Under the hood of IBM Integration Bus on z/OS (WLM, SMF, AT-TLS and More)

David Coles
IBM Integration Bus Level 3 Technical Lead,
IBM Hursley – dcoles@uk.ibm.com

2nd March 2015
17065

#SHAREorg

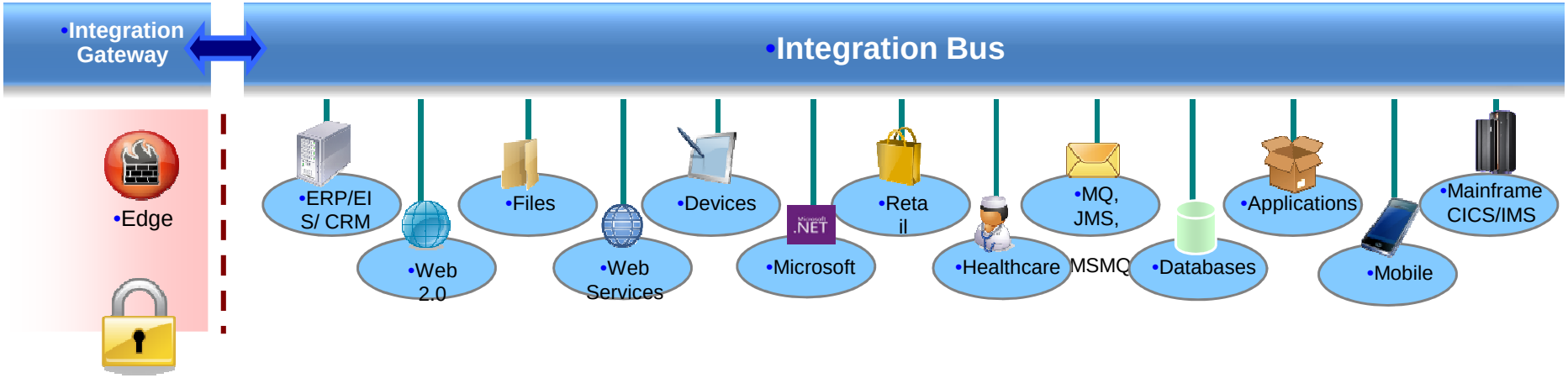
SHARE is an independent volunteer-run information technology association that provides education, professional networking and industry influence.

Session objectives

- IBM Integration Bus Overview.
  - Introducing IBM Integration Bus
  - Components of IBM Integration Bus
  - Message Flows + Nodes
  - Product Version Highlights
- IBM Integration Bus on z/OS.
- Value of IBM Integration Bus on z/OS.

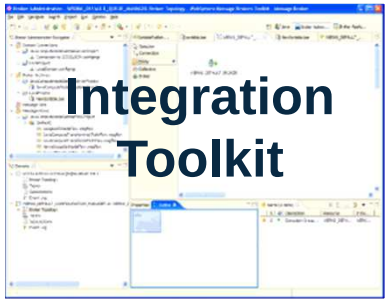
Introducing IBM Integration Bus

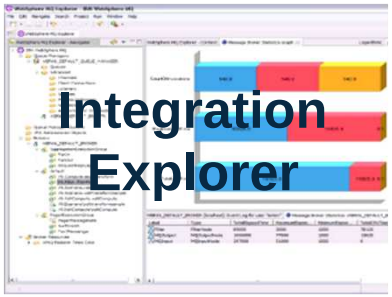
- **IBM's Strategic Integration Technology**
  - Single engineered product for .NET, Java and fully heterogeneous integration scenarios
  - DataPower continues to evolve for integration gateway use-cases

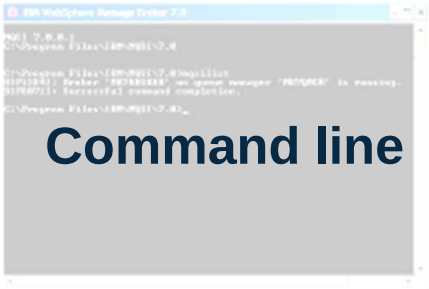


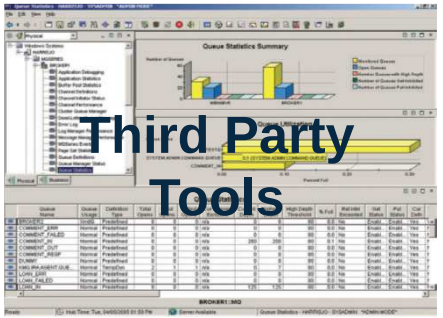
- **IBM Integration Bus is the new name for WebSphere Message Broker**
  - Technology progression over 15 years, installed at 2500+ customers worldwide across all industries
  - Fully supported worldwide by IBM global support network, standard 5 + 3 years support policy
  - Version to version migration is key design consideration
  - Global skills availability - SME's available globally via IBM and partners
  - Close interaction with growing and loyal customer base: beta and lab advocacy programs
  - Also incorporates WebSphere ESB use-cases

Components of IBM Integration Bus

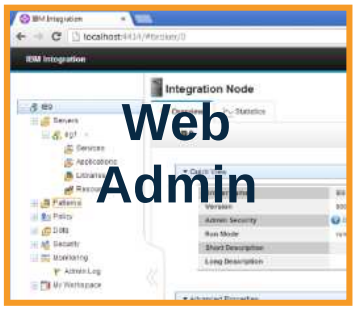




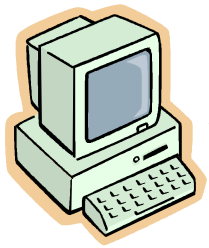




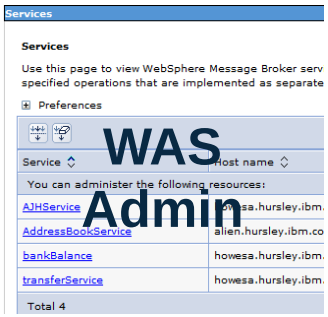
Integration API (CMP)



