Get With the Program: Understanding What IMS Applications Can Do and How to Benefit From APIs

Session IMS-02

Suzie Wendler
IBM – Washington Systems Center
wendler@us.ibm.com

Abstract

IMS clients worldwide have a wealth of assets in the form of IMS applications. These business-critical apps have been running your business for decades, at performance levels hard to rival. Their durability, reliability, and scalability are legendary. IMS continues to add rich function for applications in every release. So how are you taking advantage of them NOW? Whether you are an Enterprise Architect, Application Architect, or Application Developer, you will learn something valuable in this session.

Topics

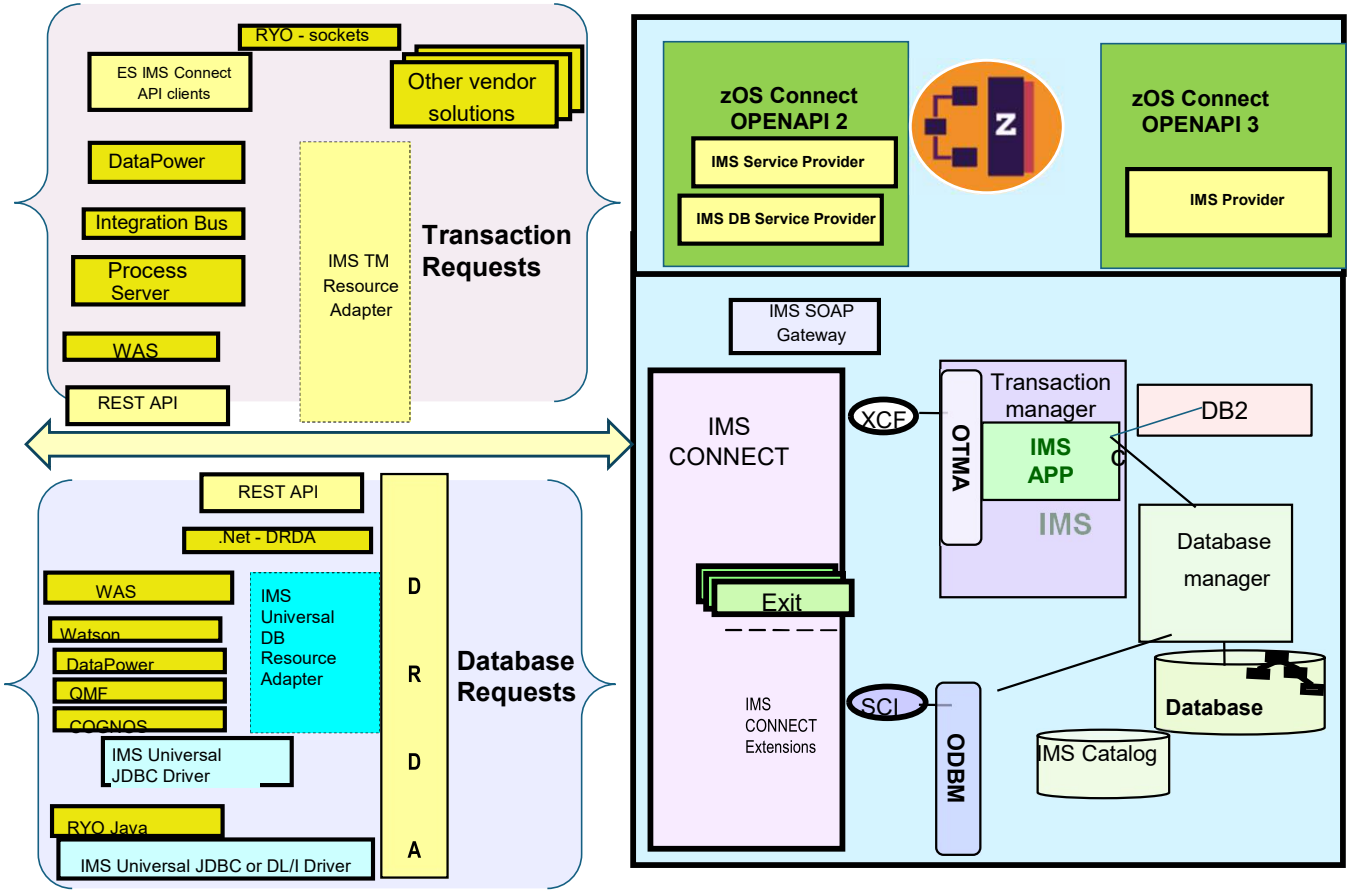
- Expanding technologies – hybrid clouds, mobile apps and REST APIs
  - z/OS Connect and RESTful APIs
- Existing application patterns and support
  - Leveraging what IMS has had for a while
  - Taking advantage of what is new
- What Else
  - Web enablement toolkit – issuing http calls directly from the IMS application
  - Synchronous Callout
  - Transaction Orchestration

The IMS Environment is Ever Evolving ---- >

IMS Connect is a gateway for evolving technology requirements …

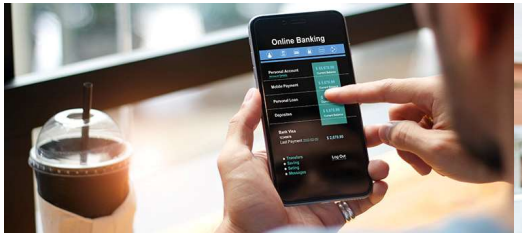
Resilient support for strategic solutions:

- **Cloud**
  - Ability to quickly deploy new services
  - Ability to allow on premise provisioning of IMS resources w/o an outage
- **Analytics**
  - Adding quickly and easily re-deploy applications after adding analytics
- **Mobile**
  - Scale due to increase loads
  - Enhance information
  - Maintain availability for 24x7 access
- **Future….**



And NOW Includes

- **The digital transformation evolution**
  - Many use cases drive data and functionality requirements to assets that are hosted on IBM Z
  - For Example:



Account queries on mobile devices using open APIs

Evolution of online apps To hyrid cloud

Raise credt card issues

- **A key piece of digital transformation → *APIs***
  - They enable the increase of speed in which the business can operate
    - Connect applications through clear protocols and architectures

"More than 90 % of financial institutions use or plan to use APIs to generate additional revenue among existing customers"

Mckinsey & Co, 2021

https://www.mckinsey.com/industries/financial-services/our-insights/from-tech-tool-to-business-asset-how-banks-are-using-b2b-apis-to-fuel-growth

Level Set… APIs

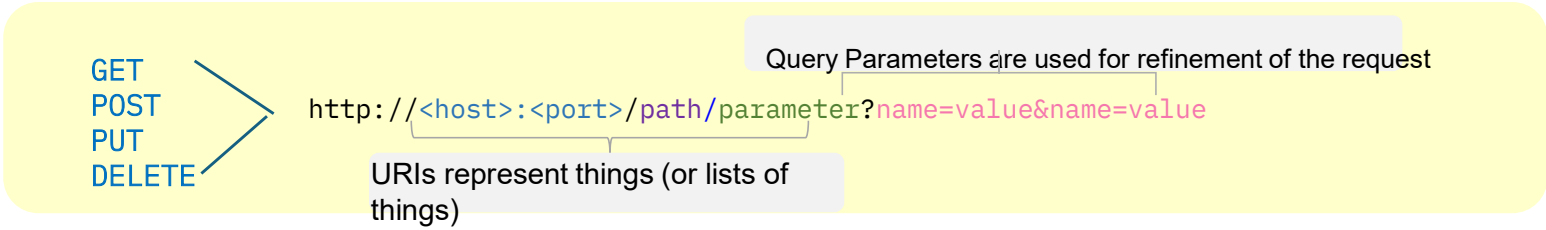
- API architecture - framework of rules and structures for creating software interfaces
  - The rules determine how to provide server functionality to users

- **Popular types of API architectures**
  - SOAP (**S**imple **O**bject **A**ccess **P**rotocol)
  - GraphQL
  - Apache Kafka
  - AsyncAPI
  - RPC (Remote Procedure call)

- **REST (REpresentational State Transfer) API**
  - A type of web API … With a **defined standardized format (OAS - Open API Specification)**
    - ***Definitions and protocols that describe the interface***
  - Enable different applications across multiple platforms to communicate with each other

Level Set…

- RESTful APIs with the Open API Specification
  - A ***client*** application that invokes a RESTful API provides:
    - An identifier for the target server resource This is the URL for the resource, also known as the **endpoint**
    - The operation you want the server to perform on that resource, in the form of an **HTTP method**, or **verb**, *e.g,*

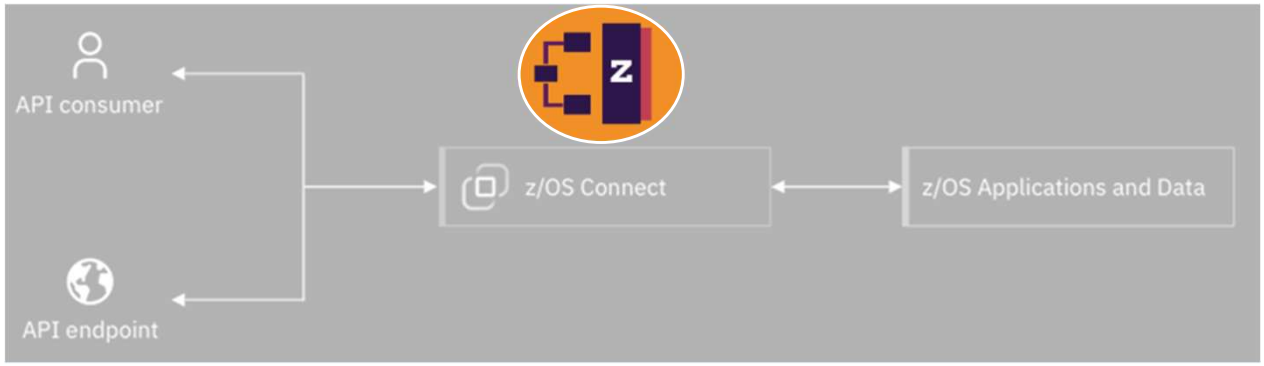


  - The ***server*** responds by *transferring* a *representation* of the *state* of the requested resource
    - Typically in a **JSON** (Java Script Object Notation) format - open standard and commonly used for file and data interchange
      - Uses human-readable text to store and transmit data objects consisting of attribute/name –value pairs and arrays

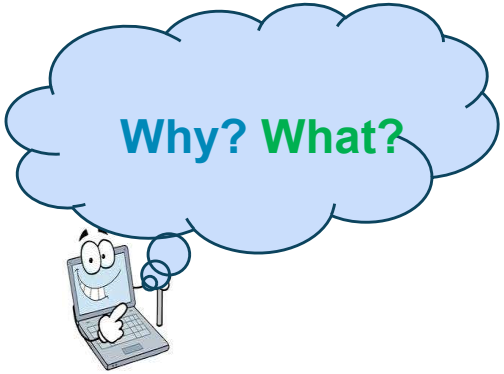
(Note: information transferred via HTTP can be in JSON, HTML, XML, or plain text. JSON is the most commonly used file format for REST APIs. JSON is language-agnostic and readable by humans.)

