CICS Connectivity and Networking

Ian J Mitchell
IBM System Z Middleware CTO

ianj_mitchell@uk.ibm.com

Session 15886, Thursday 7th August 2014
Disclaimer

IBM’s statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM’s sole discretion. Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision. The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract. The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.
Agenda

- Connectivity landscape review
- IPIC enhancement in CICS TS v5.2
- CICS Transaction Gateway v9.1
- z/OS Connect
- A little “gotcha”
Adding the Web container to the Connectivity landscape
Adding the Web container to the Connectivity landscape

WMQ Triggering & DPL Bridge

Complete your session evaluations online at www.SHARE.org/Pittsburgh-Eval
Adding the Web container to the Connectivity landscape

Native WS & Axis2

WMQ Triggering & DPL Bridge

HTM/REST

Service enablement layer

WMQ
CTG
WS
REST
Adding the Web container to the Connectivity landscape

Core Application Layer

Service enablement layer

BMS TR

DPL

WMQ CTG

WS REST

WMQ CTG

HTML/REST

(TN)3270

WMQ CTG WS REST

HTML/REST

WMQ Triggering & DPL Bridge

Native WS & Axis2

IPIC & EXCI

WOLA TRUE

Complete your session evaluations online at www.SHARE.org/Pittsburgh-Eval
Adding the Web container to the Connectivity landscape

- Web Container (built on Liberty)
- WMQ Triggering & DPL Bridge
- Native WS & Axis2
- IPIC & EXCI
- WOLA
- TRUE
- Core Application Layer
- Service enablement layer
- BMS
- TR
- (TN)3270
- HTML/REST

Complete your session evaluations online at www.SHARE.org/Pittsburgh-Eval
High Availability features

- Single entry point
- Removal of single point of failures
  - Continuous availability
  - Resilience to failures
- Upgrades and maintenance operations do not effect end users
- Clients unaware of complexities of systems they are using
IPIC HA Requirements

- Provide single point of access to a cluster of CICS TS regions via an IP network, to offer high availability by removing single points of failure.
- The solution should not be limited to a single technology and so will support
  - TCP/IP port sharing
  - IBM Communications Server's Sysplex Distributor function based around the use of Dynamic Virtual IP Addresses (DVIPA)
  - Other non-IBM connection balancing technologies
- Built from extensions to the IP Interconnectivity (IPIC) function that is provided from CICS TS
- CICS to automatically resolve UOW affinities following the re-establishment of a failed connection.
- Clients can connects back to the cluster, when they have outstanding UOW affinities with specific cluster regions when that region is not currently available. CICS remembers which UOWs are associated with which server and will attempt to resolve them when a client releases and reacquired its connection to the cluster.
Terminology *(new ones in blue)*

- **AIVIPA** – Application Instance Virtual IP Address, defines the **Specific Entry Point** that an **HA CICS Region** is using.
- **Client CICS Region** – CICS region running outside of the **HA Cluster**, but connecting to regions in it.
- **Distributor Stack** – stack containing a **DVIPA** that is used to intercept connection requests and select a CICS region that is listening on a common **PORT** on the DVIPA's address.
- **DVIPA** – Dynamic Virtual IP Address – as provided by Sysplex Distributor, and used as the **Generic Entry Point** to an **HA Cluster**.
- **Generic Entry Point** – host name and port, defined with a **DVIPA**, and advertised by a **Distributor Stack**, or that used in a **Port Sharing** environment, as the entry point to an **HA Cluster**.
- **Generic TCPIPSERVICE** - one that listens on a **DVIPA** or a shared port.
- **HA Cluster** – set of CICS regions, running in a sysplex, listening on a common **DVIPA**, or on a shared port.
- **HA CICS Region** – CICS region that is a member of an **HA Cluster**.
- **HOST** – host name/IP address that a **Stack** publishes
- **Local Stack** – one that is running on the same LPAR as a CICS region
- **PORT** – TCP/IP port
- **Port Sharing** – CICS regions running on a single LPAR listening on a common IP address and port number
- **Specific Entry Point** – the host name and port number, defined with an **AIVIPA** on the local stack which a CICS region listens on for connection requests.
- **Specific TCPIPSERVICE** – one used to listen on the **Specific Entry Point** of an **HA CICS Region**.
- **Stack** – An instance of a TCP/IP stack running on an LPAR
Components of Sysplex Distributor