REST API

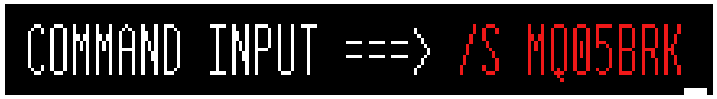


REST API

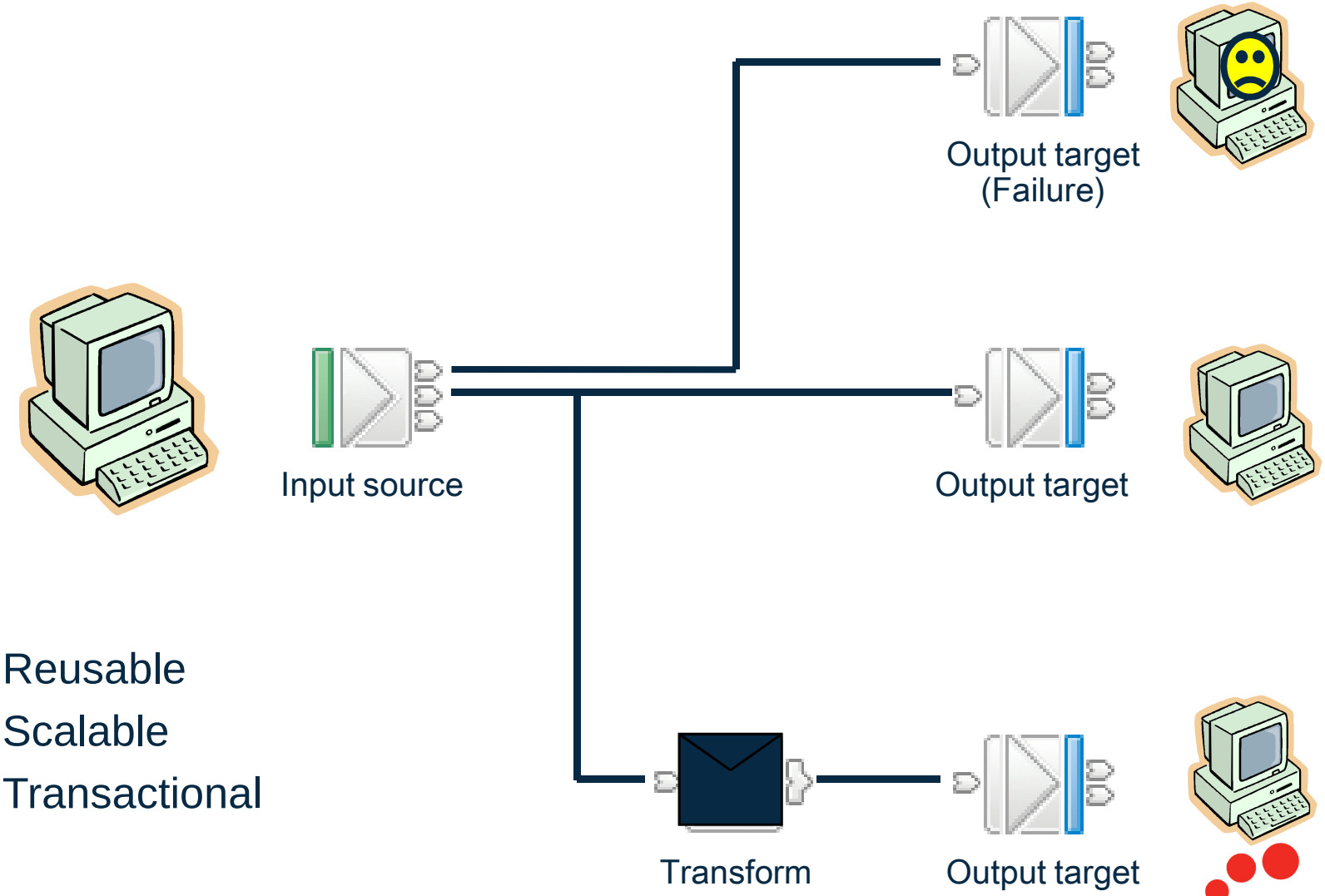


Integration Node

Console



Message Flows

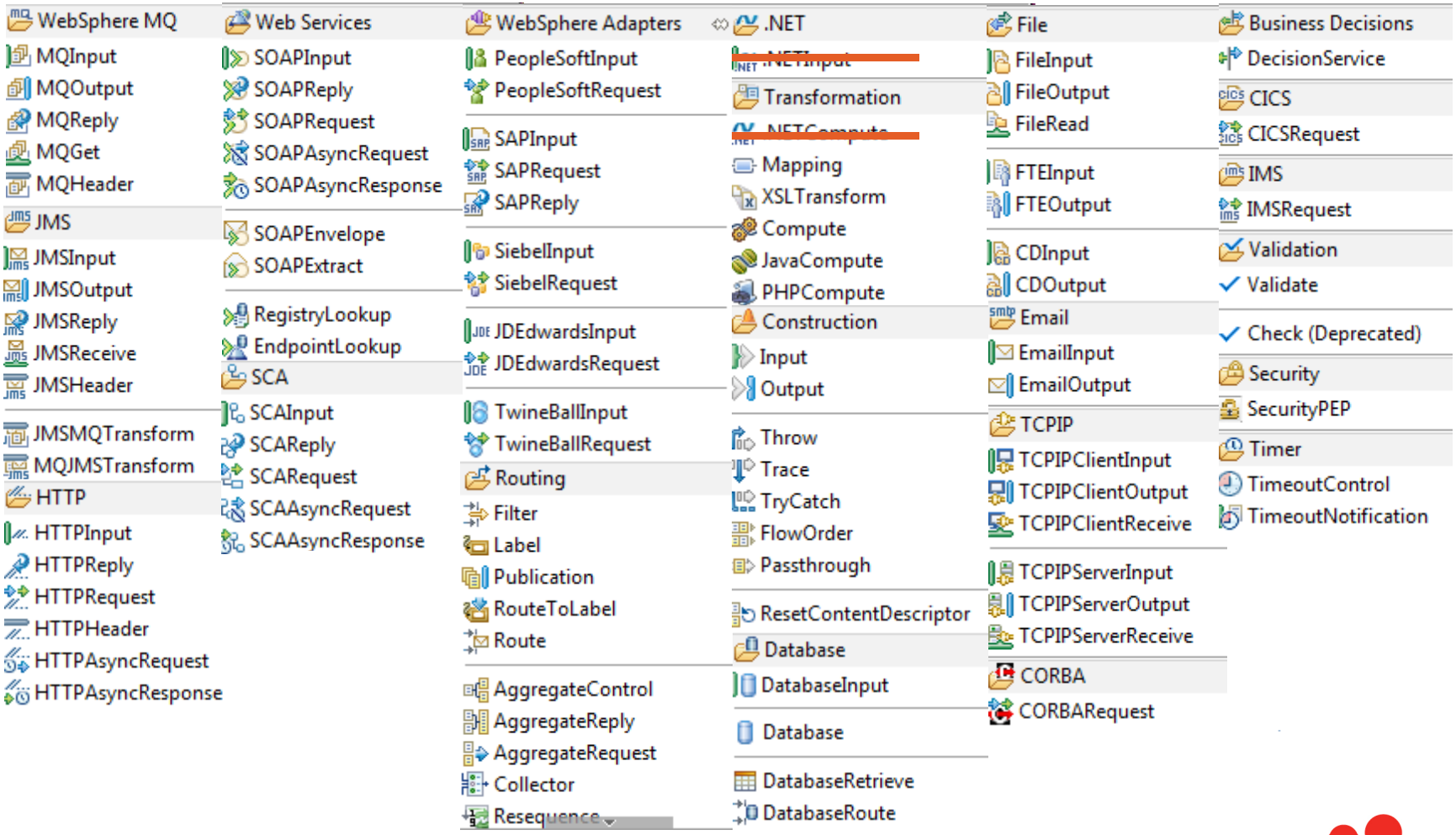


- Reusable
- Scalable
- Transactional

Notes : Message Flows

- Message flows provide the processing sequence required to connect applications together.
- A message flow contains the set of operations required to take a message from an originating application and deliver copies of it, some possibly transformed, to any number of connected applications for processing.
- As a message passes through a message flow, it is transformed and routed according to the nodes it encounters, and the processing decisions made within those nodes. Later we'll see how the nodes can modify the values within, or transform the structure of, a message to provide the data transformations necessary to drive backend server applications.
- For a given application scenario, the message flow describes all possible outcomes when processing a message. For example, if the message has a high monetary value, a copy of it might have to be routed to an audit application. Or if the message is not well-formed (may be it's not encrypted in the right format), it might be routed to a security application to raise an alert.
- Equally important is the visualization of the application integration within then organization. Very often, for any particular application scenario, the application connectivity requirements (*business*!) is held within the heads of domain experts. Being able to view the integration structure brings benefits in scenario understanding, reuse potential, and application architecture/standards conformance.
- After a message has been processed by a message flow, the flow does not maintain any state. It is possible to maintain such state in an external database, or within the message by using an extensible header such as the MQRFH2.
- Message flows are transactional.
  - Message flows provide vital processing and data manipulation and are therefore fully transactional. A message flow either completes all or none of its processing successfully.
  - However, if required, individual nodes can elect to perform operations outside of the message flow transaction. (e.g. audit)
- Message flows are multithreaded.
  - A given message passing through a series of nodes will execute on a single thread. To allow increased message throughput, message flows can be defined with many additional threads assigned to them. Peak workloads use additional threads, which are pooled during inactivity. We'll see more implementation details later. This means application scaling can be an operational rather than design time decision.
- Message flow nesting and chaining allow construction of enhanced capabilities.
  - Sophisticated flows can be rapidly constructed by linking individual flows together as well as nesting flows within each other.
- References:
  - Message Flow overview at http://publib.boulder.ibm.com/infocenter/wmbhelp/v7r0m0/topic/com.ibm.etools.mft.doc/ac00310_.htm

Message Flow Nodes



V7 Highlights

- Simplification
  - Broker system database removal
    - Only MQ and Java pre-reqs
  - Removal of the Configuration Manager and UserNameServer components
- Enhancements
  - z/OS 64bit, Now a true 64bit application
  - Execution Group specific profiles
  - WLM, Assign service classifications to individual address spaces
