- IMS Connect is an IBM provided OTMA client for TCP/IP
- MQSeries includes an IMS OTMA client
  - "MQSeries-IMS Bridge"
- You can write your own OTMA client using the OTMA Callable Interface

Introduction



Telcordia Technologies
Performance from Experience

IMS OTMA Interface

- TPIPEs (Transaction Pipes)
  - OTMA equivalent of LTERMs
  - Input TPIPE names are specified by the client
    - *For IMS Connect for CM1 messages it is the TCP/IP Port*
    - *For IMS Connect for CM0 messages it is the Client Name*
  - Asynchronous output (ALTPCB) TPIPE names
    - *The destination in the CHNG call for modifiable ALTPCB's*
    - The destination name in static ALTPCB's
    - Can be overridden by the DFSYDRU0 exit
  - Control blocks are dynamically created by IMS
  - TPIPE name must be unique only within a client (multiple clients can use the same TPIPE name)
    - IMS recognizes TPIPEs as **XCFmember**.**TPIPEname**
  - A TPIPE name cannot be the same name as an IMS transaction
  - Client manages the use of TPIPEs (number, routing, etc.)

IMS OTMA Interface

- OTMA Commit Mode
  - Specified by the OTMA client for each *input* message
  - Determines how *output* messages are sent
  - Similar to APPC-IMS Commit Mode
  - Commit Mode 0 - Commit-then-send
    - Output is queued
    - IMS sends the output after syncpoint is complete
    - Queued on a control block called a QAB
    - *OTMA requires ACK to dequeue the message*
    - *Always used for ALTPCB output*

IMS OTMA Interface

- OTMA Commit Mode
  - Commit Mode 1 - Send-then-commit
    - IMS sends the output and may wait for an ACK before syncpoint is complete
      - Synch level 0 (None) – no ACK
      - Synch level 1 (Confirm) – ACK
      - Set in *input* message OTMA prefix
    - Increases region occupancy
    - Output messages are not queued
    - If the message can not be sent to the client or is NAKed by the client (SL1) the transaction abends with U119 and backs out
    - For SL0 even though the OTMA client gets the message before the syncpoint is complete it should not send the message until it receives a "deallocation" message from OTMA indicating the status of the syncpoint
      - If the syncpoint did not complete successfully the OTMA client should discard the message and send an error message to the client instead
      - IMS Connect does this

IMS OTMA Interface

- The OTMA client communicates information to IMS and gets information from IMS in a prefix passed in front of the message
  - The prefix has 4 sections mapped by macro DFSYMSG:
    - **Control**: TPIPE name and type, message type, chaining, etc.
    - **State Data**: Commit mode, Sync Level, IOPCB LTERM and MODNAME override, etc.
    - **Security**: Security scope, userid, RACF group, Utoken
    - **User**: Client specific
      - ICON shares with the application

IMS OTMA Exits

- DFSYIOE0: Input/output edit exit
  - Used to modify the length or data of segment
  - Can cancel a segment or a message
    - *Can not return a message to the ICON Client*
  - Gets the address of the OTMA prefix User Data
    - Can be updated
    - Cannot change length
  - Can provide IOPCB LTERM override name on input
  - Can provide IOPCB MODNAME override name on input
  - Can provide MODNAME override name on output
  - Can be used to invoke MFS Segment Edit Routine
    - This is not standard or documented – but I did it
    - You have to establish the environment
    - See next foil
  - *Gets the address of the IMS Control Region initialization exit table (DFSINTX0)*

IMS OTMA Exits

```
CLI    PIOFLAG,X'00'          INPUT MESSAGE?
BNE    RC0                       NO  - RETURN - RC=0
CLI    PSEGFLAG,X'00'         FIRST MESSAGE SEGMENT
BNE    RC0                       NO  - RETURN - RC=0
MVC    MFSSEGE,ZEROES         CLEAR MFS SEG EDIT AREA
MVI    SEGVECT,X'0A'          SET VECTOR TO 10
L      R3,PSEGADDR            ADDRESS OF MESSAGE SEGMENT
ST     R3,SEGADDR             STORE IN MFS SEG EDIT AREA
L      R15,=V(DFSME120)       ADDRESS OF MFS SEG EDIT RTN
LTR    R15,R15                IS IT THERE?
BZ     RC8                       NO - RETURN - RC=8
LA     R1,MFSSEGE             ADDRESS OF MFS SEG EXIT AREA
BALR   R14,R15                BRANCH TO ROUTINE
LTR    R15,R15                ZERO RETURN CODE?
BZ     RC0                       YES - RETURN - RC=0
C      R15,=F'4'              CANCEL MESSAGE SEGMENT?
BE     RC4                       YES - RETURN - RC=4
B      RC8                    TREATE ANYTHING ELSE AS AN 8

ZEROES   DC    24XL1'00'

MFSSEGE  DS    0CL24                  MFS SEGMENT EDIT EXIT AREA
         DS     XL3
SEGVECT  DS     X                     EXIT VECTOR
         DS     F
SEGADDR  DS     F                     ADDRESS OF SEGMENT
         DS     F
         DS     CL8
```