- CICS-CL
- Sysplex
- z/OS LPAR A
  - CICSA
  - Target Stack A
  - Advertised IP address (DRVIPA) 1.2.3.4
- z/OS LPAR B
  - CICSB
  - Target Stack B
  - Advertised IP address (DRVIPA) 1.2.3.4
- Each TCPIP service binds to the DRVIPA 1.2.3.4

- CICSD
- Distributing Stack
- CoUpling Facility
- Advertised IP address for cluster address
- 192.168.0.1 xcf
- 192.168.0.2 xcf

Complete your session evaluations online at www.SHARE.org/Pittsburgh-Eval
All regions run in a single LPAR, and listen on same IP address and port via a common stack.

CICS A connects using this IP address and port and the local stack assigns the request to region 8.

If connectivity is lost then CICS A needs to connect back to CICS 8 to resolve any UOW affinities.
Solution based around Sysplex Distributor

- DVIPA identifies the Generic Entry Point to the cluster, and resides on the Distributor Stack.

- AIVIPA identifies the Specific Entry Point of an HA Region, and moves with that region within the Sysplex. If no need to move regions between LPARs then VIPAs can be used.
Local Stack

1.2.3.4
CICS A
APPLIDA

IPC

Name -CIC1
Applid - blank
ip_addr
-1.2.3.99
Port - 8999
Tcpipservice – TCPIPSA
HA - YES

Local Stack

1.2.3.4
Distributing Stack

ip_addr 1.2.3.99
CICS 1
APPLID1

TCP/IP SERVICE

Name – TCPIPS1
Port – 8888
ip_addr – 1.2.3.4

TCP/IP SERVICE

Name – TCPIPSGR
Port – 8999
ip_addr – 1.2.3.99
SEPSERVICE = TCPIPS1

Primary CAPEX Request:

Primary CAPEX Response

Secondary CAPEX Request:

Secondary CAPEX Response

Secondary CAPEX Request:

Secondary CAPEX Response

Primary CAPEX Request:

Primary CAPEX Response

Cluster Connect Request:

Cluster Connect Response

Resync messages

Complete your Session evaluations online at www.SHARE.org/Pittsburgh-Eval
HA solution using TORs to balance workloads
Resynchronisation Affinities

- CICS user tasks uses RM links to indicate that recoverable work has been done over a connection.
- These links persist on the system log for UOWs that are either in-doubt or committed and awaiting forget should a network or region failure occur during syncpointing.
- Each region has its own system log that has a unique log name. Log names are exchanged as part of the CAPEX sequence that takes place when a connection is being acquired.
- Resync processing takes place once an IPIC connection is reacquired assuming that the log names of both regions have not changed since the connection was previously acquired.
- If a client region is unable to reconnect to the same region following an outage then any UOWs can only have heuristic decisions applied to them by CICS, or are left for processing later. If left then resource locks could be held until a resync operation takes place later.
Resync operations based around region re-start

- One region fails, and its partner tries to reconnect to it
- The operation fails and so the partner connects to another region in the cluster
- The failed region restarts, and attempts to reestablish the old connection
- This succeeds and the outstanding work is resynchronised.
- The connection between the original regions is then released.
Agenda

- Introducing CICS TG product suite
- CICS TG for z/OS V9.1 open beta
  - Service Enablement
  - Secure connectivity
  - Modern connectivity
The CICS TG Product Suite

Capabilities your developers need
- Mobile
- C / C++
- Microsoft .NET Framework
- Java
- JEE
- COBOL

Scalable integration with your systems
- CICS TG Desktop Edition
- CICS TG for Multiplatforms
- CICS TG for z/OS

Transactional access to your key business assets
- CICS TS for i
- TXSeries
- CICS TS for VSE
- CICS TS for z/OS

Complete your session evaluations online at www.SHARE.org/Pittsburgh-Eval
The IBM® CICS® Transaction Gateway for z/OS V9.1 open beta offering highlights new mobile support within CICS TG which enables new, modern workloads to be quickly and efficiently incorporated into existing CICS systems. In addition, enhancements in connectivity and security are also included, addressing key customer requirements.