  - IBM Sterling ConnectDirect nodes (IC75621)

V8 Highlights

- **Simple & Productive**
  - Apps & Libs for streamlined development, packaging, deployment & management facilities
  - Programmable message flow API for totally flexible solution creation and customization
- **Universal & Independent**
  - Simple & high performing data modelling with DFDL
  - Code-free graphical transformations with GDM
  - Enhanced JMS connectivity, including JMS Receive node
  - SOAP nodes now support W3C WS-RM
  - Database & file enhancements
- **Industry Specific & Relevant**
  - Healthcare Connectivity Pack: Clinical application connectivity, healthcare patterns & end-user tooling
- **Managed & Dynamic**
  - Web 2.0 browser console with REST management API for ubiquitous access
  - Record & Replay to capture, view & replay in-flight messages
  - Message flow activity trace for rapid flow analysis by end-users
  - Deployable sub flows, ESQL, Maps & schemas for dynamic transforms
- **Different users per integration server on z/OS**
  - The userid associate with each integration server (execution group) address space is now configurable
  - The integration server exhibits the userid for all resource manager interactions (e.g. MQ, DB2)
  - Configurable via an integration server profile; takes effect when an integration server is started

Integration Bus V9 Overview

- **Simple & Productive**
  - Graphical Mapper: stored procedures, patterns and enhanced conversion of older maps
  - BPM Express/Standard (Lombardi) Integration: Process Designer synergy and integrated deployment
  - Web Tools: Real-Time Performance Statistics for understanding system behaviour
- **Universal & Independent**
  - WESB Conversion: Import and conversion of mediation flows and “to do” list
  - MQ service discovery to facilitate sharing of service definitions
  - Database discovery and analysis tools for diversified access to systems of record
  - DFDL improvements including lengthKind “pattern” and enhancements for TLOG
  - .NET Input node, Dynamics and MSMQ samples and patterns, support for Windows Server 2012
  - System of awareness for service mapping application-oriented integration
- **Industry Specific & Relevant**
  - Healthcare Pack update: MB8 Exploitation, DICOM Imaging, Analytics with Netezza and COGNOS
- **Dynamic & Intelligent**
  - Integrated Workload Traffic shaping policies to manage back-end system load
  - Managing unresponsive integration flows for improved overall system reliability
  - Business Decision Services using ODM technology for business rules integration
  - Security enhancements: Improved BasicAuth, Multiple certificates, CRL checking
- **High Performing & Scalable**
  - Embedded cache extensions: External cache, expiry and SSL support
  - Flexible Cloud Provisioning with IWS, SCAS and Pure, including Pure POWER support
- **Standard Edition Pricing on z/OS**
  - New entry-level edition offers flexibility to fulfil either broad-capability or high-performance scenarios
- **Co-ordinated transactions for CICS requests**
  - The CICSRequest node now supports coordinated transactions (one-phase commit)
  - Allows multiple requests to a CICS server to be handled as part of the same transaction
- **Activity log for CICS transactions**
  - Provides a high-level overview of the recent interactions between IBM Integration Bus and CICS
  - Includes CICS invocation successes, failures, abends, security, timeouts and transactional state