- DFSYIOE0 invocation of MFS Segment Edit Routine

Telcordia Technologies
Performance from Experience

IMS OTMA Exits

- There are two IMS OTMA output routing exits
  - DFSYPRX0: “Pre-routing exit”
    - *Should be “Destination Resolution Exit”*
  - DFSYDRU0: “Destination resolution exit”
    - *Should be ”OTMA User Data Formatting Exit”*
- Invoked for CHNG call to modifiable ALTPCB
- Invoked for ISRT call to static ALTPCB
  - Invoked in XM mode so no SVCs or IMS services
- Invoked even if input message was not from OTMA
  - Allows asynchronous output to OTMA
- Not invoked for ISRT to IOPCB
- Not invoked for ISRT to transaction

IMS OTMA Exits

- DFSYPRX0:
  - RC=0: Input message came from OTMA, destination is same or different OTMA client or input message did not come from OTMA, output is not OTMA
  - RC=4: message originally did not come from OTMA, but destination is OTMA
    - Need to set XCF member name of OTMA client
    - *The trick is to determine which OTMA client gets the message*
      - *More later*
  - RC=8: message came from OTMA, but destination is not OTMA
  - Needed because there can be a different DFSYDRU0 exit for each client
    - This exit determines the client and thus the DFSYDRU0 exit to invoke

IMS OTMA Exits

- DFSYDRU0: (default name)
  - Each OTMA client can have their own exit
  - Name of the exit for a client can be specified in DFSYDTx in the IMS PROCLIB
  - Name of the exit can be overridden by the OTMA client when it connects
    - In CSQFSYSP in CSQZPARM for MQSeries
    - On DATASTORE card for IMS Connect
    - Via otma_openx function for OTMA Callable Interface (PQ32398)