You can find the Announcement Letter for CICS TG V9.1 open beta [here](http://www.ibm.com/software/htp/cics/openbeta/).
CICS TG for z/OS V9.1 open beta - video

http://ibm.biz/cicstg91beta
CICS Transaction Gateway V9.1 open beta
Mobile integration, robust connectivity, and strong security options
Service Enablement

Service enablement

Mobile integration with JSON web services

• Dynamic routing of mobile workload
• Shared tech with CICS TS + z/OS connect
• JSON xform from COBOL, C and PL/1
• Full monitoring and statistics
Systems of Engagement Meet Systems of Record

Cloud and API Economy

API Management
DataPower

WebSphere Application Server
MQ
IMS
CICS TS
DB2
zLinux
z/OS
CICS on Multi-platforms

Available as Value Unit Editions

Complete your session evaluations online at www.SHARE.org/Pittsburgh-Eval
Interoperable with Worklight
JSON web services – Overview

Significant new capabilities for CICS TG

- New style of remote client and data representation
  - No client-side IBM code required
  - Active data transformation within the Gateway daemon
  - Service-enablement for any release of CICS server
  - Exploit the high availability and instrumentation features of CICS TG

- Top-down style service enablement
  - Generate COBOL, C, PL/1 language structures from a JSON schema
  - Non-RESTful can be used with COMMAREA of channel programs
  - RESTful must use channel programs (and therefore IPIC)

- Bottom-up style service enablement
  - Generate JSON schema from COBOL, C, PL/1 language structures
  - Target program is not REST-aware, so JSON web service is non-RESTful; COMMAREA or channel programs supported
JSON web services – Overview

**Powered by Liberty, compatible with z/OS Connect and CICS TS JSON support**

- Based upon proven technologies
  - Uses a “private” *embedded* WebSphere Liberty profile within the Gateway daemon for the HTTP server
  - Uses common data transformation components at run-time from CICS TS for z/OS
  - JSON ws-bind files are interoperable with CICS TS, CICS TG for z/OS V9.1 open beta and z/OS Connect solutions
- The JSON web services assistant is included with CICS TG for z/OS V9.1 open beta
  - Uses common tooling components with a simplified interface for CICS TG

Complete your session evaluations online at www.SHARE.org/Pittsburgh-Eval
CICS TG for z/OS – Solution architecture

Application machine

- JEE application server
- JCA Resource Adapters
- Java clients
- ECI v2 C clients
- .NET Framework-based clients
- JSON web services

z/OS

- Protocol Handler
- Gateway daemon
- EXCI module
- JCA Resource Adapter
- IPIC
- WebSphere Application Server

Remote clients

Local clients
New protocol handlers for HTTP and HTTPS

- Define at most ONE each of HTTP and HTTPS protocol handlers
- Both are compatible with TCPIP port sharing capabilities:
  - SHAREPORT
  - SHAREPORTWLM with Gateway daemon health reporting
  - Sysplex Distributor
- New configuration sub-sections HTTP, HTTPS within the GATEWAY section
- No timeout values to define
  - Defined at the web service level
- No user security attributes to define
  - Common SSL resources, CICS connections define authentication
New pool of “listener” threads

- New pool of “listener” threads
  - Logically equivalent to Connection Manager threads
  - Shared between HTTP and HTTPS protocol handlers
  - Defined in the GATEWAY section by `maxhttpconnect`, e.g. `maxhttpconnect=100`

- Define at most **ONE each** of HTTP and HTTPS protocol handlers

- The HTTP thread pool is shared between the HTTP and HTTPS protocol handlers
  - As Connection Manager threads are shared between the tcp and ssl protocol handlers, if both are defined

- The Worker thread pool is shared by **ALL** protocol handlers
Configuring JSON web services - The new protocol handlers

The HTTP protocol handler

- Simpler syntax compared to tcp, ssl protocol handlers, e.g.