Session objectives

- IBM Integration Bus Overview
- IBM Integration Bus on z/OS
  - Preparing for installation and customization
  - Install and Customization
  - Starting the integration node (broker)
  - Displaying integration node output
  - Database Connectivity
  - Transaction Model
  - Non swapping integration server
  - Alternative integration server user ID
- Value of IBM Integration Bus on z/OS

Preparing for installation and customization

- Customize Unix System Services
  - There are certain USS parameters which must be customized for successful Integration Bus start-up and operation
  - `DISPLAY OMVS,OPTIONS`
    - MAXCORESIZE – 2147483647 (unlimited)
    - MAXCPUTIME – 2147483647 (unlimited)
    - MAXASSIZE - > 393 216 000
      - recommended 2147483647
    - MAXPROCUSER
      - Set greater than all nodes/servers/QMs/Chinits running with same user ID
    - MAXFILEPROC – Set > 65536
    - MAXTHREADS, MAXTHREADTASKS
      - Calculate and set based on number of flows, their instances, listener threads and then add 50
    - MAXMMAPAREA – Set > 55000
  - Semaphores + shared memory
    - Set according to number to number of Integration Server address spaces (n)
      - IPCSEMNIDS – n x4
      - IPCSHMNIDS - n
      - IPCSHMNSEGS - n

- Do not include the broker if you use the IEFUSI exit to limit the region size of OMVS address spaces

Preparing for installation and customization

- Ensuring sufficient space for temporary files
  - Envvar *TMPDIR* to point at a temp directory
    - Uses /tmp by default if not set
  - >100mb required

- **Do not run the broker as root!**

- Broker user ID
  - Requires OMVS segment + home directory
  - Execute

```
LU userid OMVS
```

```
USER=MQP1BRK NAME=SMITH, JANE OWNER=TSOUSER
CREATED=99.342 DEFAULT-GROUP=TSOUSER PASSDATE=01.198
PASS-INTERVAL=30
......
OMVS INFORMATION
----------------
UID=0000070594
HOME=/u/MQP1BRK
PROGRAM=/bin/sh
CPUTIMEMAX=NONE
ASSIZEMAX=NONE
FILEPROCMAX=NONE
PROCUSERMAX=NONE
THREADSMAX=NONE
MMAPAREAMAX=NONE
```

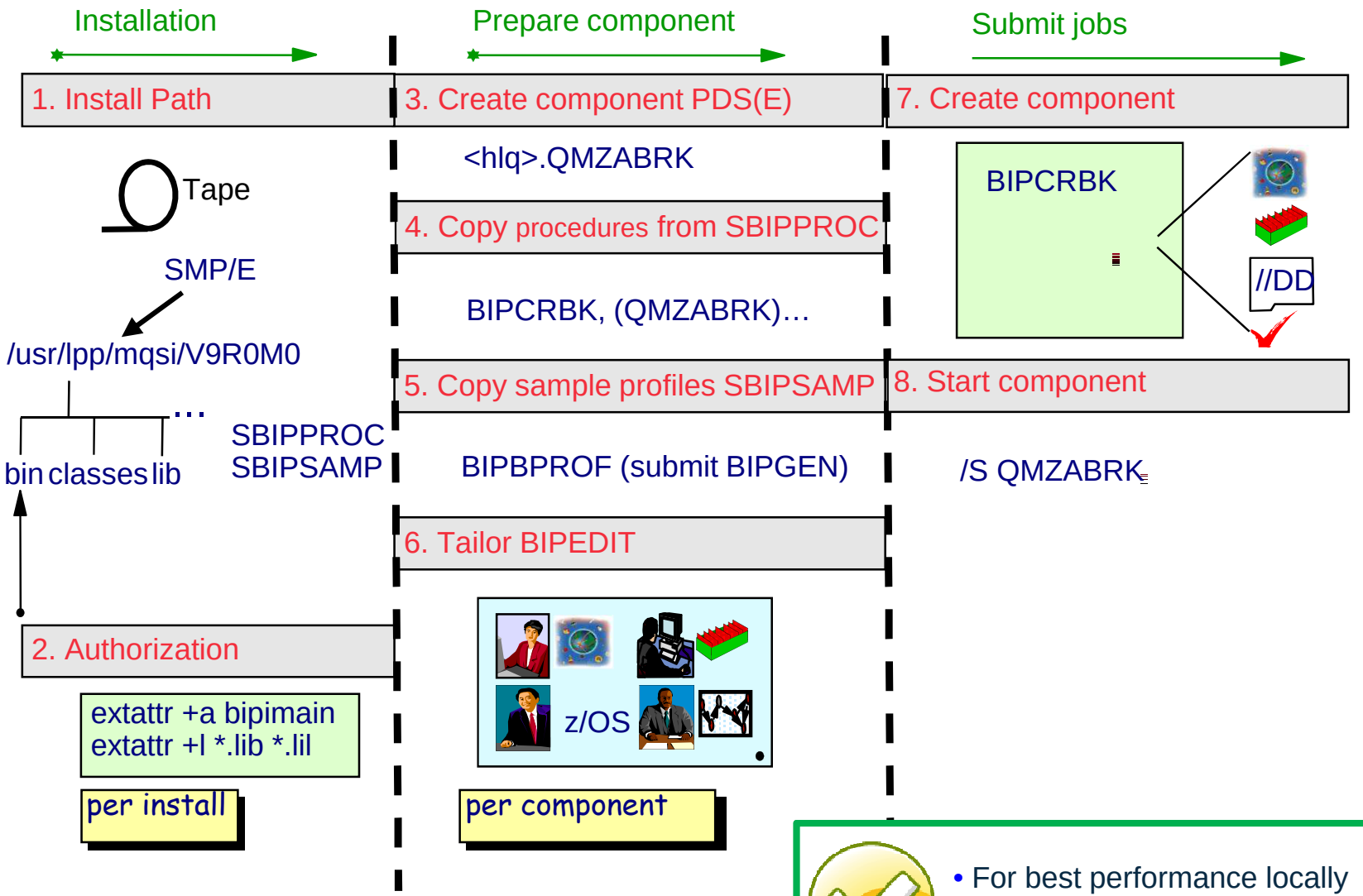
  - Home directory must be large enough to take any unexpected dumps which could be written there (> 2gb)
    - svcdumps normally written as datasets to the system dump HLQ
  - Associate the userid with the started task name for the broker

Component directory

- It is recommended that each node (broker) has its own component HFS
  - Stops any space fill ups effecting more than 1 node
- Holds the runtime-deployed configuration information and output directories for the component
  - Includes deployed flows, jar files, wsdl, xsd, etc
  - Also includes trace files
  - Configuration information
    - ++COMPONENTDIRECTORY++\components
    - ++COMPONENTDIRECTORY++\config
    - ++COMPONENTDIRECTORY++\registry
    - ++COMPONENTDIRECTORY++\shared-classes
  - Traces & logs
    - ++COMPONENTDIRECTORY++\common
      - Could mount as separate HFS for best reliability
- Allow minimum of 1Gb for configuration and 5-10Gb for traces and logs

•The Component directory and the home directory must be separate to ensure that, when dumps are taken in the home directory, they do not cause problems with the runtime broker that still has to write to the Component directory.