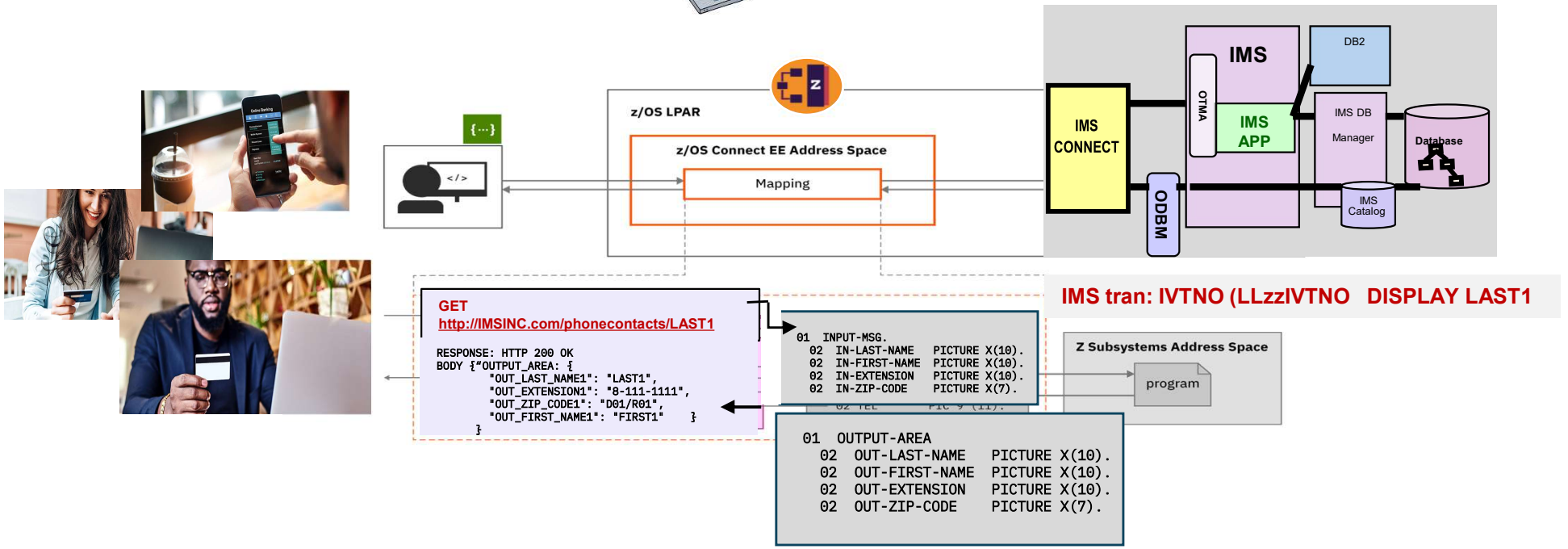
Level Set … And For the Z Environment

- **z/OS Connect allows IBM Z to capitalize on the value of the API economy**
  - Provides ***seamless integration*** with the Z subsystems as truly RESTful APIs
    - Implements ***secure and robust business API***s by leveraging the recognized secure methods of IBM Z
      - *Fast, secure, reliable connectors to reach any z/OS asset*
    - Supports the creation of consumable APIs in minutes to make Z applications/data central to the hybrid cloud strategy
    - Provides Z applications with the capability of calling APIs to enhance them with the power of cloud native functions



z/OS Connect



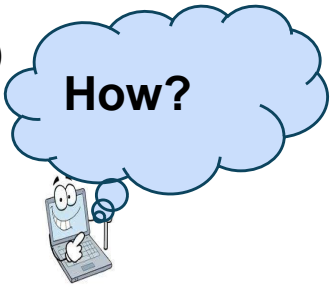


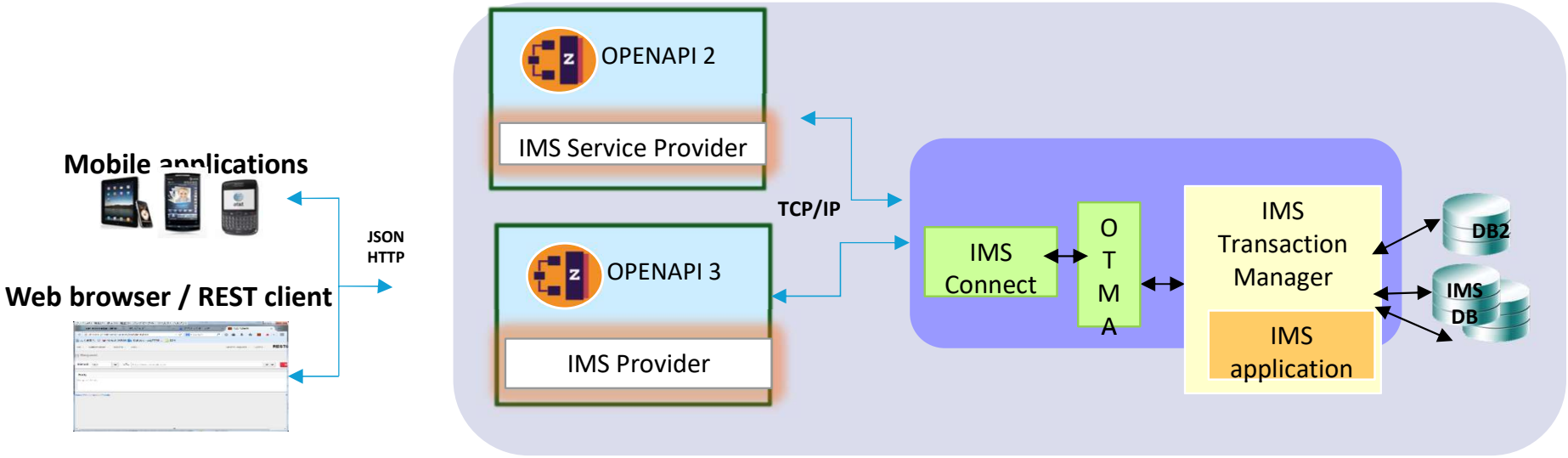
Why has REST become increasingly popular as an integration pattern?
Because it is stateless, relatively lightweight, is relatively easy to program to, and operates well with discovery mechanisms.

What: The software architecture style of the web environment supports communication over HTTP using HTTP verbs (GET, PUT, POST, DELETE…) that browsers use to retrieve and send information

How? --- z/OS Connect (Providers and Tooling)

- ***WebSphere Liberty Profile, z/OS Connect, and IMS SP***
  - ***Liberty Profile*** provides an integrated REST endpoint
    - Supports the lightweight data-interchange format JavaScript Object Notation (JSON)
  - ***z/OS Connect*** is a fast, secure, scalable, and reliable connector for z/OS assets
    - Provides common services and management for consistent operations
      - *A singular approach for System z clients using WAS, CICS, IMS, and DB2*
  - The ***IMS Service Provider*** *(OPENAPI 2- Swagger 2.0)/* ***IMS Provider*** *(OPENAPI 3 – OAS3)* deliver IMS resources to mobile and cloud developers in a secure, governed, and optimized way through:
    - *An integrated platform that supports full discovery, modeling, enablement, and deployment of IMS transactions*





z/OS Connect supports two OpenAPI Standards

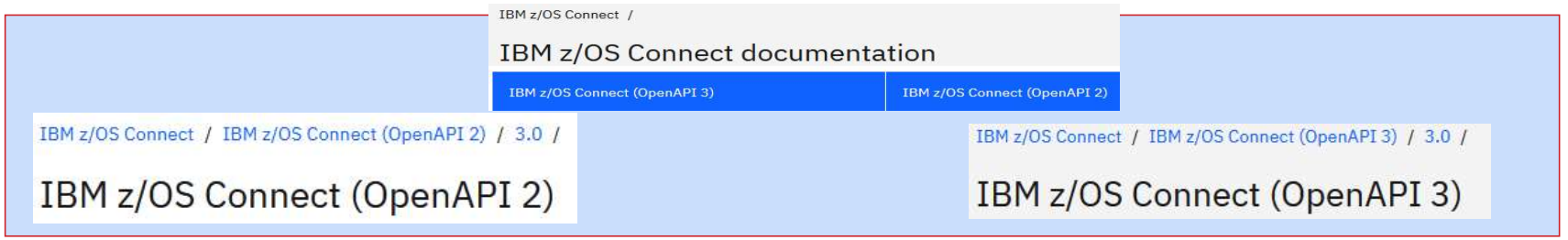
- One product, one deliverable: OPENAPI 2 (Swagger 2.0) and OPENAPI 3 (OAS3)

OPENAPI 2 --> OPENAPI 3 is not a required migration path
Rather, it is a choice based on the use case

  - Knowledge Center home page: https://www.ibm.com/docs/en/zos-connect
    - Each supported OpenAPI specification has own z/OS Connect Knowledge Center

https://www.ibm.com/docs/en/zos-connect/zosconnect/3.0

https://www.ibm.com/docs/en/zos-connect/zos-connect/3.0



OpenAPI2 : *Bottom-up* development

- Starts with the target environment, e.g., IMS application and COBOL copybook to build a service and API to drive it
- Exposes the z/OS resource (transaction/data) to a standard language-agnostic interface to HTTP/HTTPS REST API callers
- Allows the z/OS application to use the same functionality to call external APIs

z/OS Connect produces the specification document that describes the methods and request and response messages.

OpenAPI3: *Meet-in-the-middle or* API First development

- Provides the ability to define the APIs based on the business requirements or those defined by governance boards
  - Development of public APIs that must adhere to external standards bodies or governments that may weigh in on API designs and standards to ensure interoperability across the industry

z/OS Connect consumes the specification document that describes the methods and request and response messages

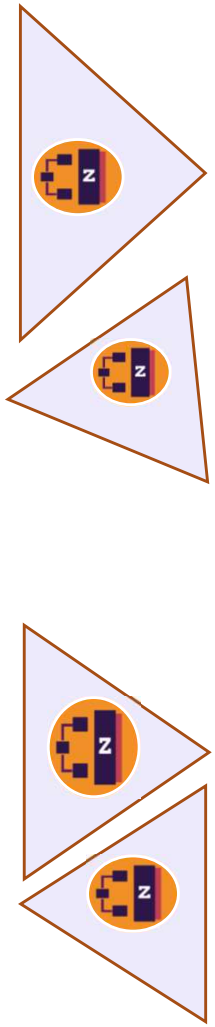
APIs and z/OS Connect Support for IMS

- **OPENAPI 2 – z/OS Connect**

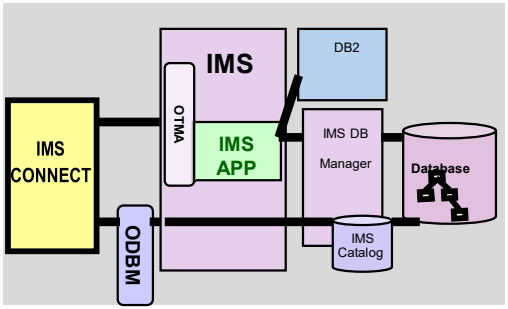
  - ***IMS Service Provider***
    - Access to IMS transactions
    - Access to IMS large messages
  - ***IMS DB Service Provider***
    - Access to IMS databases (Transactions are not involved)
  - ***API Requester***
    - Access from IMS transactions

- **OPENAPI 3 – z/OS Connect**

  - ***IMS Provider***
    - Access to IMS transactions
    - Access to IMS large messages
  - ***API Requester***
    - Access from IMS transactions



- IMS existing infrastructure functionality supports access from both specifications
  - ***IMS Connect*** (IMS TCP/IP gateway)
    - Configured to access transactions and/or databases
  - ***IMS***
    - *OTMA for transactions*
    - *ODBM for databases*
    - *Common Service layer*
      - SCI (structured call interface)
      - OM (Operations manager)



Do the applications in IMS need to be modified?

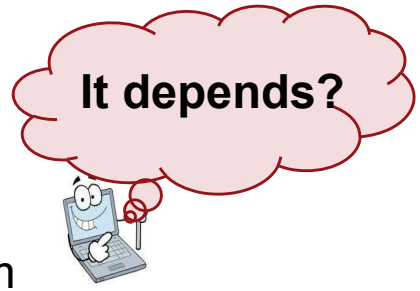
Yes - Requires modifications to the program

In most cases –no- but depends on the application and access pattern and if large messages are involved

No – because no IMS applications are invoked

From an IMS Application Perspective

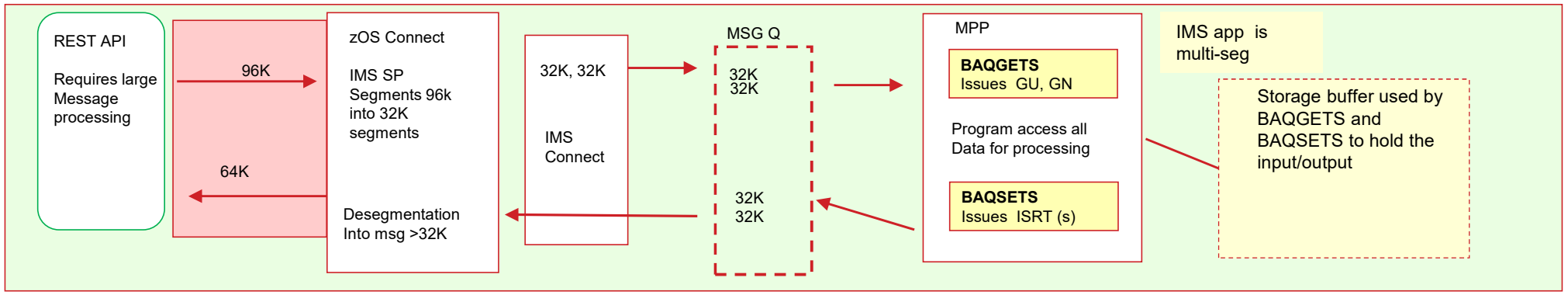
- ***Do the IMS applications require modifications ….??***?