  SUBSECTION HTTP
  port=2080
  bind=my.server.name
  ENDSUBSECTION

- Compatible with TCPIP port sharing capabilities:
  - SHAREREPORT
  - SHAREPORTWLM with Gateway daemon health reporting
  - Sysplex Distributor
Configuring JSON web services - The new protocol handlers

The HTTPS protocol handler

- Shared Gateway daemon SSL resources and configuration
  - Common key ring with SSL protocol handler, IPIC SSL connections
  - Common NIST SP800-131A settings
  - Supports secure HTTP connections up to TLS 1.2, hardware crypto
- Common attributes with the HTTP protocol handler, plus
  - client authentication, defaults to off
  - cipher suite specification, defaults to all available
- Common syntax with the HTTP protocol handler, e.g.

```
SUBSECTION HTTPS
  port=2080
  bind=my.server.name
  ClientAuth=on
  CipherSuites=CipherSuite1,CipherSuite2
ENDSUBSECTION
```
Authentication and identity assertion for JSON web services

- HTTP(S) client includes an Authorization Request Header
- User name and password details are set on the ECI request
- Authentication then depends upon the target CICS connection protocol:
  - IPIC: IPConn defined with `USERAUTH=VERIFY` or if the target CICS uses client authentication, defaults to `off`
  - EXCI: Gateway daemon env-var `AUTHUSERPASSWORD=YES`
- Identity assertion is also possible (i.e. no password required)
  - IPIC: IPConn defined with `USERAUTH=IDENTIFY`
  - EXCI: CONNECTION defined with `ATTACHSEC=IDENTIFY`
- HTTPS combined with Basic Authentication is a likely implementation
JSON web services – defining a specific service

The new WEBSERVICE section

- Each JSON Web Service requires a single WEBSERVICE section

- Defined in the CICS TG configuration file, e.g.

```cicsscript
SECTION WEBSERVICE = inqcust
  Uri = customers/inquire
  bindfile = LGICUS01.wsb
  server = CICSAOR1
  timeout = 30
  transactionid = MYMI
  defaultmirror = Y
ENDSECTION
```

- Symbolic name for WS
- HTTP client uri mapping
- Data transform ws-bind
- Target CICS server
- Maximum wait time
- Mirror EIB TRNID value
- Attach default or ‘MYMI’
ECI vs HTTP errors with JSON web services

- JSON Web Services support utilize an internal HTTP server within the Gateway daemon.
- All responses map to standard HTTP return codes, e.g.:
  - **200**: Everything is OK
  - **403**: Security error – e.g. authentication failure
  - **404**: Not found – e.g. bad URI
  - **500**: Server error – e.g. unknown CICS server
    - Possibly a defect if combined with `ECI_ERR_SYSTEM_ERROR`
  - **503**: Service unavailable – CICS server unavailable
ECI vs HTTP errors with JSON web services

- Further details are encapsulated in the JSON response data as a fault string with reason codes.
- For example, an `ECI_ERR_NO_CICS` results in an HTTP 503 error (Service Unavailable), together with:

```json
{
    "Fault":{
        "detail":{
            "Description":"Communication with the target CICS server could not be established"
            "CICSServer":"<server name>"
        },
        "faultstring":"ECI_ERR_NO_CICS"
    }
}
```
New statistics in the Protocol Handler (PH) resource group

Port numbers

- **PH_SPORTHTTP**
  - HTTP protocol handler port number

- **PH_SPORTHTTPS**
  - HTTPS protocol handler port number

Bind address

- **PH_SBINDHTTP**
  - HTTP protocol handler bind address

- **PH_SBINDHTTPS**
  - HTTPS protocol handler bind address
The new WebServices (WS) statistics resource group

- **WS_SCOUNT, WS_SLIST, WS_ILIST, WS_LLIST**
  - Number and list of defined web services, lists of active web services

- **WS_IALLREQ, WS_LALLREQ**
  - Total number of web service requests processed

- **WS_IAVRESP, WS_LAVRESP**
  - Average Web Service response times

- **WS_IREQDATA, WS_LREQDATA, WS_IRESPDATA, WS_LRESPDATA**
  - Total amount of web service request and response data transferred

- **WS_IREQHI, WS_LREQHI**
  - High water marks for concurrent Web Service requests