Installation and Customization

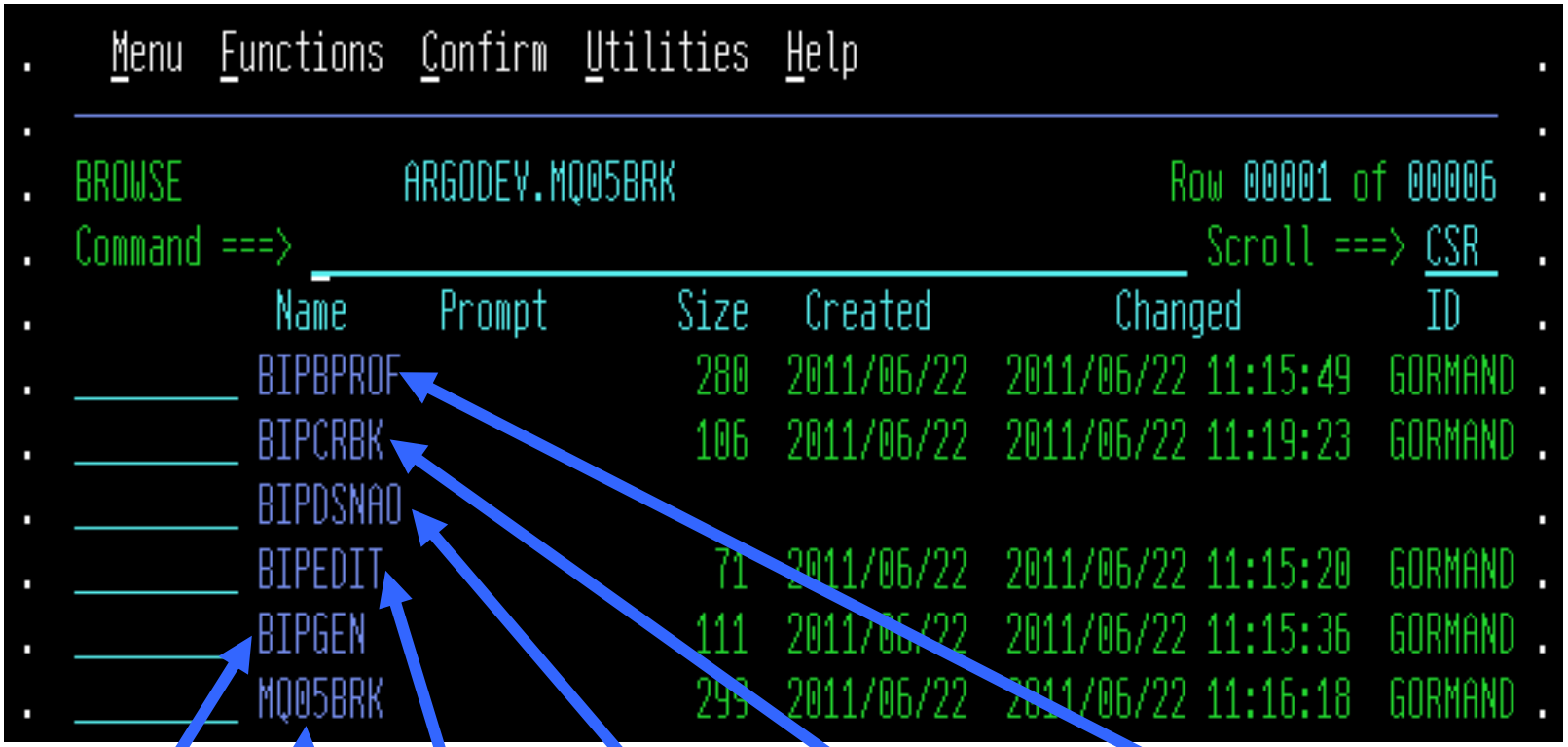


• For best performance locally mount any HFS used by the broker to the LPAR where it's running

Notes : Installation and Customization

- **Install IBM Integration Bus via SMP/e.**
- **Allocate integration node (broker) PDS(E)**
  - Copy required component procedures from SBIPPROC
  - Rename component as required e.g. QMZAxxxx
- **Customize component dataset**
  - Copy profiles from SBIPSAMP
    - BIPBPROF - integration node profile
  - Other parameter files
    - BIPEDIT – JCL edit file
- **Tailor BIPEDIT JCL edit procedure**
  - Assign settings for registry, MQ, Java etc
- **Target appropriate profile with BIPEDIT**
  - Use BIPGEN to create ENVFILE
- **Target appropriate jobs with BIPEDIT**
  - BIPCRBK– Create integration node
  - Optionally modify options according to authority
    - Default is all resources
      - -1 only create the registry in USS. (USS Administrator).
      - -2 only do the MQ creation pass (MQ Administrator).
- **Create component**
  - Submit BIPCRBK to create component
- **Target component JCL with BIPEDIT and rename**
  - e.g. BIPBRKP
- **Start component**
  - e.g. /S QMZABRK
- **Component verification automatically runs as the first step**

JCL



Generate environment file

Integration node Started Task JCL

Customization JOB

ODBC configuration

JOB to create an integration node (broker)

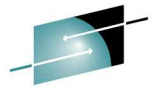
Integration node profile

A full set of JCL exists for all IBM Integration Bus commands.