  - It depends
    - **Do you need to** streamline the existing program
      - *Exposing the application to REST APIs can result in opening up pent-up demand*
    - **Or** leverage enhancements for new functionality
      - *Support for large data structures -- **greater than 32K***
      - ***Callout to external APIs***

From an IMS Application Perspective…

- **Do the IMS applications require modifications ….???**
  - **It depends …**
- **For large data structures >32K**
  - **No change if** the IMS application is already (MSGTYPE=MULTSEG) and coded to understand segmentation
    - E.g., GU,GN.. To retrieve a large message and/or ISRT,ISRT to send a large message
  - **Otherwise**
    - Modify the program
    - ***Or*** leverage utilities provided for Cobol, PL/I (if the message >32K)
      - *BAQGETS/BAQSETS (Cobol) or BAQPGETS/BAQPSETS (PL/I)*
      - *Large data structures sent in through the API must not begin with an LLZZ or LLZZ<TRANCODE> prefix*
      - *The BAQGETS utility generates segment prefixes with the first segment prefix set to the transaction code that is specified in the API toolkit*



https://ibm.biz/Bdmf4F (Open API3))

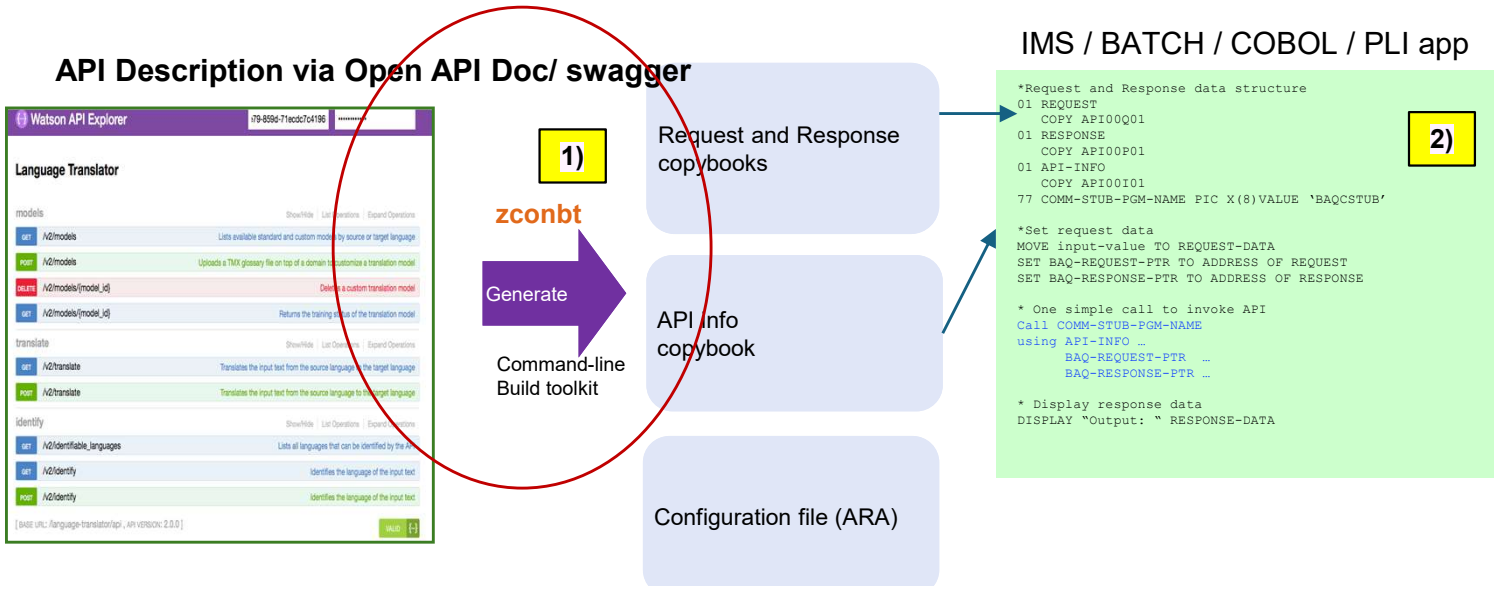
https://ibm.biz/Bdmf4D (Open API2)

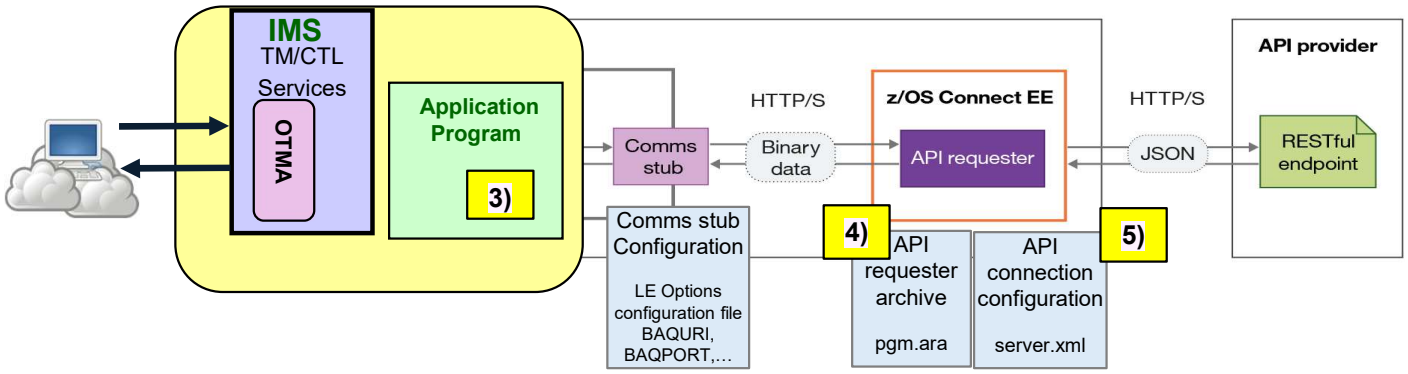
Large Data Structures - Recommendations

- Review the IMS application environment to determine the value proposition
  - If the remote program sending the API request simply need to sends a large unsegmented message (larger than 32K) or retrieve a large unsegmented message
    - If the IMS application is already multi-segment, then the use of this support could allow the IMS application program to remain unchanged
- Ensure the IMS application is multi-segment
  - E.g., Modify the program if it is currently single segment
    - IMS code to do GU, GN calls
    - Or, replace the IMS message calls with the BAQGETS/BAQSETS'
- Understand the impact of BAQGETS/BAQSETS
  - ***Use these if:*** the 'large' message is >32K
  - Large data structures sent in through the API must not begin with an LLZZ or LLZZ<*TRANCODE*> prefix
    - The BAQGETS utility generates segment prefixes with the first segment prefix set to the transaction code that is specified in the API toolkit
  - PH09920 – adds a 'space delimited' option to the API toolkit v3.0.6.5 or v 3.2.6.5 and z/OS Connect EE V 3.0.21
    - option to place only a single blank space character after the trancode, preserving any user data bytes at the end of the 8 byte trancode area.

From an IMS Application Perspective…

- **Do the IMS applications require modifications ….???**
  - For calling out using the **API Requester** support …**YES**
    - **But** tooling generates the code/artifacts which simplifies the process
      - *Note this visual describes: support for Open API2*





1) **Generate the API requester archive (.ara file) and API code from the swagger doc**

2) **Update the IMS application program using the generated client code**
   - **Compile and bind**

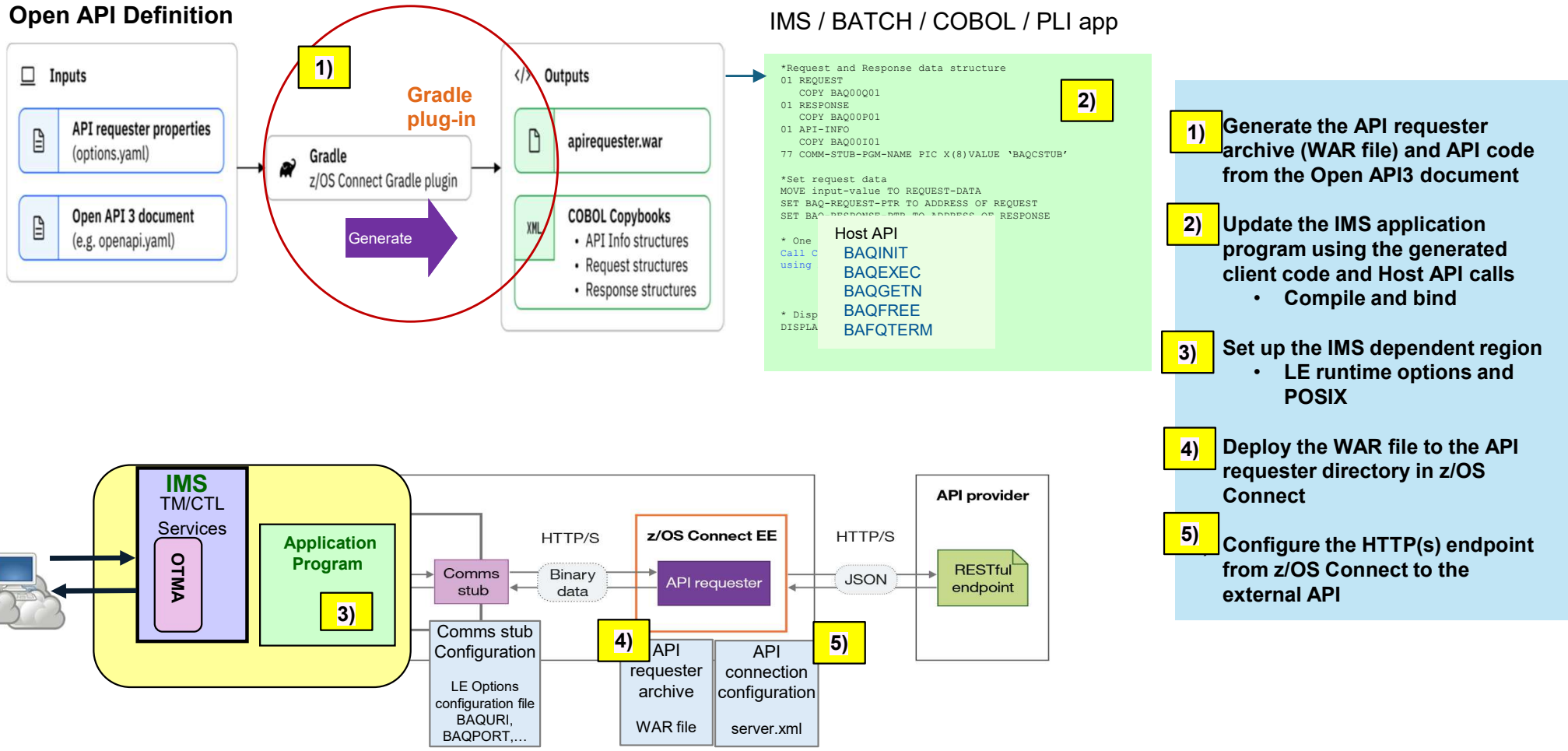
3) **Set up the IMS dependent region**
   - **LE runtime options and POSIX**

4) **Deploy the .ara file to the API requester directory in z/OS Connect**

5) **Configure the HTTP(s) endpoint from z/OS Connect to the external API**

From an IMS Application Perspective…

- **Do the IMS applications require modifications ….???**
  - For calling out using the **API Requester** support …**YES**
    - **But** tooling generates the code/artifacts which simplifies the process
      - *Note this visual describes: support for **Open API3***



1) **Generate the API requester archive (WAR file) and API code from the Open API3 document**

2) **Update the IMS application program using the generated client code and Host API calls**
   - **Compile and bind**

3) **Set up the IMS dependent region**
   - **LE runtime options and POSIX**

4) **Deploy the WAR file to the API requester directory in z/OS Connect**

5) **Configure the HTTP(s) endpoint from z/OS Connect to the external API**

IMS Dependent Region Setup

IMS Compile link-edit RENT with Cobol V4 +

```
//COMPLNK1 EXEC IGYWCL,
//  LNGPRFX='IGYV5R20',
//COBOL.SYSIN    DD DISP=SHR,
//  DSN=IMS.APPL.SOURCE.COBOL(HOTELAPB)
//COBOL.SYSLIN   DD DISP=SHR,
//  DSN=IMS.APPL.COBOL.OBJECT(HOTELAPB)
//LKED.OBJECT    DD DISP=SHR,
//  DSN=IMS.APPL.COBOL.OBJECT
//LKED.RESLIB    DD DISP=SHR,
//  DSN=IMS.SDFSRESL
//  DD DSN=IMS.APPL.COBOL.LOAD,DISP=SHR
//  DD DSN=ZOSCON.V3R0.SBAQLIB,DISP=SHR
//LKED.SYSLIB    DD DSN=SYS1.CSSLIB,DISP=SHR
//  DD DSN=SYS1.SCEELKED,DISP=SHR
//  DD DSN=SYS1.SCEELKEX,DISP=SHR
//LKED.SYSLIN    DD *
     INCLUDE OBJECT(IMMRAPI0)
               INCLUDE RESLIB(BAQCSTUB)for OPEN API2
               of INCLUDE RESLIB((BAQHAPIW) for OPEN API#
     INCLUDE RESLIB(DFSLI000)
     ENTRY IMMAPI
     NAME IMMRAPI0(R)
/*
//LKED.SYSLMOD   DD  DSN=IMS.PGMLIB.PDS(HOTELAPB),DISP=SHR
   (or PDSE if Cobol 5.1+)
/*
```

```
ZOSCON.V3R0.SBAQLIB
Name     Prompt      Alias-of     Size      TTR     AC    AM    RM
BAQCSTUB             BAQWEBT
BAQCTERM             BAQWEBT
BAQEXEC              BAQHAPIW
BAQFREE              BAQHAPIW
BAQGETN              BAQHAPIW
BAQHAPIW
BAQINIT              BAQHAPIW
BAQPUTN              BAQHAPIW
BAQTERM              BAQHAPIW
BAQWEBT
```