- **WS_CREQ, WS_CWAITING**
  - Web service requests waiting for CICS, waiting for a Worker thread
The new specific WebServices (WSx) statistics resource group

- **WSx_SURI**
  - The HTTP uri mapping for Web Service “x”

- **WSx_SSSEVER**
  - The actual or logical CICS server to call for Web Service “x”

- **WSx_SPROGRAM**
  - The target CICS program associated with Web Service “x”
  - Derived from the WS BIND file

- **WSx_SEIBTRNID, WSx_SMIRROR**
  - Mirror transaction attributes for Web Service “x”
The new specific WebServices (WSx) statistics resource group

- **WSx_IALLREQ, WSx_LALLREQ**
  - Number of requests for web service “x” processed

- **WSx_IAVRESP, WSx_LAVRESP**
  - Average response times for web service “x”

- **WSx_IREQDATA, WSx_LREQDATA, WSx_IRESPDATA, WSx_LRESPDATA**
  - Amount of request and response data transferred for web service “x”

- **WSx_IREQHI, WSx_LREQHI**
  - High water marks for concurrent requests to web service “x”

- **WSx_CREQ**
  - Web service “x” requests waiting for CICS
Request monitoring capabilities have been extended to include unique attributes of JSON web service requests

- The request monitoring exit method, `eventFired`, receives a Map, with attributes defined by enumerated data type: `com.ibm.ctg.monitoring.RequestData`

- New attributes are provided for JSON web service requests:

  - `HttpPayload` – payload of mobile requests
  - `HttpVerb` – GET|POST|PUT|DELETE
  - `HttpPath` – The URI being invoked
  - `HttpStatusCode` – The return code sent to the client
JSON web services - the JSON web services assistant

Proven data transformation technologies and tooling

- JSON web services assistant uses shared components
  - CICS TS mobile feature pack
  - z/OS Connect
- Generates language structure mappings in ws-bind files, and JSON schemas
- The ws-bind files are used to transform data between JSON and binary representations, for COMMAREA and Channel programs
JCL samples for the web services assistant in the PDS, SCTGSAMP

- **CTGLS2JS** - Generates a web service binding file and JSON schemas from a language structure
- **CTGJS2LS** - Generates a web service binding file and language structures that you can use in your application programs, from JSON schemas
- **CTGJS2R** - Generates a web service binding file and a language structure that you can use in your RESTful applications, from a JSON schema
JSON web services - the JSON web services scenario
Get started by following the JSON web service scenario
Separate JSON web service workload to dedicated regions, and exploit the Gateway daemon’s high availability features

- Create a DSSGROUP representing the CICS servers dedicated to serving the mobile workload, using FailOver or RoundRobin algorithms
- Configure the WEBSERVICE to use the DSSGROUP

```
SECTION DSSGROUP = MOBIAORS
    Servers = MOBIAOR1, MOBIAOR2
    Algorithm = RoundRobin
ENDSECTION

SECTION WEBSERVICE = inqcust
    Uri = customers/inquire
    bindfile = LGICUS01.wsbind
    server = MOBIAORS
    timeout = 30
    transactionid = MYMI
    defaultmirror = Y
ENDSECTION
```
Modern connectivity

Connection management

• For 24x7 continuous operation

Exploits IPIC heartbeat support

• Improved availability across larger TCP/IP networks
IPIC heartbeat exploitation

Pro-active and continuous verification of connectivity status

- Increases reliability of IPIC connections over WANs
  - Reduces time to discover network issues
- Avoids problem of connection being silently dropped by firewall
- Communication while connection is idle
- Default setting is to send heartbeat every 30 seconds
- If response not received from target system
  - Connection is closed
IPIC connection management

Gateway daemon system management for IPIC connections

- Ability to stop and start IPIC connections
  - First time capability for CICS TG on z/OS, not possible with EXCI

- Selected and controlled quiesce of workload for a specific CICS server
  - Avoids the need to shut down the Gateway daemon
  - Carry out planned maintenance on selected CICS regions

- Allows for DSS group resilience
  - Take a connection out of use before stopping CICS
  - DSS algorithms continue to distribute work to alternative CICS servers
IPIC connection management – operations