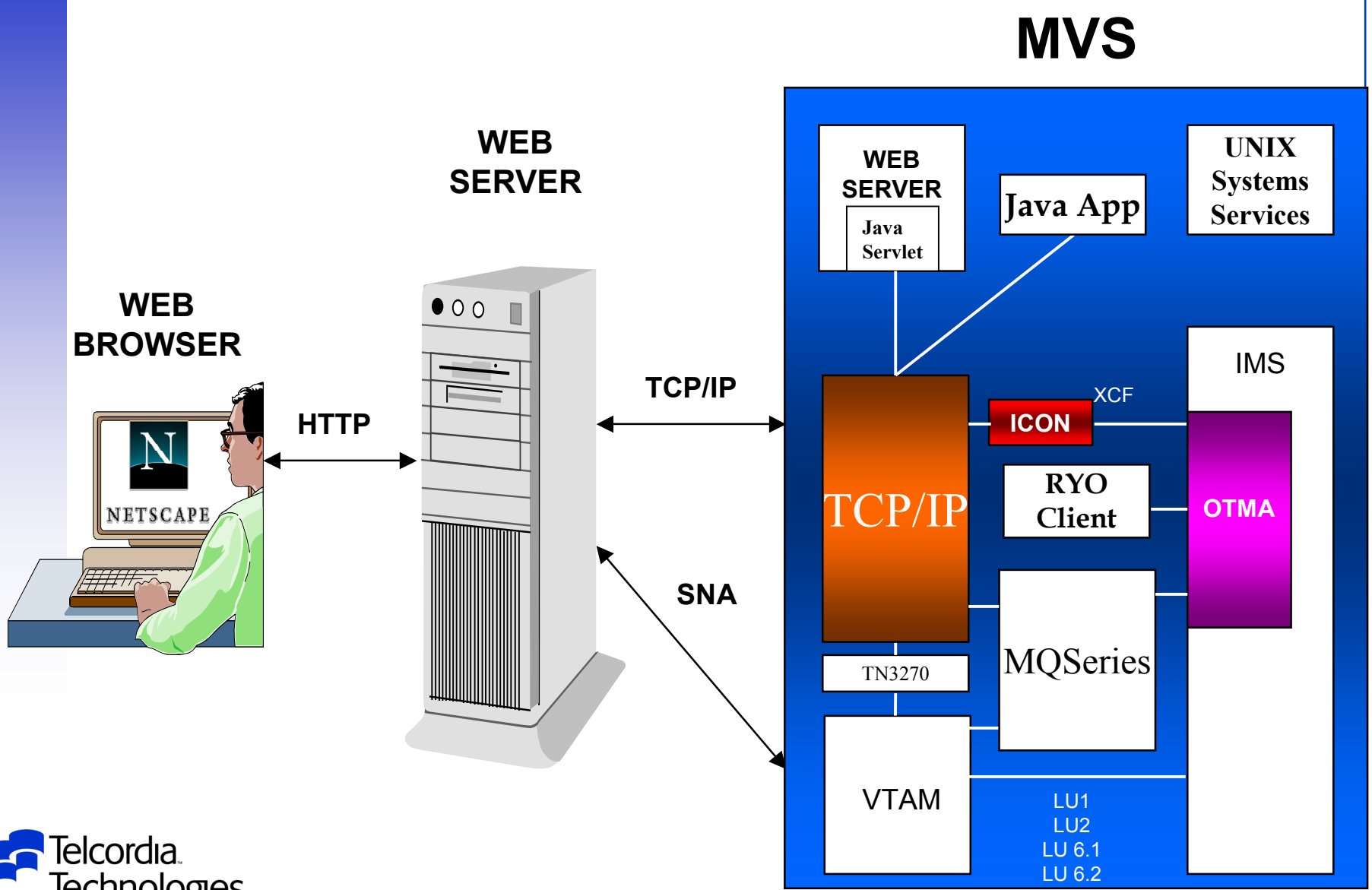
IMS OTMA Exits

- DFSYDRU0: Destination Resolution exit
  - RC=0: destination is the original OTMA TPIPE
  - RC=4: destination is non-OTMA LTERM
  - RC=8: destination is new OTMA Client (need to specify)
    - The new client DRU0 exit will then be invoked
    - No need for this after PQ32402
  - RC=12: destination is invalid
    - A1 status on CHNG call
  - If DFSYPRX0 is coded properly the destination resolution should not take place here
    - Always return RC=0
  - Can override the output TPIPE name - PQ27207
    - Client is waiting on a specific TPIPE
  - *Must format the OTMA User Data for async output*

IMS Connect

- IMS Connect is an IBM provided, TCP/IP server and an IMS OTMA client
  - Runs in a separate address space
- Accepts input messages from TCP/IP client
  - Could be any TCP/IP Socket application
  - Could be Java Application, Bean, Applet, Servlet
- Passes input message to user exit for formatting
- Sends input message to IMS OTMA
- Receives response message from IMS OTMA
- Passes response message to user exit for formatting
- Sends response message to TCP/IP client

IMS Connect



Telcordia Technologies
Performance from Experience

IMS Connect

- There are many wonderful enhancements in IMS Connect compared to ITOC
  - SMP installed and maintained
  - Persistent sockets for CM1 messages
    - ICON normally does a disconnect after each message
    - Looking to expand to CM0 messages
  - Enhanced dump formatting

IMS Connect

- More wonderful enhancements in IMS Connect
  - Initialization Exit
    - User can store up to 2,000 bytes of data to pass to Message Edit exits
    - We do not use yet
  - Datastore Table
    - List of current IMS datastores and their status
    - Passed to Message Edit exits
    - Can be used to route messages to surviving IMS copies
    - We do not use yet – but I know someone who does – and loves it
      - It works exactly as advertised

IMS Connect

- More wonderful enhancements in IMS Connect
  - Support for asynchronous output from IMS (7.1)
    - ALTPCB output
    - NAK'ed output
    - IMS OTMA puts the messages on a special OTMA ICON Async Output Queue
    - *Critical to our new application*

IMS Connect

- More wonderful enhancements in IMS Connect
  - Send-Only messages
    - Prevents DFS2082 from Nonresponse Mode transaction which does not reply
    - *If the application does send a reply to the IOPCB it will be NAK'ed by IMS Connect and the message will be put on the OTMA ICON Async Output Queue*
      - *Until PQ61137*
      - *More later*
    - *Critical to our new application*

IMS Connect

- More wonderful enhancements in IMS Connect
  - Resume TPIPE
    - Allows ICON clients to retrieve messages from the OTMA ICON Async Output queue
    - *Critical to our new application*