Configure the IMS Dependent Region JCL

- CEEOPTS specify LOCALHOST or z/OS Connect URI and port number

```
//CEEOPTS DD DSN=IMS.PROCLIB(MPRLEOPT),DISP=SHR
or
//CEEOPTS DD *
POSIX(ON),
 ENVAR("BAQURI=HTTP://ZOSEE.SERVER.IBM.COM","BAQPORT=7777",
 "BAQVERBOSE=ON","BAQTIMEOUT=60")
```

- STEPLIB

```
//STEPLIB  DD DSN=IMS.SDFSRESL,DISP=SHR
//         DD DSN=IMS.PGMLIB,DISP=SHR
//         DD DSN=IMS.PGMLIB.PDS,DISP=SHR
//**  if COBOL 5.1+  DD DSN=IMS.PGMLIB.PDSE
//         DD DSN=ZOSCON.V3R0.SBAQLIB,DISP=SHR
//*              BAQ Host API  load module
```

API Requester Recommendations

- Web Enablement Toolkit zconbt (OpenAPI 2) or Gradle plug-in (OpenAPI 3)

- Preload the program that issues API Requester callout

- LE Cobol runtime option RTEREUS should not be used in POSIX environment
  - Can cause issues

- Set TLIM = 1
  - If there any abends the MPR is automatically terminated
    - If TLIM is >1, pseudo abends cause DSFRRC00 to reattach DFSPCC20 and problems can occur

- Make Sure you are up to date on maintenance

API Requester Recommendations…

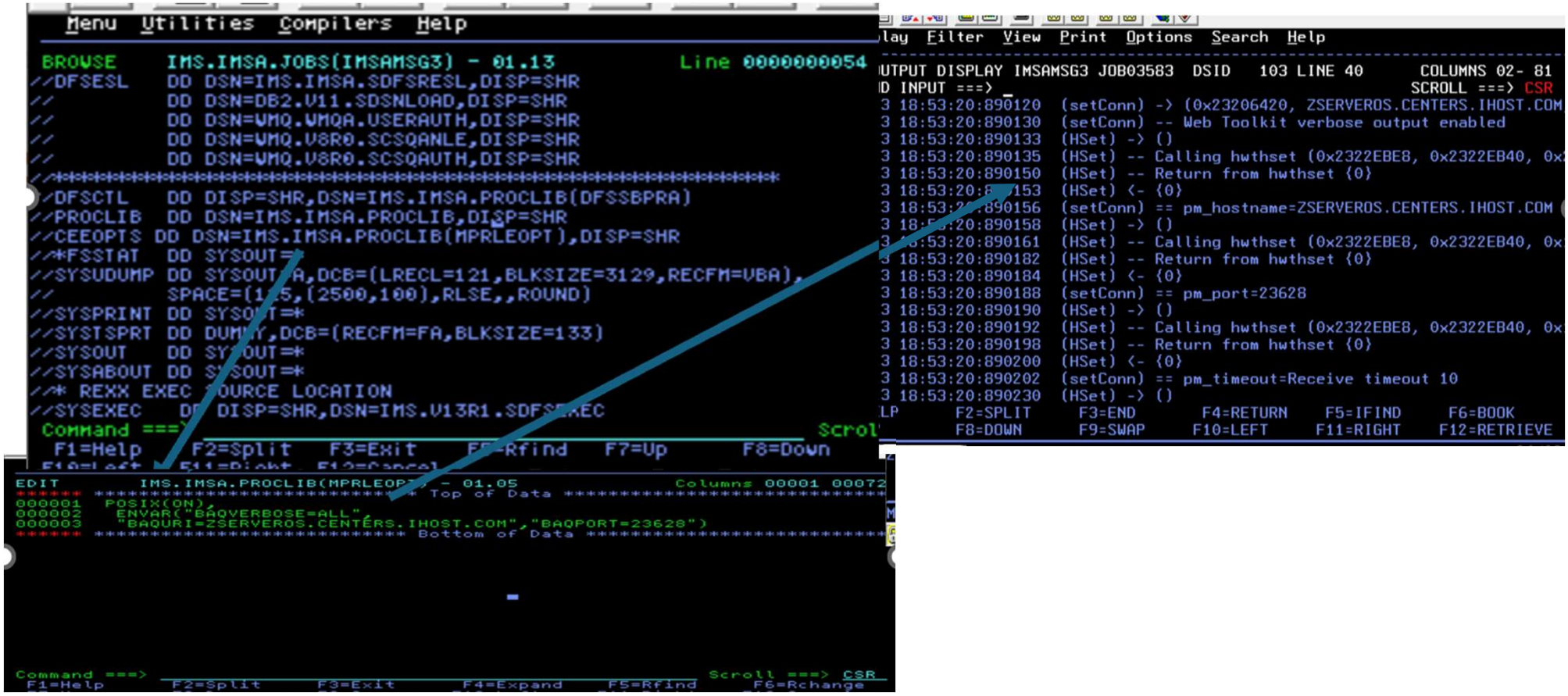
- **POSIX must be enabled in the dependent region**
  - **Create new regions for this environment**
    - Transactions that run in these regions that are non-POSIX may run into problems
    - Cobol runtime option RTEREUS causes problems with POSIX

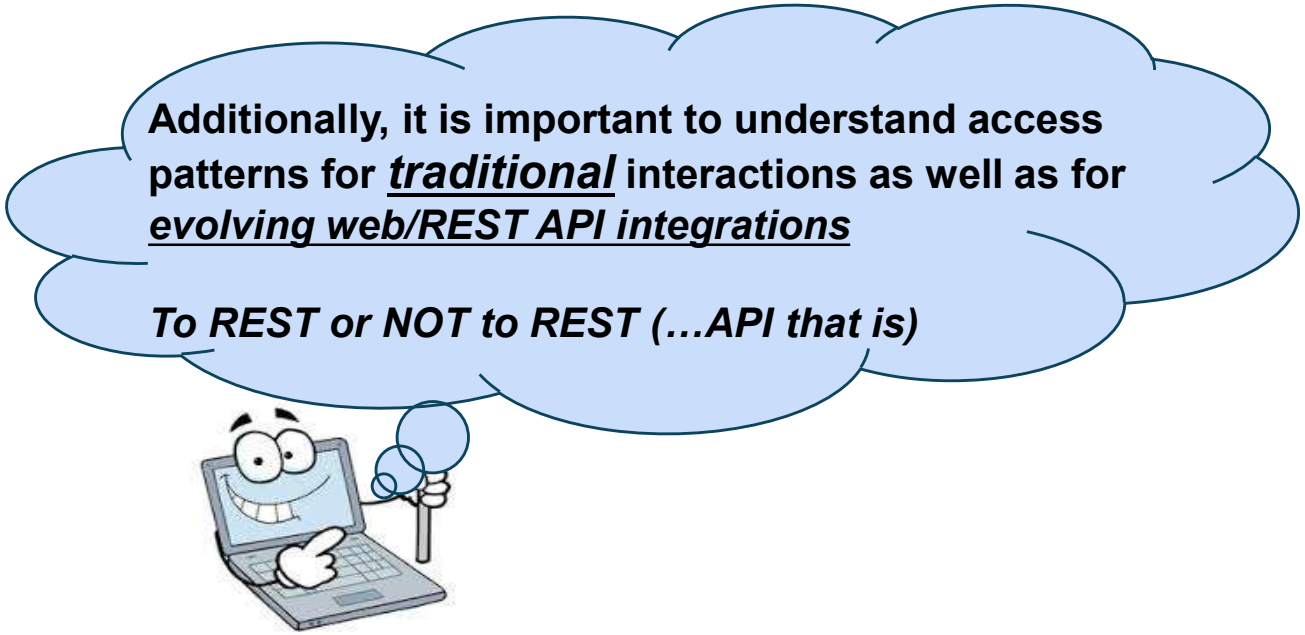
- As with other IMS applications that synchronously call out from the program, the interaction holds IMS resources:
  - **Consider creating a class and specific dependent regions for this program**
    - Fences this program from impacting other transactions that could be classed to run in the same region
    - Facilitates problem determination if the program stays hung

  - **Any database calls that were issued prior to the API request will hold locks**
    - If the IMS database call was an update, then this could have performance implications for other transactions
    - If the IMS database call was a read, consider using the RLSE call which can release IMS DB read locks

API Requester Recommendations …

- Ensure that during ***testing,*** BAQVERBOSE is turned on in the dependent region
  - Point to the LE options member from the CEEOPTS DD
  - Provides information about unusual abends (e.g., S01F) or hangs which can occur

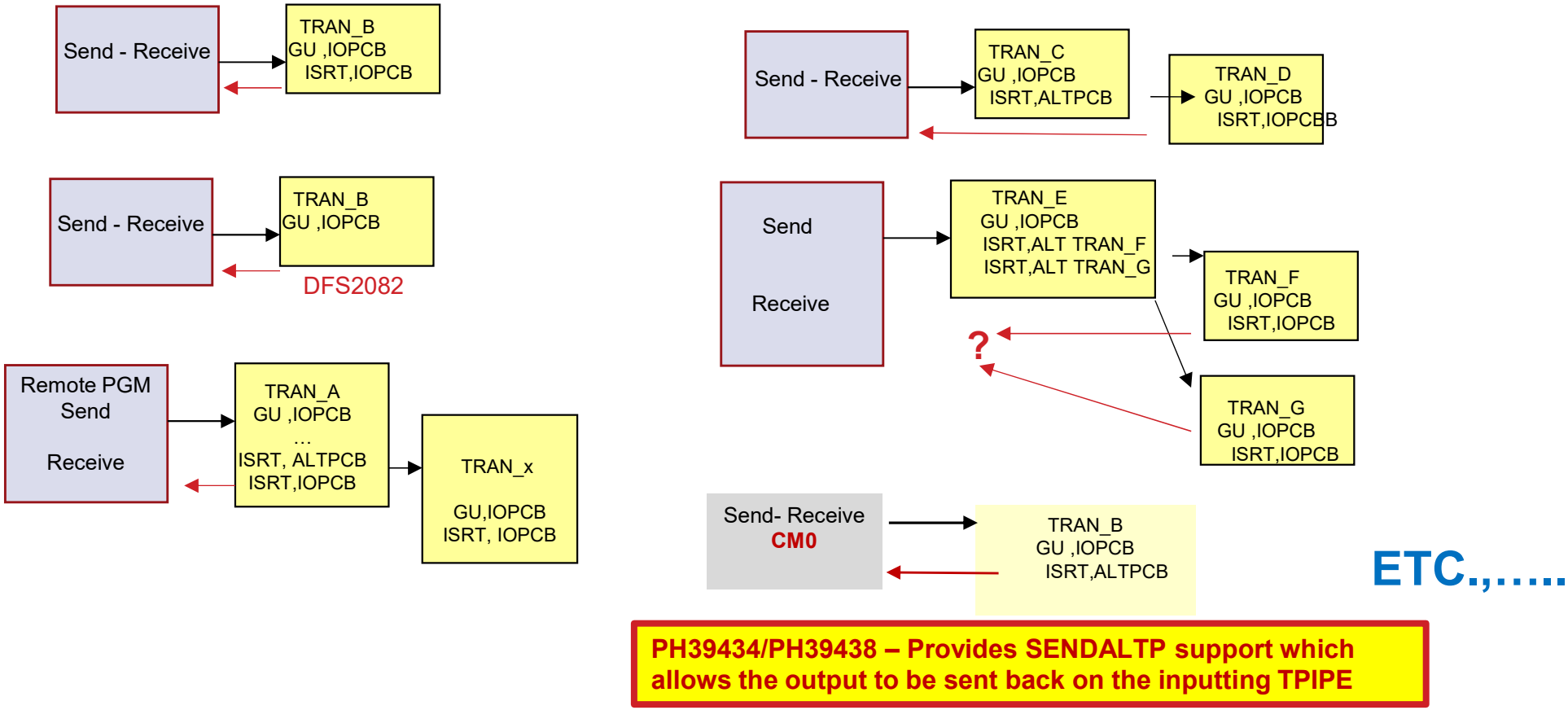




- And for existing support for ***inbound access***
  - Leveraging what IMS has had for a while
  - Taking advantage of what is new

Inbound Access - Patterns

- Existing application integration patterns (variations on a theme)…
  - **Considerations for evolving technologies (APIs)**
    - **E.g., Access through IMS Connect**
      - *CM1 (IMS send-then-commit) and CM0 (IMS commit-then-send)*



How Simple or Complex is the IMS transaction interaction?