New z/OS console SERVER commands

/F <jobname>,APPL=SERVER,STOP=<SERVER>

- Normal close of connection
- Allows for in-progress transactions to complete
- No new transactions can start

/F <jobname>,APPL=SERVER,STOP=<SERVER>,IMM

- Immediate stop of an IPIC connection
- In-progress transactions receive an error

/F <jobname>,APPL=SERVER,START=<SERVER>

- Start a server connection that was previously stopped
IPIC connection management – status visibility

New statistic for specific IPIC connection status

- **CSx_CSTATUS**
  - Represents the current status of specific IPIC connection “x”

- **Possible values for CSx_CSTATUS:**
  - **NOTSTARTED**: The Initial state of the connection
  - **STARTING**: The connection is in the process of being established
  - **AVAILABLE**: The connection is established, Gateway accepts requests
  - **UNAVAILABLE**: The connection has failed, Gateway rejects requests
  - **STOPPING**: The connection is closing, Gateway rejects new requests
  - **STOPPED**: The IPIC connection is closed, the Gateway rejects requests
z/OS Connect
(for those who didn't make breakfast!)
Extend existing enterprise data and business logic to Web, Mobile or Cloud apps

- Use WebSphere Liberty z/OS Connect for secure enterprise connectivity to easily extend existing assets to Mobile and Cloud applications using RESTful services and JSON.
- Leverage Websphere Liberty Java Connector Architecture (JCA) feature to connect to and extend existing enterprise backend systems
- IBM WebSphere Liberty Optimized Adapters for z/OS (WOLA): a function of WAS Liberty for z/OS that allows very fast, efficient, and low-latency memory to memory exchanges between WAS z/OS and CICS, IMS & Batch.

Administer production apps with the WebSphere Liberty Administrative Center
- Flexible, extensible, mobile ready, next generation admin UI to manage Liberty Servers

WebSphere Liberty Repository to pick up new Liberty product features, samples, and tutorials:
- Easily extend your development and production environments with new features

Different options for using WebSphere Liberty in the Cloud
- Build applications using the Liberty Buildpack and Caching services on IBM BlueMix
- Deploy WAS Liberty patterns using Pure Application Pattern service on SoftLayer
- Bring your own existing entitlement of WAS to SoftLayer or Amazon cloud environments
Problem Statement (s)

Customers on the z/OS platform today are increasingly expressing concerns about their ability to handle large spikes of new requests originating from any number of almost instantly available clients and systems that have a need for the business assets available there.

The fast advancing worlds of mobile and cloud computing are putting more and more pressure on applications and business logic located on z/OS in environments like CICS, IMS, batch, and others.

Customers have expressed an interest in a common solution that can be used by cloud, mobile, web and components like API management, that enables simple discovery and secure access to z/OS business and infrastructure assets using REST technology.

Infrastructure providers (cloud-based IaaS and SaaS providers) and mobile services registries (ie: API Management) require a uniform way to interact with z-based middleware for discovery, provisioning, data transformation, and service invocation.
What is it and what are the benefits for customers?

z/OS Connect is a Liberty based gateway that provides a secure and simple way to discover and call in to application assets/infrastructure on z/OS from Web/Cloud/Mobile applications using RESTful services.

The benefits include:

- **Fast on-ramp for z/OS customers to discover and reach z/OS applications** securely/simply using RESTful services. Service references can be copied from z/OS Connect and stored in any repository – cloud based (such as IBM Cloud OE) or mobile based (such as IBM Worklight, API Management) or any other web technology.

- **Light-weight and modular** providing flexibility to run multiple copies on the same or different z/OS systems and assign higher/lower priority to specific Liberty servers.

- **Integrated with z/OS management** makes the operations of the environment automated and consistent with the environments it is exposing.

- Provides ability to standardize on security access for calling in to z/OS applications in all major environments - CICS, IMS, batch, Unix System Services, and ISV software. Supports SAF-based security integration allowing for individual z/OS Connect services to have unique sets of authorized users.