IMS Connect

- One ICON can connect to multiple IMS control regions in multiple XCF groups
- One ICON can have multiple connections to the same IMS copy
- One IMS control region can connect to multiple ICON's
- ICON and IMS can be on different MVS copies in the same Sysplex
- *We use all of these configurations*

IMS Connect

- ICON runs as an MVS job or started task
- Controlled by two input files:
  - BPECFG
    - Internal trace parameters
    - Just use shipped sample
    - Only change trace parameters if requested by IMS
  - HWSCFG
    - Next page

IMS Connect

- HWSCFG
  - HWS control card
    - Gives name to this ICON instance
    - Defines RACF processing
    - Number of fullwords in the initialization table

IMS Connect

- HWSCFG
  - TCP/IP control card
    - HOSTNAME - TCP/IP Host name (TCP/IP JOBNAME)
    - RACFID - Default RACF ID to assign to messages
    - PORTID - List of from 1 to 50 ports for ICON to listen on
    - MAXSOC - Maximum number of concurrent sessions on each port (50 - 2000; default=50)
    - *TIMEOUT - Time interval in hundredths of a second after which ICON disconnects a client if no response from IMS*
      - *Range = 0 to 2,147,483,647*
      - *Default = 0 (no timeout)*
      - *More later*
    - Exits - List of from 1 to 15 user exits to process messages
    - ECB – Performance enhancement if TCP/IP >= 3.7

IMS Connect

- HWSCFG (continued)
  - DATASTORE control card
    - Translates logical “datastore” name passed by the TCP/IP client into the IMS XCF member name and therefore IMS control region
    - Defines XCF group to join
    - *Defines XCF member name of ICON for this connection*
    - *Defines name of the DFSYDRU0 exit for messages destined for this OTMA client*
    - There can be multiple DATASTORE cards

IMS Connect

- To use ICON you must write one or more user exits
- Each exit gets control during ICON initialization and passes ICON two 8 character ID's that TCP/IP client applications can use to identify it
  - One is EBCDIC
  - One is ASCII
- The TCP/IP client must pass a valid exit ID in the prefix of the input message
  - A bad ID will bring down the connection
  - If the ID is ASCII the exit usually assumes that the data is ASCII and should be translated to EBCDIC going to IMS and back to ASCII going to the TCP/IP client
  - You can also use a flag set by the client in the message prefix to determine if translation is required
  - *But the exit can do anything it wants*

IMS Connect

- The exit gets control for input messages and builds the message to pass to OTMA
  - *This includes the OTMA headers*
- The exit translates the message from ASCII to EBCDIC as required
- The exit builds multiple segment input messages from one TCP/IP client message if required
- The exit can optionally invoke a security exit to get a RACF Utoken to pass to OTMA
  - *or you can issue RACF calls directly in the ICON exit or your own security exit*
- The exit passes message length and client name override to ICON in the HWSEXPRM parameter list
- The exit passes other information to ICON in the OTMA headers

IMS Connect

- There is a sample exit provided with ICON
  - HWSSMPL0
- The Assembler Language sample exit is full of lower case comments and variable names and not straightforward logic !!!!!!!!!!!!
- *We used it as a basis for all of our exits – but we wrote our own*
- *The exit sends and receives the Java Client message prefix in the OTMA user data*

IMS Connect

- There is a sample security exit also provided with ICON
- *We wrote our own security exit*
  - *It invokes RACF to verify Userid and optionally Password*
  - *It can also update the users password*
- *It builds and deletes the ACEE for every request*
  - *This is inefficient and I would like to cache them*
- *It does not yet pass a UTOKEN to OTMA*
  - *This is inefficient and I would like to use it*
- *Be careful NOT to delete the ACEE if the RACF return code is not zero*
  - *People get upset when ICON abends*

IMS Connect

- The exit gets control for output messages and strips the OTMA headers and builds the output message
- Invokes EBCDIC to ASCII translation as required
- Builds one message to the TCP/IP client from multiple segment IMS output messages if required

IMS Connect

- It is possible for a Socket Close issued by a client to reach ICON before all the data from the previous send
  - ICON gets very upset by this
- Affected by TCP/IP “SO_LINGER=Y/N, VALUE=N”
  - SO_LINGER=Y,VALUE=0
    - Immediate return to Client Code
    - Socket Close can lose previously sent data
  - SO_LINGER=N
    - Immediate return to Client Code
    - Socket Close can lose previously sent data
  - SO_LINGER=Y,VALUE=10
    - Return to Client Code when ACK is received from the host or after 10 seconds
    - Socket Close should not lose previously sent data
  - Thanks to Gerald Hughes of IBM ICON development for this tip