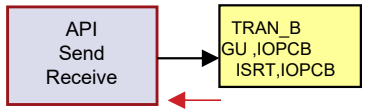
Ensure that someone who knows the architecture of the application is involved in the process

Otherwise you can run into problems that could/should have been avoided

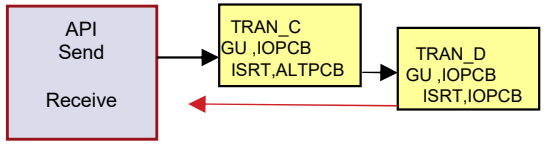
IBM

Simple

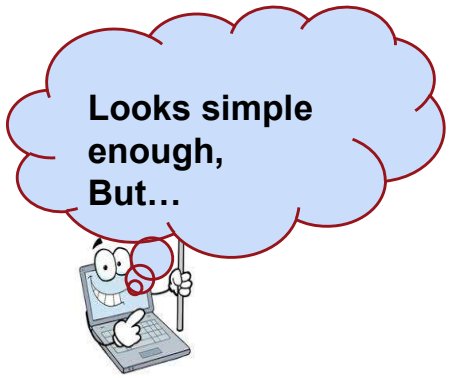
- Considerations – The IMS (Service) Provider uses CM1 Send-Commit (tpipe is the port number)
  - *Can use CM0 Commit-send*
- **Simple interactions**
  - Straightforward API implementation



Input and output layout for the Tool are from TRAN_B



Input layout for the tool is for TRAN_C
And output layout is from TRAN_D



- **AND!**
  - *IMS transaction is non-conversational*
  - Cobol/PLI programs provide the input/output message layouts (copybooks or include files)
  - Any program-to-program switches are done with
    - *Non-express ALTPCBs*
    - *Final switched-to transaction responds to the IOPCB*
    - *No spawning of multiple transactions – switching is simply from one transaction to another*

Complex – Example 1

- **What could cause complexity in this environment?**
  - No available copybooks to provide the toolkit to build the services?
    - *Analysis will need to be done of the input/output IMS messages to **create the layouts***
  - What about having to determine the flow of the application business logic when multiple transactions are involved in ***program-program*** switches?
    - *The exposure to REST API implementation means that the **input layout needs to be tied to the final output** to determine which output message from a switched-to transaction matches the original input message*
    - *IMS application architects should be involved to determine this flow*
- Possible resolutions:
  - Execute the transaction flow and analyze
    - 01/03 log records
      - *Run DFSERA10 / DFSERA30 and assemble the IMS Log record DSECTs*
    - Or: take a trace and analyze the trace records
    - Or use tooling

Log Record Analysis

1. Run DFSERA10 / DFSERA30

```
OPTION   PRINT OFFSET=5,FLDTYP=X,VALUE=01,FLDLEN=1,COND=E,EXITR=DFSERA30
OPTION   PRINT OFFSET=5,FLDTYP=X,VALUE=03,FLDLEN=1,COND=E,EXITR=DFSERA30
```

To get:

```
-03 RECORD -
00000000 000000   022F0000 03C38210  040000D1 040000D1   01DE0000 C9C9D4E2  40404040 D4C64F35  *.....CB....J...J....IIMS    MF³.*
00000020 000020   C9A40C53 C9C9D4E2  40404040 D4C64F35   C9A40C53 00000000  00000000 00000000  *IU..IIMS    MF³.IU.............*
00000040 000040   00408100 C8000000  00000000 00000000   00000000 00000000  00000000 00000001        *. A.H.........................*
```

2. Assemble the IMS log record DSECTs to get the record layout

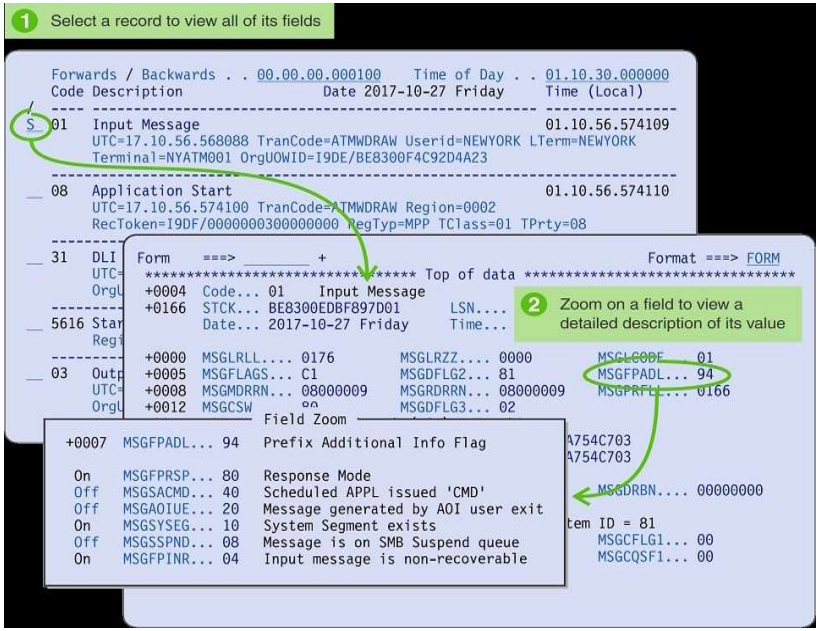
ILOGREC ID=

```
110+MSGLRZZ  DS              H          ZZ FIELD RESERVED FOR QSAM
111+MSGLCODE DS    X          LOG CODE (01 OR 03)
113+MSGFLAGS DS    X          MESSAGE FLAGS
114+MSGFFRST EQU  X'80'       FIRST RECORD OF MSG, FULL PREFIX
115+MSGFLAST EQU  X'40'       LAST DRRN (RECORD) OF MESSAGE
 116+MSGFCANC EQU  X'20'      MESSAGE CANCELED
117+MSGFNRQU EQU  X'10'       NON-RECOVERABLE QUERY MESSAGE
118+MSGFQNR  EQU  X'0F'       L/O NIBBLE HAS LOGICAL QUEUE NO....
...
```

3. Use the DSECT layout to analyze the log records

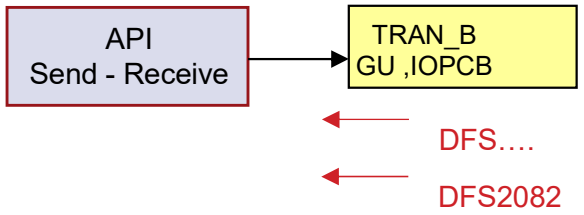
OR:

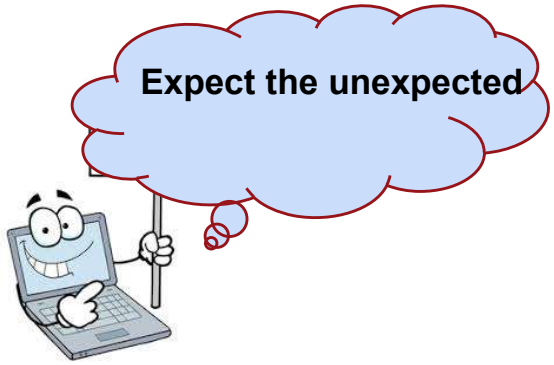
Use a tool like IMS Problem Investigator to access and analyze the IMS log records in a more user-friendly fashion



Complex – Example 2

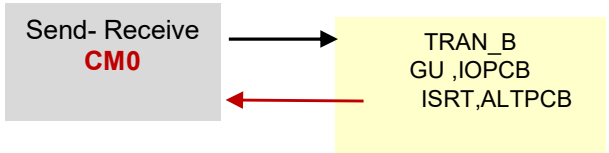
- If there are problems in the IMS environment, or the IMS transaction abends, or the transaction does not reply to the IOPCB





  - The REST API may receive an unexpected response
    - If the target transaction does not ISRT to the IOPCB or ALTPCB
      - *The API requester will get a DFS2082*
    - Or, If the target transaction experiences an error/problem on the IMS side
      - *The API requester will get a DFS... message*

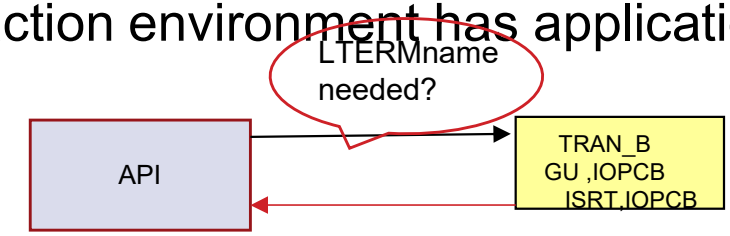
- Solutions:
  - Design the REST API to expect either the output response from the transaction or an IMS DFSxxxx message
  - Implement SENDALTP support



PH39434/PH39438 – Provides SENDALTP support which allows the output to be sent back on the inputting TPIPE

Complex – Example 3

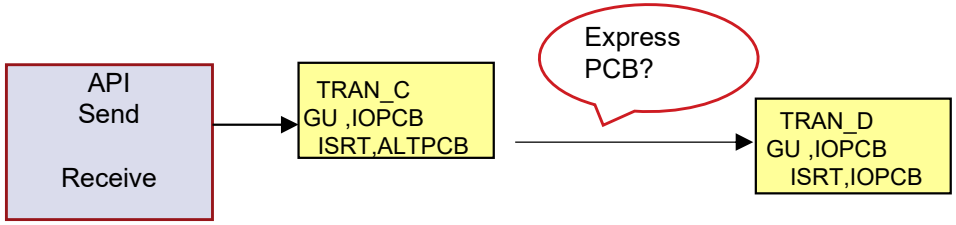
- The IMS transaction environment has application security or needs an LTERMname

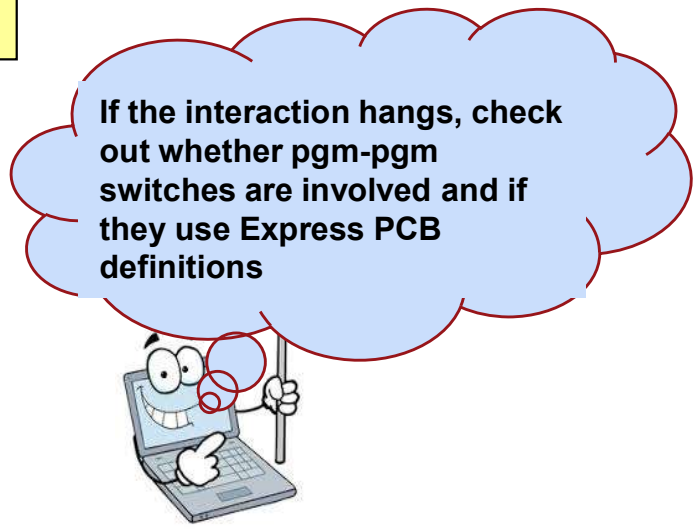


- LTERMnames may be required by the application(based on older 3270 environme
  - Determine if a generic LTERMname can be used
    - *Define it in the connection profile in z/OS Connect*
  - Otherwise, some possible solutions:
    - *Define the value needed for LTERMname in the REST API input message (e.g., LLZZ trancode data **overridename**)*
      - *Specify a dummy value in the LTERMname value of the connection profile*
      - *Write an IMS OTMA exit (DFSYIOE) or IMS Connect exit (HWSJAVA0) to recognize the dummy value from the connection profile and:*
        - *Remove the overridename from the input message*
        - *Use the value to pass it in as an LTERMname*
    - Other possibilities would depend on the specific customer environment