Notes : JCL

- All JCL for a specific integration node (broker) are copied to the integration node's component dataset.
- The JCL is customized using BIPEDIT.
- JCL exists to run all integration bus commands, such as mqsicreatebroker and mqsilist etc…

JCL Variables



JCL variable	Description	Example value
++COMPONENTDATASET++	The data set where all JCL relevant to a particular component is saved.	TESTDEV.MQP1BRK.BROKER
++COMPONENTDIRECTORY++	The file system directory where the component exists. This directory includes subdirectories, for example, `/log` and`/registry`	`mqsi/brokers/MQP1BRK`
++COMPONENTNAME++	The name you give the component when you create it.	MQP1BRK
++HOME++	The file system home directory for the user ID of this component. This value is required to dynamically generate the ENVFILE.You must have the appropriate external security manager authorities to write to this file system directory when submitting JCL to run a command. For example, use RACF® to define authorities.	`/u/mqp1usr/mqp1brk`
++INSTALL++	The directory where you install the product.	`/usr/lpp/mqsi`
++JAVA++	Location of your Java™ installation.	`/usr/lpp/java/IBM/J1.5`
++LANGLETTER++	The letter for the language in which you want messages shown.	E (English)
++LOCALE++	Locale of environment where commands are run by submitting JCL.	C
++MQPATH++	WebSphere® MQ location.	`/usr/lpp/mqm`
++OPTIONS++	Many commands submitted by JCL require additional options. See the reference material for additional information about options specific to each command.	N/A
++QUEUEMANAGER++	The name of the queue manager associated with the component for which you submit the JCL.	MQP1
++STARTEDTASKNAME++	Name of the Started Task JCL. This name can be a maximum of eight characters.	MQP1BRK
++TIMEZONE++	Time zone of environment where commands are run by submitting JCL.	GMT0BST
++WMQHLQ++	WebSphere MQ high-level-qualifier	MQM.V700

Integration Server Specific Profiles

Integration node default profile

Jobs to create ENVFILEs

Integration server specific profile

```
VIEW          ARGODEV.MQ05BRK                    Row 00001 of 00008
Command ===>                                     Scroll ===> CSR
      Name     Prompt      Size  Created        Changed          ID
_____ BIPBPROF              280  2011/06/22  2011/06/22 11:15:49  GORMAND
_____ BIPCRBK               106  2011/06/22  2011/06/22 11:19:23  GORMAND
_____ BIPDSNAO
_____ BIPEDIT                71  2011/06/22  2011/06/22 11:15:20  GORMAND
_____ BIPEGEN                 1  2011/07/15  2011/07/15 12:16:30  GORMAND
_____ BIPGEN                111  2011/06/22  2011/06/22 11:15:36  GORMAND
_____ DAVE                    1  2011/07/15  2011/07/15 12:16:22  GORMAND
_____ MQ05BRK               299  2011/06/22  2011/06/22 11:16:18  GORMAND
```

- There is a base profile for each integration node. This is used by default for all integration node address spaces.
- Integration server specific profiles can also be specified.
- Change BIPGEN to create specific profiles.

```
VIEW       ARGODEV.MQ05BRK(BIPGEN) - 01.00
Command ===>
000108 //**********************************
000109 //BIPEG01 EXEC PROC=BIPEGEN EG=DAVE
```

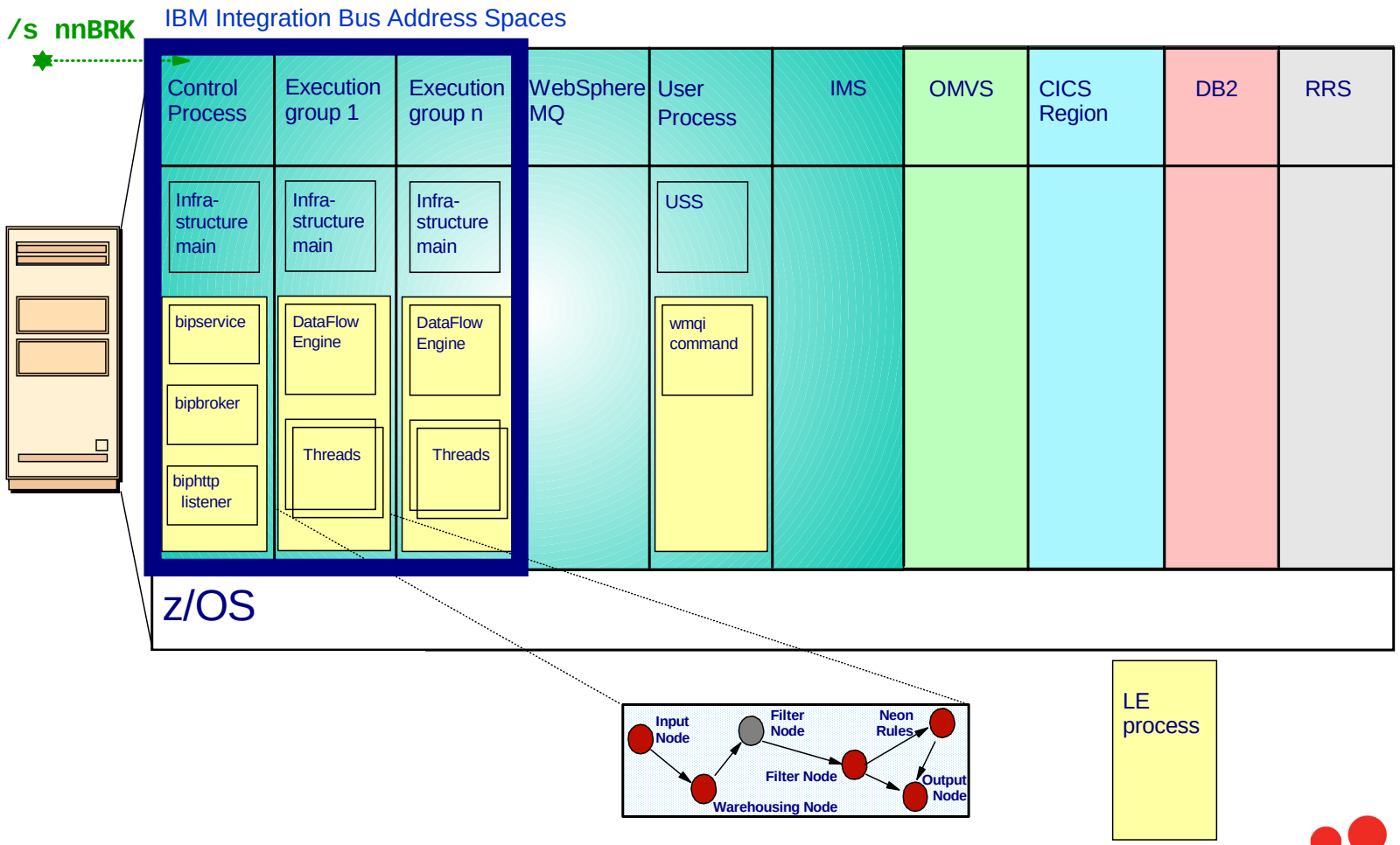
Notes : Integration Server Specific Profiles

- By default integration servers run with the normal integration node profile.
- However, each integration server address space can run with a customised profile.
- The integration server specific profile is created by customizing a new BIPEPROF member in the component dataset, modifying and submitting BIPGEN.
- This produces an ENVFILE specific to the integration server, and can include anything that can be specified as an environment variable.
- Displaying the integration node in SDSF tells you which address spaces are running with specific profiles or the default profile.
- The IBM Integration Bus infocenter explains how to configure this.