IMS Connect Applications

- Telcordia is a vendor of IMS applications
- We have been building IMS applications for over 20 years
- We had already implemented 3 applications using the IMS TCP/IP OTMA Connection (ITOC) before ICON and all of its enhancements
  - One of these was CM1 IOPCB input to a nonresponse transaction
  - Two of these were IOPCB input/output transactions

IMS Connect Applications

- In the first application a Windows client passes requests for data in the form of tag-value contracts to an existing NONRESPONSE IMS MPP
- The MPP puts the request in an IMS database
- A BMP job reads the request and returns the data via FTP
- The message is set to Commit Mode 1 so that IMS returns a DFS2082 message which is translated by the ICON exit into a successful completion message

IMS Connect Applications

- A second application stores very complex telephone network work requests in the form of text documents in an IMS database which customers wanted displayed at a workstation
- The workstation builds graphic representations of telephone network requests
- Screen scraping required many interactions and IMS transaction schedules
- New IMS contract interface transactions were developed to send large amounts of data to the workstation in one transaction
- Initial response time was reduced from 60 seconds to less than 5 seconds
- Complex process completion time was reduced from several minutes to 15 seconds
- Users LOVE it

IMS Connect Applications

- A third application wanted to build a Web front-end to allow more customer interaction with the system which processes all 800 and 8xx numbers
- IMS Connect sends and receives existing MID/MOD data to and from a user written Java application
- Uses the ICON exit to provide data translation and security checking
- Uses DFSYIOE0 to invoke an MFS segment edit routine

IMS Connect Applications

- The newest ICON implementation provides a generic Web front-end to 3270-based IMS transactions for two major applications
- We tried using 3270 screen scraping and vendor screen customization tools but we were not satisfied with the performance or capabilities
- The application uses ICON to send the normal transaction output to a workstation where it is formatted by application Java code

IMS Connect Applications

- These are applications are not straightforward IOPCB input and output
  - Some transactions return only IOPCB output
  - Some transactions return only ALTPCB output
  - Some transactions return both
    - The users have been trained in the use of PA2
  - Some transactions are worse
    - Receive input from an LTERM and save the name in an IMS database
    - Send a message to a non-IMS system
    - Receive an IMS transaction reply from the non-IMS system
    - Look up the LTERM in the IMS database and send a reply to the original user

IMS Connect Applications

- All of these permutations made it very difficult to program the Java Client
- It never knew if it was going to get back no reply or one reply or two replies or even more
- The solution was to send every message to IMS Connect as “send-only” and retrieve all messages using “resume TPIPE”
  - IOPCB output would be NAK’ed by ICON and go on the OTMA ICON Async Output Queue
- The Java Client would be in complete control of the retrieval of messages

IMS Connect Applications

- This worked very well until the first time an application sent back both ALTPCB and IOPCB output
- The details are very long and confusing – but the bottom line was that IMS Connect forgot that the input message was “send-only” and sent the ALTPCB “resume TPIPE” output to the Java Client followed immediately by the IOPCB output
- The Java Client ACK’ed the ALTPCB output and then tried to ACK the IOPCB output
- ICON was “too” smart and”knew” that an ACK had just been sent and rejected the second ACK
- Now there was a message in OTMA which would never get dequeued

IMS Connect Applications

- The OTMA and ICON developers (Jack and Jerry) very nicely and quickly provided a coordinated solution
- An additional “send-only” flag was added to the OTMA header
- Now OTMA would put “send-only” IOPCB output on the OTMA ICON Async Output Queue immediately rather than sending it to ICON and getting it NAK’ed
- This required a one line change to the ICON user exit to set the flag

IMS Connect Applications

- These are the fixes for this coordinated solution
  - IMS 7.1
    - PQ60534
    - UQ67027
  - IMS 8.1
    - PQ61093
    - UQ67034
  - ICON 1.1
    - PQ60498
    - UQ67031
  - ICON 1.2
    - PQ61137
    - UQ67032

IMS Connect Applications

- The Datastore cards in the IMS configuration input specified an application specific Destination Resolution exit
- This exit received control for ALTPCB output and had to format the OTMA User Data
  - The first part of this data is specified by IMS Connect
  - The second part of this data had to be a “dummy” client message prefix so that the IMS Connect user exit could properly format the output message back to the Java Client

IMS Connect Applications

- One nice feature of this implementation is that if there are more messages waiting on the OTMA Async Output Queue when a message is delivered with the “resume TPIPE” OTMA will set a flag in the OTMA header indicating that fact
- The ICON client can keep issuing “resume TPIPE” commands until the queue is drained
- APAR PQ53176 (UQ62676) (IMS 7.1) addresses the problem where this flag was set on in error when there were no more output messages
  - PQ57025 (UQ65704) is for IMS 8.1