Complex – Example 4

- The IMS transaction program-program switch uses an Express PCB



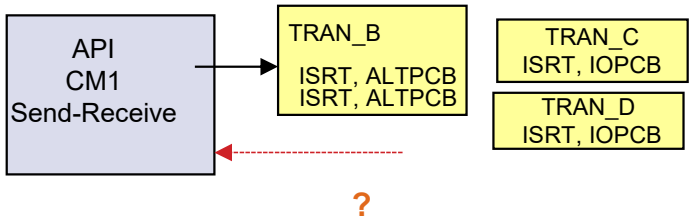


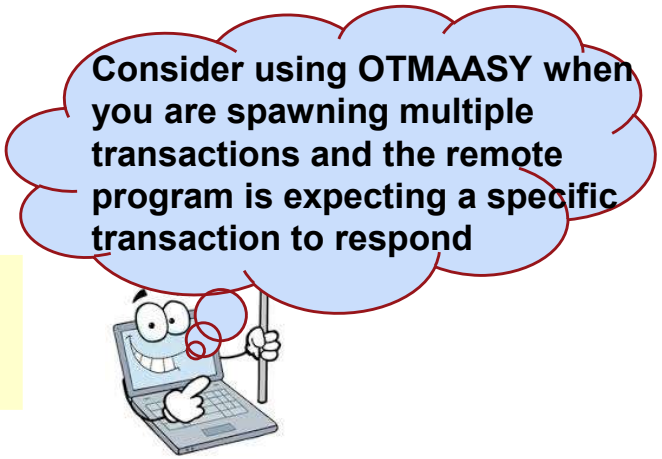
  - The REST API request will hang

- Solution:
  - *Change the Express PCB to Non-express PCB*

Complex Example 5

- ***Consideration – Which transaction responds to the outstanding Receive*?**





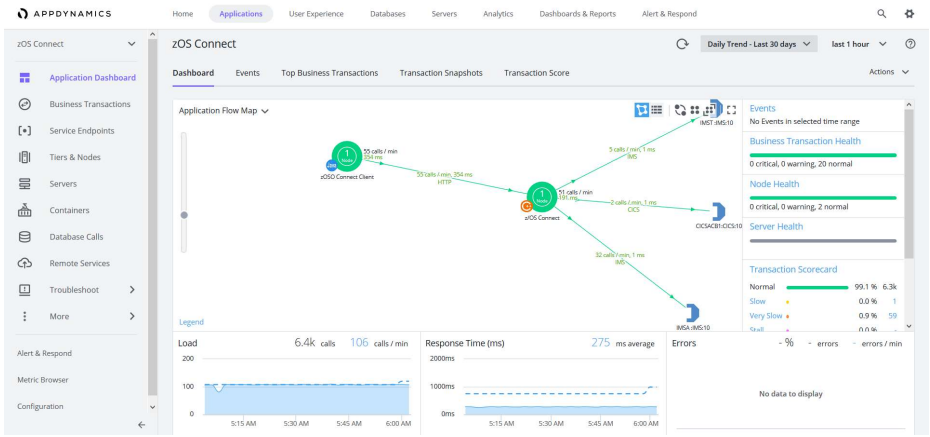
- **OTMAASY = Y | N | S**
  - Determines which transaction is switched to synchronously (eligible to reply to the synchronous request)
    - **Y** – First RESPONSE program scheduled after the switching transaction ends is scheduled synchronously
    - **N** – First program (RESPONSE or NONRESPONSE) scheduled after the switching transaction ends is scheduled synchronously
    - **S** – The first message ISRTed to a non-Express ALTPCB will be scheduled synchronously
  - Check the documentation in the IMS manuals for more detailed explanations and actions:
    - *https://www.ibm.com/support/knowledgecenter/en/SSEPH2_14.1.0/com.ibm.ims14.doc.sdg/ims_proc_parms_otmaasy.htm*
  - To control the response when OTMAASY=Y is used
    - Define the target transaction as response-mode
      - *Or, create a new TRANSACTion for the application (target of the pgm-pgm switch) as response-mode*
        - *DFSMSCE0 exit routine can change the destination trancode without changing the program (DFSMSC0 discussed later)*

Overall Recommendations

- On the IMS side, ensure that the application architecture is understood
  - Determine if the application is ‘simple’
  - Anticipate complexity or potential show-stoppers
    - Determine if it would be easier to write/modify the IMS applications
      - *Possibility: Orchestration transaction that uses IMS synchronous Program Switch support to invoke existing transactions*
- For problems,
  - Log records, traces, or tools

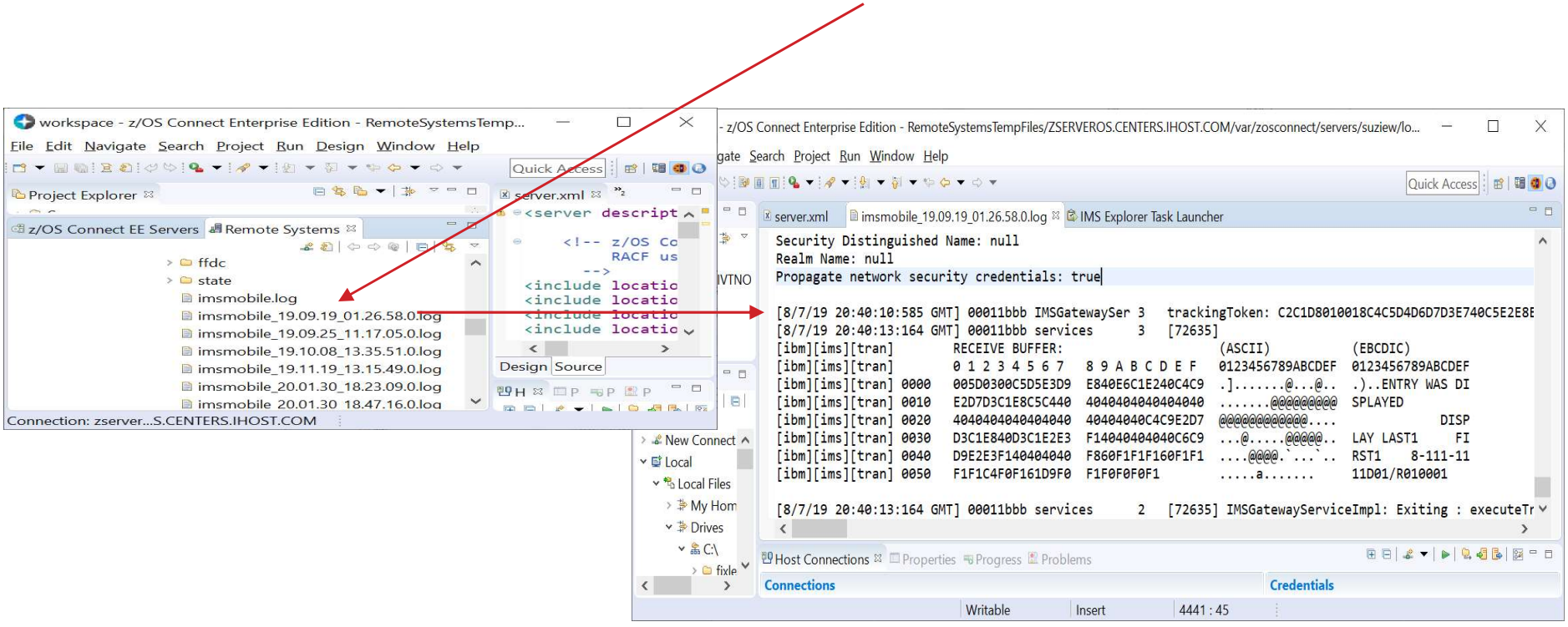
Tooling

- IMS Connect / IMS / Application
  - Journaling and control: IMS Connect Extensions (IMS CEX)
  - Problem determination: IMS Problem Investigator (IMS PI)
  - Performance: IMS Performance Analyzer (IMS PA)
    Omegamon for IMS
  - Application workflow: ADDI
- z/OS Connect
  - Omegamon for JVM – can monitor z/OS Connect API utilization
- End-to-End tracking
  - IBM z APM Connect - Can be used to determine/ target problem areas



Overall Recommendations …

- For problem determination, turn on tracing in z/OS Connect server.xml file
  - <logging maxFileSize="20" maxFiles="10" traceFileName="imsmobile.log" traceFormat="BASIC" traceSpecification="com.ibm.ims.gateway*=FINEST:com.ibm.ims.zconnect.provider*=FINEST"/>

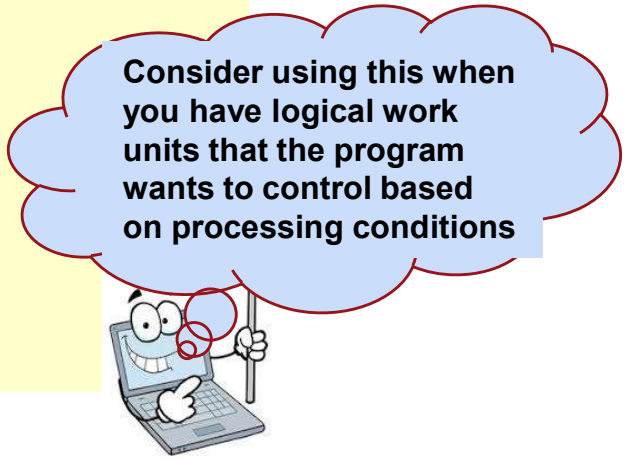


And leverage what IMS has had for a while

DL/I Calls to Consider

- SETS/SETU and ROLS
  - Provides intermediate backout points for IMS resources
    - SETU is accepted even if there are unsupported PCBs (DEDB or MSDB) or a call has been made to an external subsystem (DB2)
    - ROLS with a token backs out to the defined SETS/SETU point
      - *IMS resources only, e.g., IMS FF databases and non-express messages*
    - Allows the IMS application program to have greater control for error processing

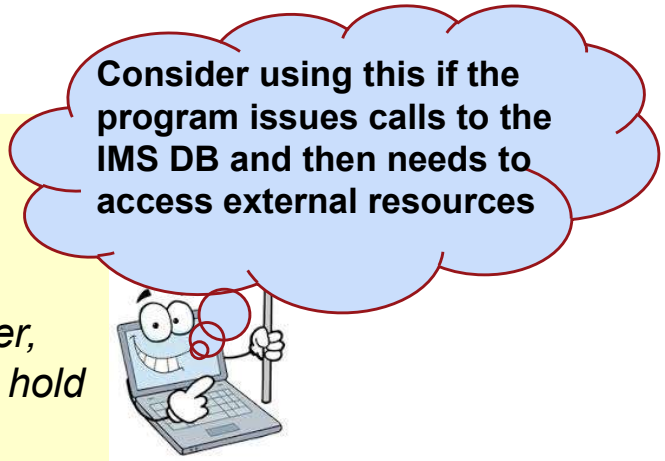
- Example:
  - *SETU tokena*
  - *Program updates IMS resources (e.g. database calls and ISRT altpcb)*
  - *Program issues DB2 calls and gets an error; or attempts*
  - *a callout to an external resource that gets timed out;*
  - *or….*
  - *ROLS tokena backs IMS resources out to positioning*
  - *of previous SETU*
  - *Program retains control and continues processing*



DL/I Calls to Consider…

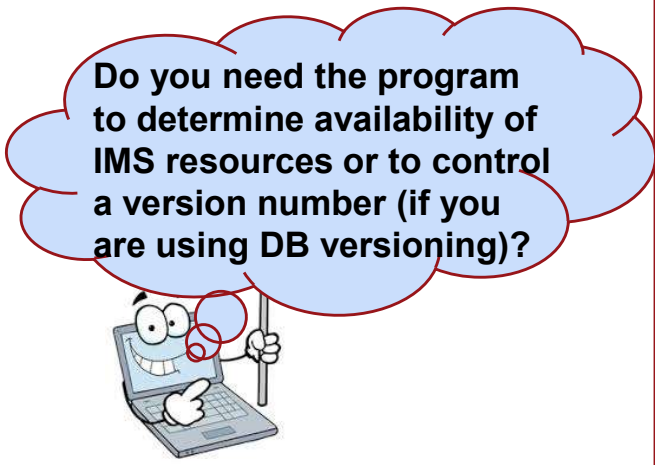
- **RLSE**
  - Provides the ability to release IMS DB read locks
    - Update locks are not released
    - Fast Path DBs - releases all locks held for unmodified data
    - Full Function DBs - release locks held by the DB PCB that is referenced
  - Note: after the RLSE call, all database position information is lost

- Example:
  - *Program reads IMS DBs*
  - *Prepares message for callout with a wait (ICAL,API Requester, MQPUT_MQGET…) which will hold the region occupied and hold locks*
  - *RLSE releases the read locks and positions on database*
    - *Keep information on positioning if needed*
  - *After the callout message is returned, re-position on databases if needed and continue processing*



DL/I Calls to Consider…

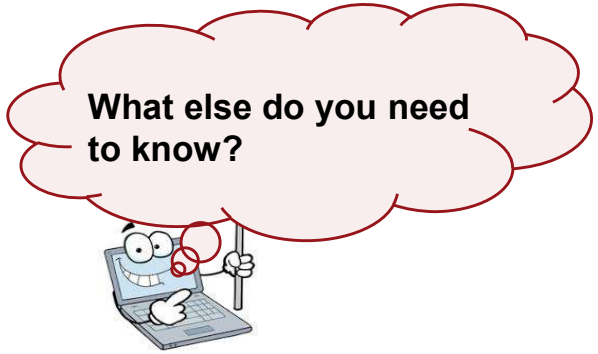
- **INIT DBQUERY | STATUS GROUPx | VERSION**
  - **DBQUERY –** automatically issued at program entry and initializes the status code for each DB PCB
    - Program can immediately determine status of database
      - *Code can determine what to do based on these status codes*
        - *Blank = available*
        - *NA = Not Available*
        - *NU = Not Updatable*