Processes

- **bipimain**
  - z/OS only. The first process in any IIB address space. APF authorised to set up authorised routines.
- **bipservice**
  - Lightweight and resilient process that starts and monitors the bipbroker process (a.k.a AdminAgent).
  - If the bipbroker process fails, bipservice will restart it.
- **bipbroker**
  - A more substantial process. Contains the deployment manager, for CMP and REST connections, as well as the WebUI server. All commands, toolkit connections and WebUI go through this process.
  - Responsible for starting and monitoring the biphttplistener and DataFlowEngine processes.
  - If either process fails, bipbroker will restart them.
- **biphttplistener**
  - Runs the brokerwide HTTP connector for HTTP and SOAP nodes.
- **DataFlowEngine** (a.k.a Integration Server)
  - Runtime engine for all deployed resources.

Starting the integration node



Notes : Control Address Space

- First address space started per integration node.
- Contains the bipservice process. This includes the console listener for commands such as /F (modify) and /P (stop).
- Contains the bipbroker (AdminAgent) process. This starts and monitors DataFlowEngine (Integration Server) processes. Also includes the Deployment Manager which handles requests from the IBM Integration Bus toolkit, IBM Integration Bus Explorer, CMP API and the Web UI.
- Contains the biphttplistener process which handles integration node wide HTTP connections.
- Includes:
  - Log of messages from bipservice, bipbroker and biphttplistener
  - STDERR and STDOUT
  - Copy of ENVFILE and BIPDSNAO

Notes : Integration server Address Space

- Each integration server runs in its own address space.
- One or many flows can be deployed to a single integration server.
- Provides workload isolation and WLM configuration options.
- Each address space can run with a unique set of environment variables (for example JVM options).
- Each address space includes:
  - A log of all messages from the DataFlowEngine process
  - STDERR, STDOUT
  - Copy of the ENVFILE (environment file)
  - Copy of the DSNAOINI (ODBC configuration file)

Displaying an integration node in SDSF

```
COMMAND INPUT ===> /S MQ05BRK
```

StepName same as JOBNAME and ProcStep BROKER, identifies this as the control address space.

```
SDSF DA MVD1     (ALL)    PAG 10  CPU/L    31/ 29      LINE 1-3 (3)
COMMAND INPUT ===> _                                   SCROLL ===> CSR
PREFIX=MQ05BRK  DEST=(ALL)  OWNER=*  SORT=JOBNAME/A  SYSNAME=*
NP   JOBNAME  StepName ProcStep JobID    Owner    SysName  C Pos DP Real     Paging
     MQ05BRK  MQ05BRK  BROKER   STC52908 MQ05BRK  MVD1       IN  F0   23,762   0.00
     MQ05BRK  DAVE2    EGNOENV  STC52910 MQ05BRK  MVD1       IN  F0   50,590   0.00
     MQ05BRK  DAVE     EGENV    STC52909 MQ05BRK  MVD1       IN  F0   44,392   0.00
```

Integration node JOBNAME. All address spaces associated with a single integration node have the same name!

Integration Servers, identified by name! (last 8 chars of name)

EGENV identifies this address space as running with a specific profile!

EGNOENV identifies this address space as running with the integration node profile!