IMS Connect Applications

- One problem we found was the loss of the MODNAME specified by the application with the ISRT call
- Normally this is passed back in the OTMA header
- However when ICON NAK'ed the "send-only" output and OTMA put the message on the OTMA ICON Async Output Queue it did not copy the MODNAME in the message header
- The application got the message and did not know how to format it
- This was quickly fixed by PQ54625 (UQ61200)

IMS Connect Applications

- Another minor problem we found was that IMS Connect issued message HWSP1435E “Socket closed; Request message incomplete; M=SDRC” for each “send-only” input message
- The request message was actually complete and processed properly
  - All the HWSP1435E message did was scare developers and fill up the ICON JES JOBLOG
- This problem is fixed with PQ53848 (UQ68076) for ICON 1.1 and PQ62496 (UQ68078) for ICON 1.2

IMS Connect Applications

- The next design issue was to determine how long the Java Client should wait for a message after sending the “resume TPIPE” before giving up and telling the user that something was wrong
- IMS Connect supplied 3 options
  - No wait
    - If there was no message on the OTMA ICON Async Output Queue disconnect immediately
  - Wait from .01 to .25 seconds (user specified)
  - Wait forever
- None of these options suited our needs

IMS Connect Applications

- We really wanted to wait about 2 seconds which meant we had to tell IMS Connect to Wait Forever
- After two seconds the Java Client disconnected from TCP/IP and notified the user
- Then the Java Client tried to reconnect to IMS Connect and ICON refused the connection because it thought the Java Client was still connected and waiting forever
- We needed a way to tell IMS Connect to “wait forever or until we tell you to stop waiting”
- APAR PQ54024 has been opened but the target closing date is 12/13/02
  - It is anticipated that it will close earlier
  - This is not critical because of the next item

IMS Connect Applications

- Instead of waiting forever the Java Client waited for the maximum time allowed (.25 seconds)
- If ICON did not a receive a message after the "resume TPIPE" within .25 seconds it disconnected the Java Client
- Now the Java Client could wait for a second and then connect again and issue another "resume TPIPE" with another .25 second wait
  - And do it again and again and again
- We really wanted to be able to specify longer timeout values
- APAR PQ54020 was opened and has a projected closing date of 12/13/02
  - But once again ICON development has come to the rescue!!!

IMS Connect Applications

- ICON development is working on the timeout granularity problem as fast as they can and hopes to be ready long before 12/13.
- They will allow the following timeout values
  - .01 to .25 seconds by .01 second increments
  - .25 to 1 second by .05 second increments
  - 2 to 60 seconds by 1 second increments
  - 2 to 60 minutes by 1 minute increments
- A new return code and reason code will also be returned if the timer expires

IMS Connect Applications

- The granularity timeout fix will also solve another outstanding request
- Currently ALL normal input messages use the same timeout value
  - Specified in the Timeout value in the TCP/IP config card
- Now the timeout value will be able to be set for each input message individually
- The timeout value will be able to be set for each input message, “resume TPIPE” command and ACK

IMS Connect Applications

- A third design issue was how to get the async messages back to the proper IMS Connect datastore
  - Multiple Java Clients using multiple datastores in multiple IMS Connects could all be sending messages to the same IMS
  - All of the client names from the Java Clients for this application started with the same two characters
  - The IMS application would issue the CHNG call to this client name for async output
  - Now DFSYPRX0 knew the message was destined for OTMA but did not know which specific OTMA member (datastore)
  - Previous DFSYPRX0 exits used a table of CHNG destination to OTMA member but there were going to be far to many CHNG destinations for this application

IMS Connect Applications

- During IMS Control Region initialization user exit DFSINTX0 is invoked.
- This exit can build a table
- The address of this table is passed to most (if not all) IMS TM exits including DFSYIOE0 and DFSYPRX0
- DFSINTX0 was used to obtain storage for an area which would be used to communicate between DFSYIOE0 and DFSYPRX0 for this application
- The address of this storage was passed in the DFSINTX0 table

IMS Connect Applications

- When an input message was received for this application DFSYIOE0 would recognize it and store the Client ID and OTMA member name in the communication area
- When the IMS application issued the CHNG call DFSYPRX0 was invoked and looked up the Client ID in the communication area and got the OTMA member name
- Sounds simple – but it is not