  - **STATUS GROUPx (x= A or B)**
    - Prevents U3303 abends when database is unavailable
      - *INIT STATUS GROUPA allows the program to retain control when an attempt is made to access an unavailable DB (BA status code returned)*
      - *INIT STATUS GROUPB allows the program to retain control as in GROUPA plus the additional condition of an IMS deadlock (BC status code returned)*
    - Rather than abending, applications can do alternate processing
  - **VERSION *(dbname=version)***
    - Applies when database versioning is enabled
      - *Allows a program to override the versions specified in either the PCB or the version number determined or defaulted by IMS*

DL/I Calls to Consider…

- **INQY (blanks| DBQUERY| FIND| ENVIRON | LERUNOPT| MSGINFO | PROGRAM)**



- Provide the program with information using subfunctions:
  - **blanks -** information related to the PCB (I/O or ALT), including output destination type and location, and session status
    - *For example, OTMA information incudes: TPIPE, member name, userid, userid indicator, group, synch level, msg synch level*
    - *If the destination for an ALTPCB is a transaction, information includes: location (remote, local, dynamic), status (started or stopped), destination PSB, destination program or session status)*
    - *…*

DL/I Calls to Consider…

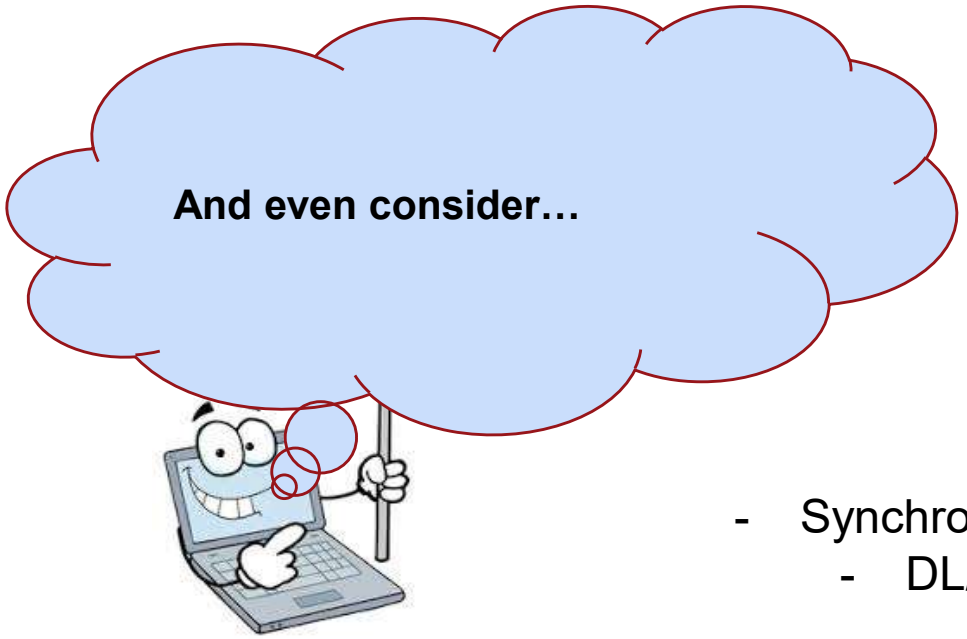
- **INQY …**
  - **DBQUERY** - information regarding the data for each PCB (similar to INIT DBQUERY) and updates status codes in the DB PCB
  - **FIND** – is usually issued after an INQY DBQUERY to get a PCB address in order to analyze the PCB status code and determine if NA/NU was specified
  - **ENVIRON/ENVIRON2** - information regarding the current execution environment
    - *IMS ID, IMS release level, control region type, application region type, region identifier, application program name, PSB name, transaction name, userid, group name, status group indicator (whether GROUPA or GROUPB was requested), address of recovery token, address of application parameter string, shared queues indicator, userid of the address space, userid indicator, RRS indicator, catalog enablement indicator*
    - *ENVIRON2 includes all the information from ENVIRON plus IMS installed level, function level, function enabled bitmap, primary language enclave addressing mode, e.g., 64 bit, language environment addressing mode for JVM, IMS managed ACBs enablement indicator*
      - *Note that the address of the application parameter string allows the program to access the APARM= parameter if it is coded in the execution parameters of the dependent region JCL.*

DL/I Calls to Consider…

- **INQY …**
  - **LERUNOPT** – address of string of LE runtime option overrides
    - IMS checks to see if there are any overrides applicable to the caller based on the specific combinations of transaction name, lterm name, userid, or program
      - *The LE overrides are used by the IMS supplied CEEBXITA exit, DFSBXITA, to allow dynamic overrides for LE runtime parameters*
  - **PROGRAM** – the application program name
  - **MSGINFO** - information regarding the current input message
    - Version number (default value is 1), and origin IMS ID
    - In IMS 15: version number will be 2 if the network security credential (distributed client's end user identity) has been passed in with the input message
      - *Network id – which can be up to 246 bytes. For example, it can be a Distinguished Name (DN) which is fully documented in the X.500 series of standards.*
        - *Example: CN=Jane Doe,OU=Sales,DC=IBM,DC=COM*
      - *Session id – which can be up to 254 bytes. It can be a realm or registry. For example, it can be a Domain name which is name of security database used to authenticate the distributed user.*
        - *Example: LDAP server ldaps://us.svl.ibm.com*

DL/I Calls to Consider…

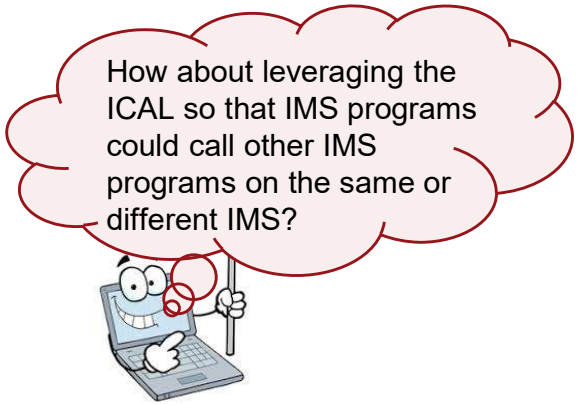
- Remember: the AIB (Application Interface Block) provides a way for the application to communicate with IMS
  - When a function does not use a PCB
    - Many subfunctions (e.g., INQY ENVIRON) use the name IOPCBbbb
  - When the application does not have a PCB address
    - Access to the IMS resource in the PSB is defined by PCB name or label
      - *This means that you do not need to know the relative PCB number in the PCB list.*
      - *And the program can make calls on PCBs that do not reside in the PCB list*
        - *LIST=NO on the TYPE=DB PCB statement*
- The calls previously discussed can all use the AIB interface
  - The INQY call must use the AIB

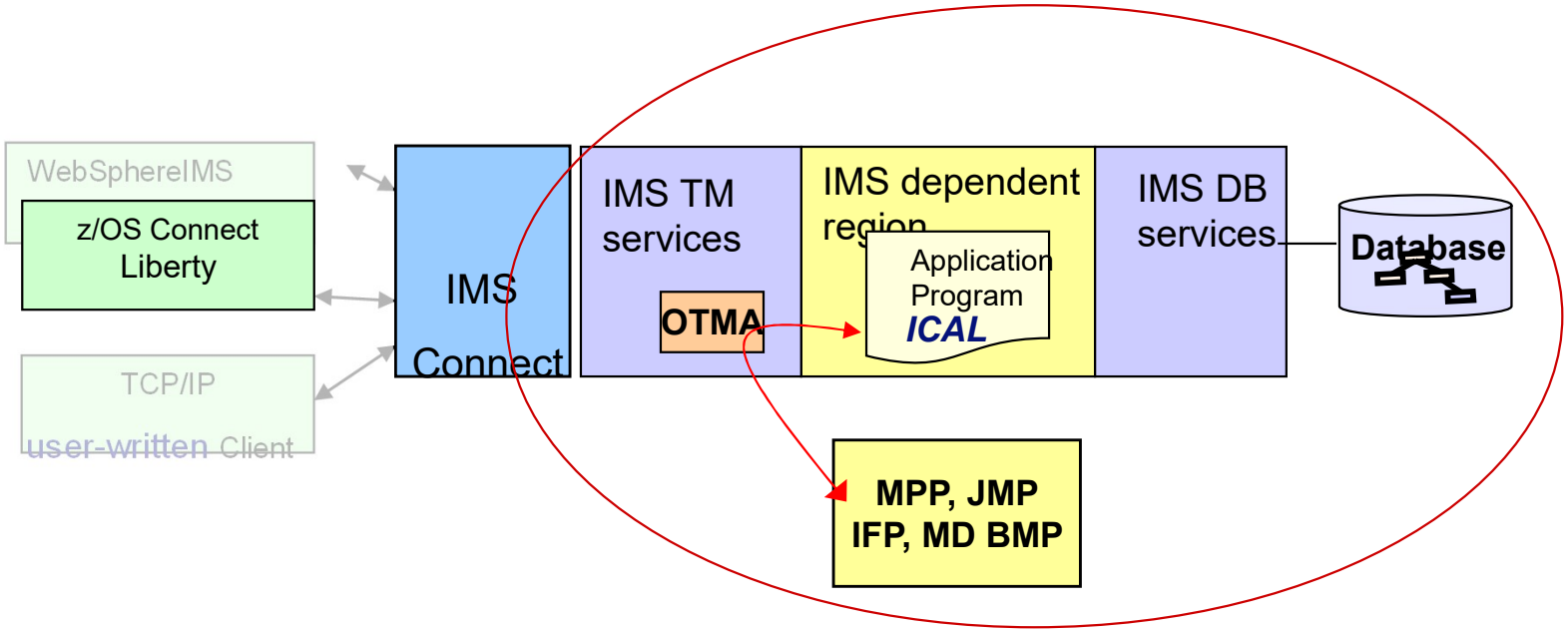


- Synchronous Program Switch
  - DL/I ICAL to another IMS program
- Synchronous Callout
  - DL/I ICAL to an external resource
- z/OS client web enablement toolkit
  - HTTP calls directly from an IMS Application
- Transaction Orchestration
  - If the IMS application program (may have been invoked by an API) also needs to access a remote process **asynchronously** as part of the transaction