- Provides ability to track and prioritize requests from cloud, mobile, web based external requestors using standard z/OS mechanisms like SMF and WLMA. Fulfills audit/chargeback needs for access to z/OS applications.
z/OS Connect

A service that encapsulate calling z/OS target applications using REST calls. z/OS Connect will support JSON payloads for calls from external cloud or mobile-based clients and will enable the conversion of the payload to the target program’s expected format. It will also provide the response payload conversion from a byte array into JSON format before returning the response to the caller.
z/OS Connect Liberty under CICS

Same z/OS Connect implementation – the CICS JCICS service provider handles requests targeted to existing CICS programs. CICS provides interceptors to integrate z/OS Connect with CICS security.

Complete your session evaluations online at www.SHARE.org/Pittsburgh-Eval
A little “gotcha”
Common Architectural Pattern

Comms Server TCPIP Stack

1. Comms Server
2. TCP/IP Stack
3. TDQ
4. ECB
5. CSMI
6. TOR
7. AOR
8. Business Logic
9. DPL

Complete your session evaluations online at www.SHARE.org/Pittsburgh-Eval
Remote asynchronous design

Comms Server TCPIP Stack

1. Comms Server
2. START
3. Business Logic
4. AOR
5. CSMI
6. TOR
7. ECB
8. TDQ
9. CSMI
Remote Asynchronous Design Notes

- EXEC CICS START dispatches asynchronous execution of the business logic
- Remote START requires the command to be shipped – incorporating a WLM opportunity, and a queue of ICEs in the AORs
  - WLM selects AOR and **synchronously ships** the START request
  - Queues of requests in the AOR
- WLM selects the AOR before the START is shipped, but execution is tracked once the business task is attached in the selected AOR – there is a **time gap** between decision and when resources are allocated
- Data is returned via a DPL, a TDQ and posting an ECB
  - Queues of responses in the TOR
Resources allocated

Comms Server TCPIP Stack

1. Resources allocated
2. Comms Server
3. TCPIP Stack
4. Session
5. Mirror Task Attach
6. Business Task Attach
7. TDQ
8. ECB
9. Long Running Child Server

Business Logic

CSMI

TOR

AOR

DPL

Complete your session evaluations online at www.SHARE.org/Pittsburgh-Eval
Resource allocation notes

- Resources consumed in shipping the START include...
  1. A session to communicate with the selected AOR
  2. A Task to receive the START request and add it to the AOR's Interval Control chain (ICE chain)
  3. A Task to run the business logic

- Resources consumed in shipping the data back...
  4. Session to communicate with the TOR
  5. Task to run the infrastructure logic to put data on the TDQ and post the ECB

- Thread may have to queue for each of these resources
- Child server must prioritise reading new work from the socket vs sending replies from the TDQ – this is flow control
  - TOR mirrors vs child server dispatch priority
Remote asynchronous dispatching – improvement opportunities

- **Workload classification and separation**
  - Safeguard critical work
  - Route work to dedicated AORs – CPSM may be doing this automatically
  - Simple “High”, “Medium” and “Low” scheme would give benefits

- **Multiple reply TDQs with different priorities**
  - Single TDQ allows no prioritisation of response processing
  - Use same “High”, “Medium” and “Low” scheme as for requests
Parallel Work – in a region

Comms Server
TCPIP Stack

Long Running Child Server

TOR

CSMI

TDQ

ECB

Business Logic

AOR
Parallel Work – in a plex

Comms Server TCPIP Stack

Long Running Child Server
CSMI
TOR

Business Logic
AOR

Business Logic
AOR
Local START with DPL

Comms Server TCPIP Stack

Long Running Child Server

Queue for session

Queue for mirror task attach

Queue for business task attach

Business Logic

CSMI

Queue for mirror attach

Queue for session

TOR

AOR

1 2 3 4 5 6
Local START notes

- Local START puts small, predictable load in TOR
  - Consumes a task in the TOR
- No need for WLM routing decision until DPL
  - Z/OS WLM Classification performed in TOR and propagated to the AOR
- Resources consumed in DPL to AOR
  - Session to communicate with AOR
  - Task to run business logic
- Three, rather than five, points where resources are consumed
  - Two fewer tasks attached (the start-shipping mirror and the DPL mirror)