IMS Connect Applications

- The most efficient way to store and retrieve the data in the communication area was with a hash table – actually a mini in-core HDAM database
- DFSYIOE0 and DFSYPRX0 must be reentrant and can and will be called by multiple OTMA tasks simultaneously
- The standard exit parameter list passes a 512-byte work area to these exits but this area is the same for all tasks
- There is also no save area chain which programmers have been known to use for a reentrant work area

IMS Connect Applications

- The application specific parameter list passed to DFSYIOE0 and DFSYPRX0 is unique for each OTMA task and can be used for a reentrant work area
- OTMA does not care if you overlay data in this parameter list
- There are a number of parameters in this parameter list which your exit may not care about and which can be used for a reentrant work area
  - RACF group, address of OTMA state prefix, etc.
- However there is still a requirement for IBM to pass a real reentrant work area to DFSYIOE0 and DFSYPRX0

IMS Connect Applications

- Since you can not trust users or customers to tell you the maximum number of Client IDs that will be in this communication area – what do you do if it fills up?
- The first thing DFSYIOE0 does is an auto purge
  - The communication area entry for a client updated with the date every time a client sends in a message
  - When the first message arrives for a new date (after midnight) the code in DFSYIOE0 will free all entries which have not been used for 3 days

IMS Connect Applications

- Warning messages are also issued via IWTO as the communication area fills up
- The Java Client can send in a dummy IMS message requesting that client entries older than only 2 days or only 1 day be freed
- The Java Client can also send a dummy message requesting that statistics about the communication area be displayed via IWTO
  - Number of RAPS and free RAPS
  - Number of data slots and free data slots
  - Current purge criteria
  - Etc

IMS Connect Applications

- Another design problem was duplicate Client ID's
  - How do you keep 2 users from using the same Client ID?
  - How do you allow the user to have multiple sessions on his PC and give each one a unique Client ID to keep the messages correlated
- The answer was to have the Java Client code generate the Client ID and suffix it for multiple sessions
- This still did not insure a unique Client ID
  - This was done using the communication area
  - Before a Client ID was used a dummy message was sent by the Java Client to IMS (DFSYIOE0) to ask that the client ID be put into the communication area
    - If it was already there an error message was returned

IMS Connect Applications

- The Java Client code also requested that the Client ID in the communication area be freed if the user was polite enough to sign off gracefully
  - This helped to keep the communication area from filling up

IMS Connect Applications

- There were some design considerations for these "dummy" transactions
- The IMS transaction code is verified before DFSYIOE0 is invoked so they must be defined in the IMS SYSGEN

IMS Connect Applications

- DFSYIOE0 can cancel a segment or a message but it can not return a message to the OTMA client directly
  - Instead DFSYIOE0 must alter the input message to add the reply and send it to the IMS transaction which just inserts the input message back to the IOPCB to go back to the client
- These “dummy” input messages are CM1 messages and not “send-only” since the client is waiting for an immediate reply

IMS Connect Applications

- Another design consideration was the MFS attribute bytes
- These were used a great deal by the applications and the Java Client wanted to use the information in them to format the output
- MFS attribute bytes do not translate well from EBCDIC to ASCII
- It would have taken a lot of code to translate the data and not the attributes
- A special ICON exit was written which translated the message prefix between the Java Client and IMS Connect but which left the entire message in EBCDIC
- The client processed the attribute bytes and did its own ASCII/EBCDIC conversion of the data

IMS Connect Applications

- One design consideration has not yet been resolved
- If a “send-only” message is sent for a stopped IMS transaction the DFS065 message is not received by the ICON client
  - Nothing is received by the ICON client
- The fix for this requires design changes in both IMS OTMA and IMS Connect
- A requirement has been submitted but there is no timeline for a fix

IMS Connect Applications

- Another longstanding problem which has not been resolved is the automatic connection of IMS Connect datastores if IMS Connect comes up before IMS or if IMS goes down and up or if OTMA is stopped and started
- OTMA broadcasts its availability but IMS Connect is not listening
- You must stop and start IMS Connect or you must issue IMS Connect OPENDS commands
- An APAR was opened but was closed because it required a design change
  - A formal requirement has been submitted

IMS Connect Applications

- There are a number of other minor design considerations and enhancements which I would like IMS OTMA and IMS Connect to work on in their copious free time
- The development organizations are aware of these items and are doing their best to fit them in when they can
- These are listed briefly on the next foil