Synchronous Program Switch

- 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 UOW

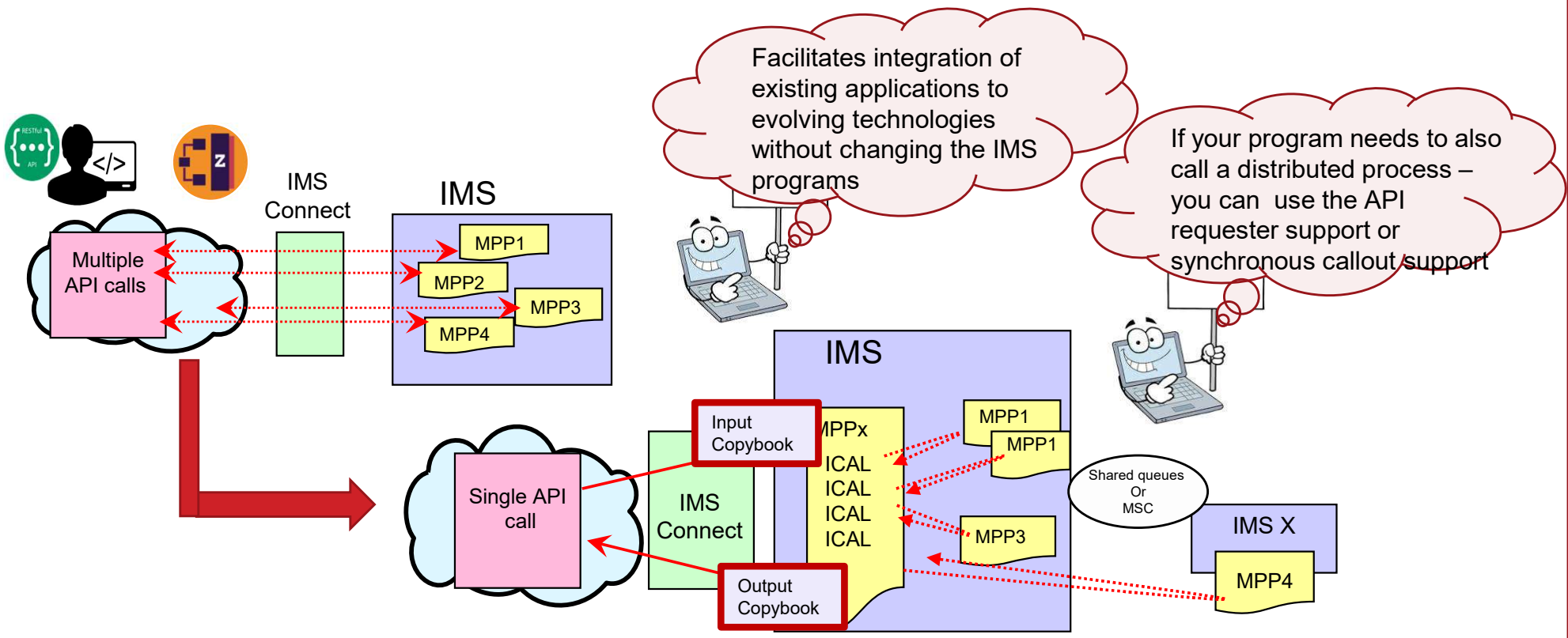




Synchronous Program Switch…

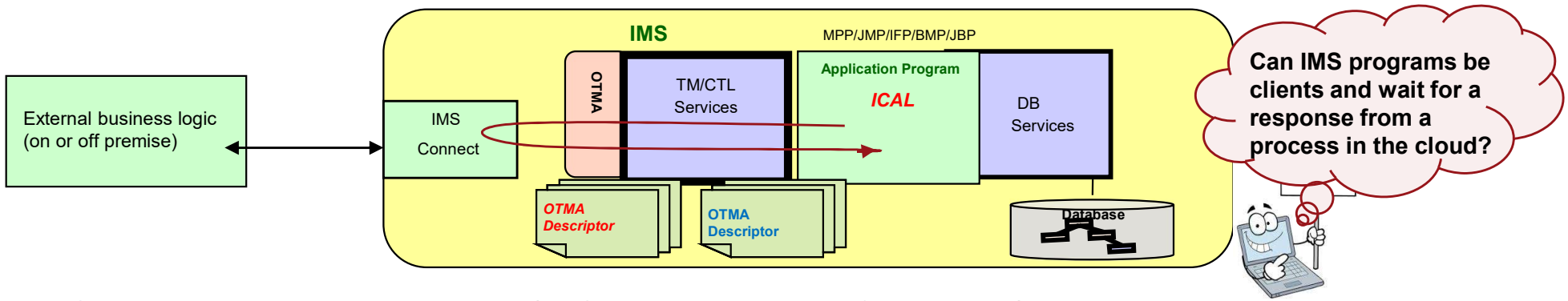
- Which can
  - Enhance the IMS application infrastructure
    - To provide an internal service flow of IMS transactions to complete a business process
      - *In the same IMS or a different IMS*

  - And even implement a process server or broker application inside IMS
    - Which could reduce unnecessary network traffic when accessing multiple applications in the same IMS or IMSplex



Synchronous Callout – DL/I ICAL

- **ICAL Positions IMS to be a full partner in integrated hybrid clouds**
  - IMS transactions can access a service outside IMS and wait for a reply within the same unit of work



  - IMS creates a **correlation token** (IMS app is unaware of this token) but the external client must pass this token back to IMS so that the response can be sent to the correct instance of the executing program
  - Leverages the AIB interface

Call AIBTDLI USING **ICAL**,*aib*,REQ_area,RESP_area

AIBSFUNC = SENDRECV or RECEIVE
AIBRSNM1 = 8 byte OTMA Descriptor name
...

the message destination and how IMS is to handle the message

OTMA Destination Routing Descriptor

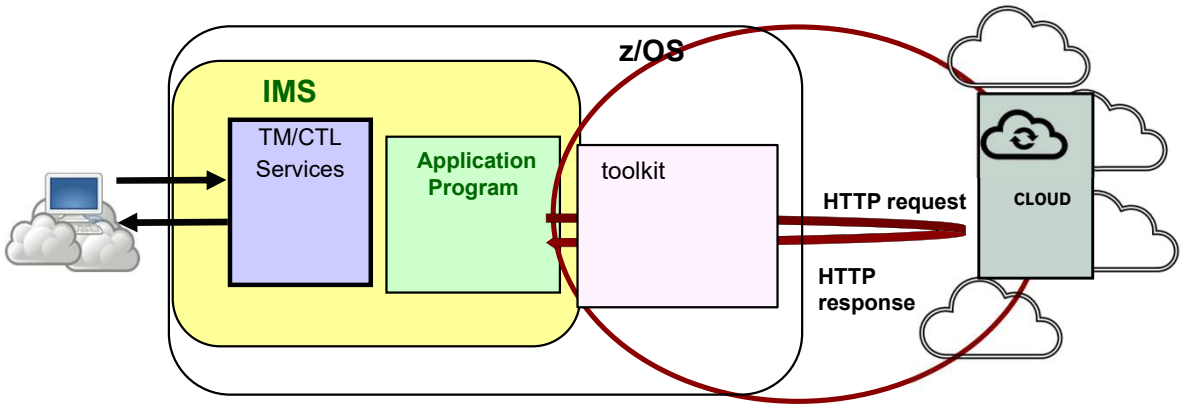
- DFSYDTx member of IMS.PROCLIB
  - TYPE: Destination type
  - TMEMBER: OTMA Target Client
  - TPIPE: Destination Name
  - SMEM: YES|NO
  - SYNTIMER=timeout
    - Note ICAL overrides this value

Type -2 Command:
Can update the descriptor values

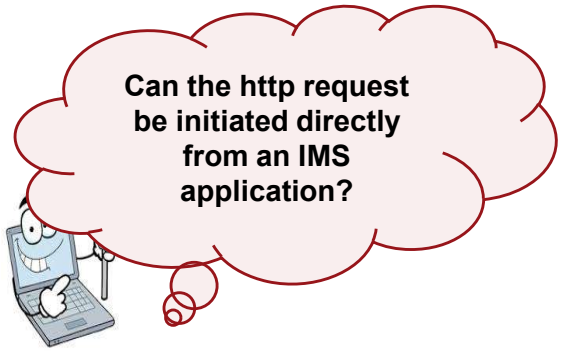
UPDATE OTMADESC NAME(OTMASYN) SET(SYNTIMER(5000))

z/OS Client Web Enablement Toolkit

- IMS Applications can be a RESTful client and ***directly*** initiate a request to a web server



- Web enablement toolkit
  - Provides a set of lightweight application programming interfaces (APIs) that enables native z/OS programs to participate in modern web services applications
  - Part of the base z/OS operating sysem



Request body | Headers

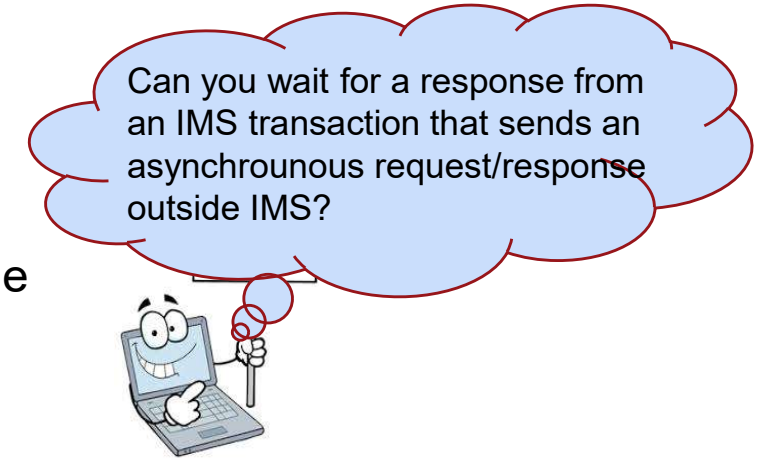
- A client makes an HTTP request to a server
  - GET (read existing resource)
  - PUT (write/update existing resource)
  - POST (write new resource)
  - DELETE (remove existing resource)
  - Request Headers
  - Request Body (PUT and POST)

Headers | Response body

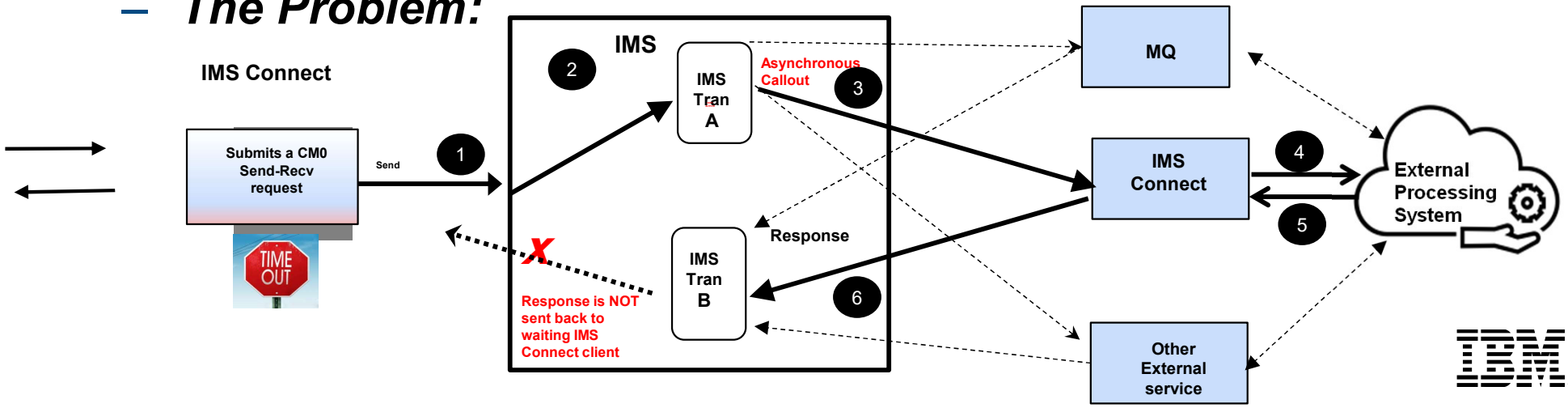
- Server replies with an HTTP response
  - Status (1xx, 2xx, 3xx, 4xx, 5xx)
  - Response headers
  - Response body (most requests)

And Something New: Transaction Orchestration

- **What is this new function?**
  - Ability for an **asynchronous** callout response from an external system to be identified and routed to the original waiting CM0 send-receive IMS Connect client



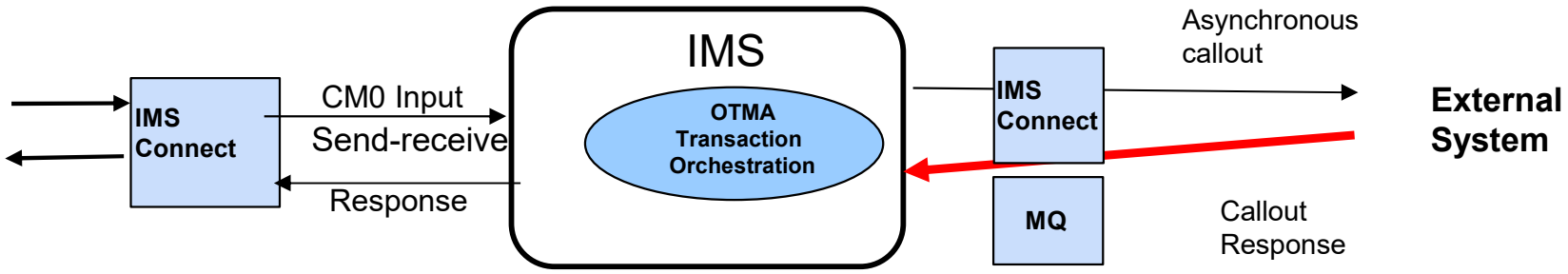
  - ***The Problem:***



And Something New: Transaction Orchestration …

- The Solution
  - *APAR PH51897/ PTFs UI96019, UI96138 for OTMA*
  - *APAR PH57295/ PTF UI98354 for IMS Connect*

– the OTMA transaction orchestration enhancement provides a way for OTMA to identify and correctly route an external service response from IMS to a waiting IMS Connect client
  - *When activated for the original CM0 send-receive message, the response from the external processing system can be routed back to the waiting IMS Connect client*
  - *TRANORCH=Y activates the function*
    - *OTMA client descriptor*
    - *IMS Connect configuration file*



Summary--- and the Message

- The intent of this presentation was to provide a focus on the IMS application programming functionality in a technical environment
  - That continues to evolve and has even greater requirements for cloud and mobile integration of your existing and proven assets

By leveraging

The **OLD** which is still very applicable

e.g.,

- DL/I calls, e.g., SETS/SETU-ROLS, RLSE, ICAL
- IMS Parameters – e.g., OTMAASY
- Exit routines

The **NEW** which continues to expand

- e.g., mobile support and the new callout to external REST APIs functionality