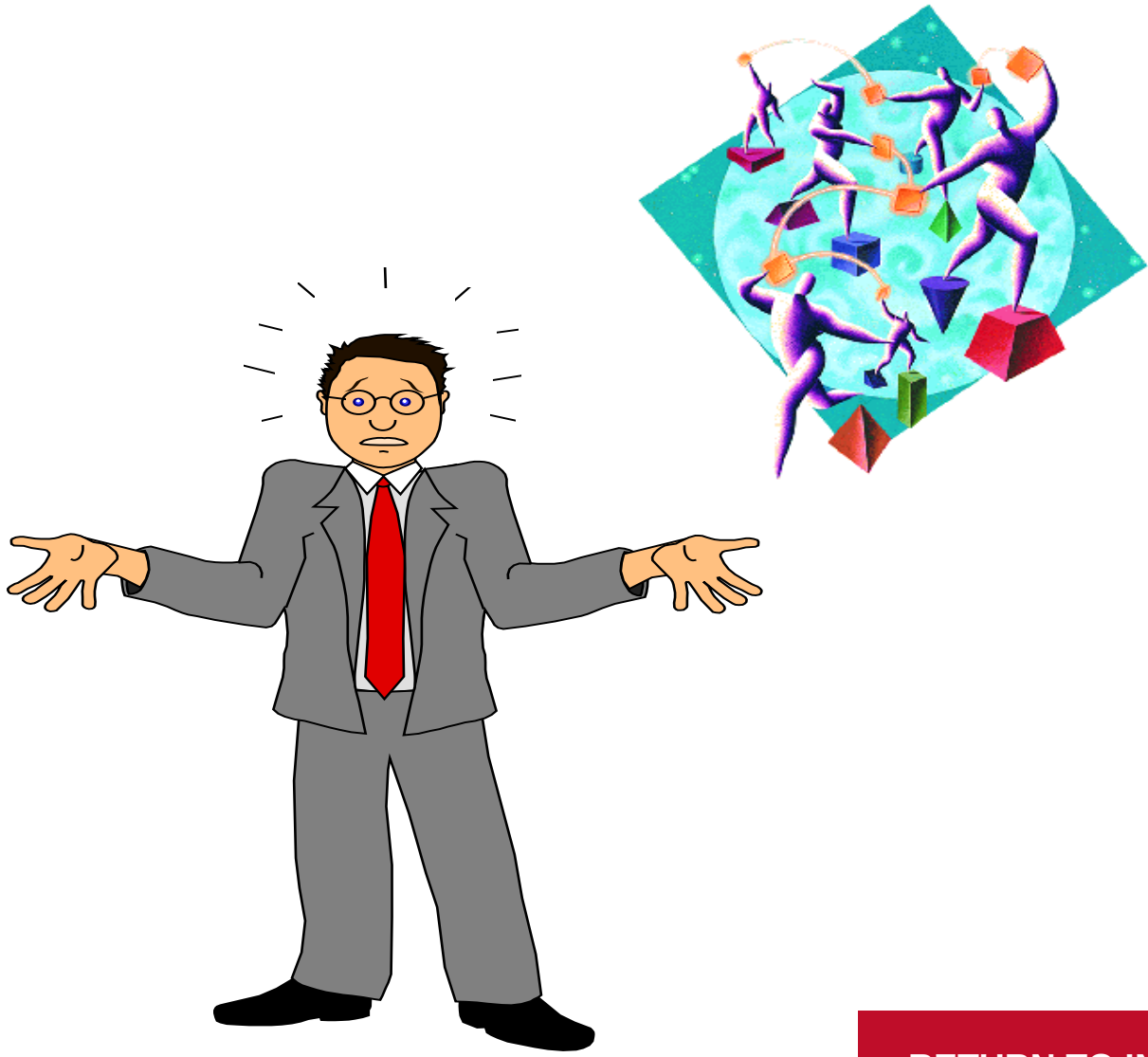
IMS Connect Applications

- Enable DFSYPRX0 to set the TPIPE name
  - This can currently only be done by DFSYDRU0
- The OTMA TPIPE display should show if it is waiting for an ACK
- If OTMA is waiting for an ACK and the ICON client goes away and is restarted it may lose its status and not know an ACK is required
  - It sends in another “Resume TPIPE” which is ignored
  - Treat the second “Resume TPIPE” as a NACK and resend the message or return an error code indicating that an ACK is pending

Conclusion

- IMS Connect and OTMA have allowed us to build many useful interfaces for our clients
- Again many thanks to the OTMA and IMS Connect staff for their outstanding support, the many enhancements they have given us, and the many things they are working on for the future

Questions?



Telcordia Technologies

Performance from Experience

RETURN TO INDEX