Displaying an integration node at the process level

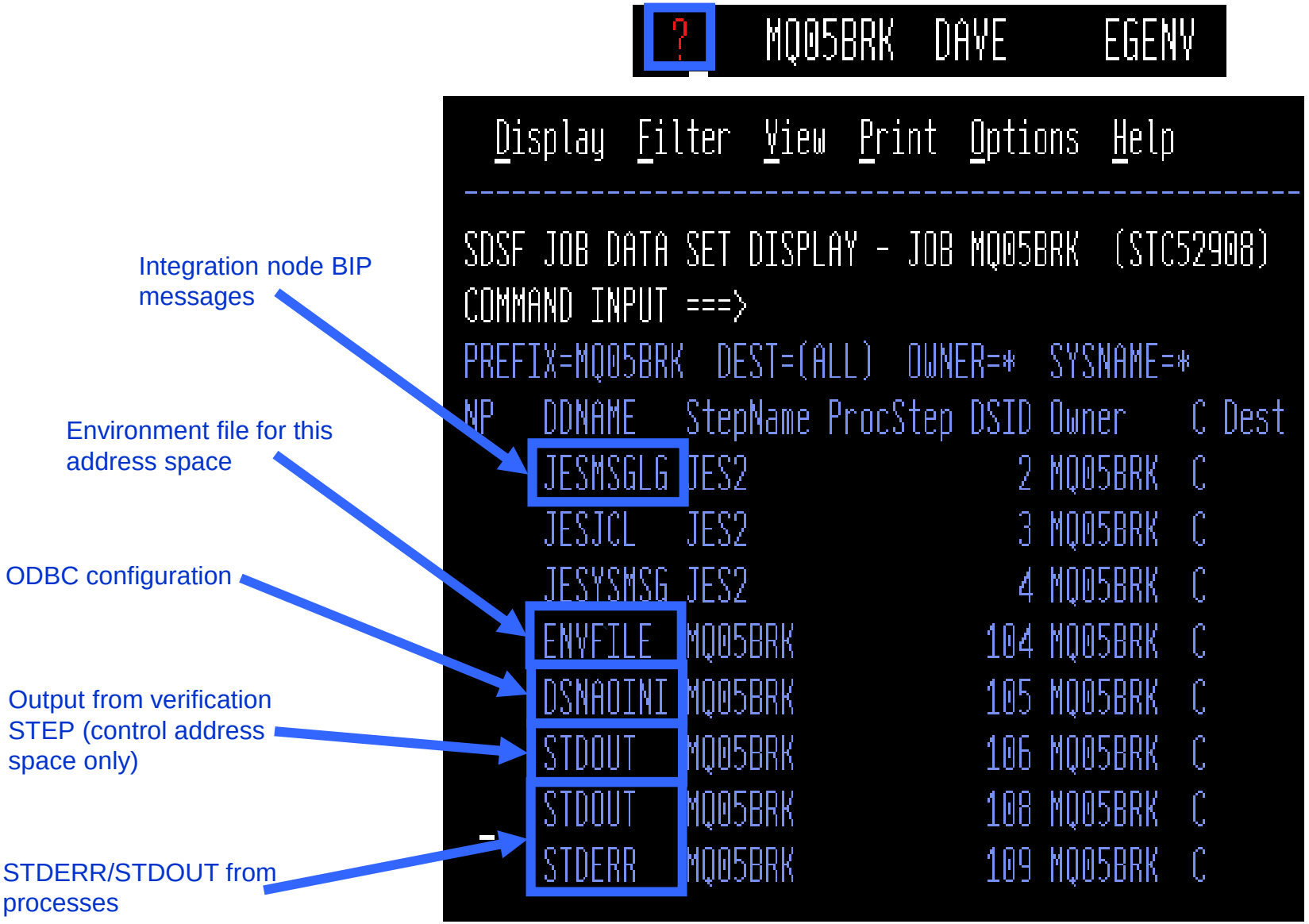
```
SDSF PROCESS DISPLAY  MVD1     ALL                          LINE 1-11 (11)
COMMAND INPUT ===> _                                          SCROLL ===> CSR
PREFIX=MQ05BRK  DEST=(ALL)  OWNER=*  SORT=ASID/A  SYSNAME=*
NP   JOBNAME  JobID    Status                                Owner    Command
     MQ05BRK  STC52908 FILE SYS KERNEL WAIT                  MQ05BRK  BPXBATA8
     MQ05BRK  STC52908 RUNNING                               MQ05BRK  biphttplistener MQ05BRK
     MQ05BRK  STC52908 RUNNING                               MQ05BRK  bipservice MQ05BRK AUTO
     MQ05BRK  STC52908 WAITING FOR CHILD                     MQ05BRK  /argoinst/DAVE/usr/lpp/mqsi/
     MQ05BRK  STC52908 RUNNING                               MQ05BRK  bipbroker MQ05BRK
     MQ05BRK  STC52910 RUNNING                               MQ05BRK  DataFlowEngine MQ05BRK 4faaf
     MQ05BRK  STC52910 WAITING FOR CHILD                     MQ05BRK  /argoinst/DAVE/usr/lpp/mqsi/
     MQ05BRK  STC52910 FILE SYS KERNEL WAIT                  MQ05BRK  BPXBATA8
     MQ05BRK  STC52909 RUNNING                               MQ05BRK  DataFlowEngine MQ05BRK cc32d
     MQ05BRK  STC52909 FILE SYS KERNEL WAIT                  MQ05BRK  BPXBATA8
     MQ05BRK  STC52909 WAITING FOR CHILD                     MQ05BRK  /argoinst/DAVE/usr/lpp/mqsi/
```

```
/u/gormand:>ps -ef | grep MQ05
 MQ05BRK   84017209   50463369 /argoinst/DAVE/usr/lpp/mqsi/bin/bipimain bipservice MQ05BRK AUTO
 MQ05BRK   84017284   84017835 DataFlowEngine MQ05BRK cc32dfb6-3001-0000-0080-a253e63dfe32 DAVE
 MQ05BRK   67240352   84017924 biphttplistener MQ05BRK
 MQ05BRK   67240502   84017209 bipservice MQ05BRK AUTO
 MQ05BRK   84017835   84017546 /argoinst/DAVE/usr/lpp/mqsi/bin/bipimain DataFlowEngine 00071016
    DFS     131792     131828 grep MQ05
 MQ05BRK   84017924   67240502 bipbroker MQ05BRK
```

Notes : Displaying integration server at the process level

- Displaying an integration node in SDSF.PS or in USS shows the different processes running inside the individual address spaces.
- On z/OS, a special process called bipimain is started first in all address spaces. This performs authorised functions.
- Bipservice is started in the Control address space. It contains the console listener, and starts bipbroker.
- Bipbroker (also known as the AdminAgent) starts and monitors the DataFlowEngine processes and the biphttplistener process. The AdminAgent contains the DeployManager, which the IIB toolkit, IIB Explorer, commands,CMP API and WebUI connect to.
- The biphttplistener process is the integration node HTTP listener.
- The DataFlowEngine process is the integration server in which message flows run.

Integration node/server JOB output



Notes : Integration node/server JOB output

- An integration server JOB includes any BIP product messages and anything written to STDOUT and STDERR by the processes contained within the address space.
- The ENVFILE and DSNAOINI (BIPDSNAO in the component's dataset) are also copied for a complete record.

Integration Server Verification

- When an integration server starts, the control address runs a verification STEP checking the configuration and environment.

```
BIP8873I: Starting the component verification for component 'MQ05BRK'.
BIP8876I: Starting the environment verification for component 'MQ05BRK'.
BIP8894I: Verification passed for 'Registry'.
BIP8894I: Verification passed for 'MQSI_REGISTRY'.
BIP8894I: Verification passed for 'Java Version - 1.6.0 IBM z/OS build pmz6460sr5-20090604_01 (SR5)'.
BIP8894I: Verification passed for '64-bit Java'.
BIP8894I: Verification passed for 'MQSI_COMPONENT_NAME'.
BIP8894I: Verification passed for 'MQSI_FILEPATH'.
BIP8900I: Verification passed for APF Authorization of file '/argoinst/DAVE/usr/lpp/mqsi/bin/bipimain'.
BIP8894I: Verification passed for 'Current Working Directory'.
BIP8878I: The environment verification for component 'MQ05BRK' has finished successfully.
BIP8882I: Starting the WebSphere MQ verification for component 'MQ05BRK'.
BIP8886I: Verification passed for queue 'SYSTEM.BROKER.ADMIN.QUEUE' on queue manager 'MQ05'.
BIP8886I: Verification passed for queue 'SYSTEM.BROKER.EXECUTIONGROUP.QUEUE' on queue manager 'MQ05'.
BIP8886I: Verification passed for queue 'SYSTEM.BROKER.EXECUTIONGROUP.REPLY' on queue manager 'MQ05'.
BIP8884I: The WebSphere MQ verification for component 'MQ05BRK' has finished successfully.
BIP8874I: The component verification for 'MQ05BRK' has finished successfully.
```

Notes : Integration Server Verification

- When an integration server is first started, a verification program runs in the control address space.
- This checks the correct level of Java is available, that it can access the necessary files, and that bipimain is APF authorised, among many other checks.
- If a problem is found, the error is reported in the JOBLOG and the integration node is not started.

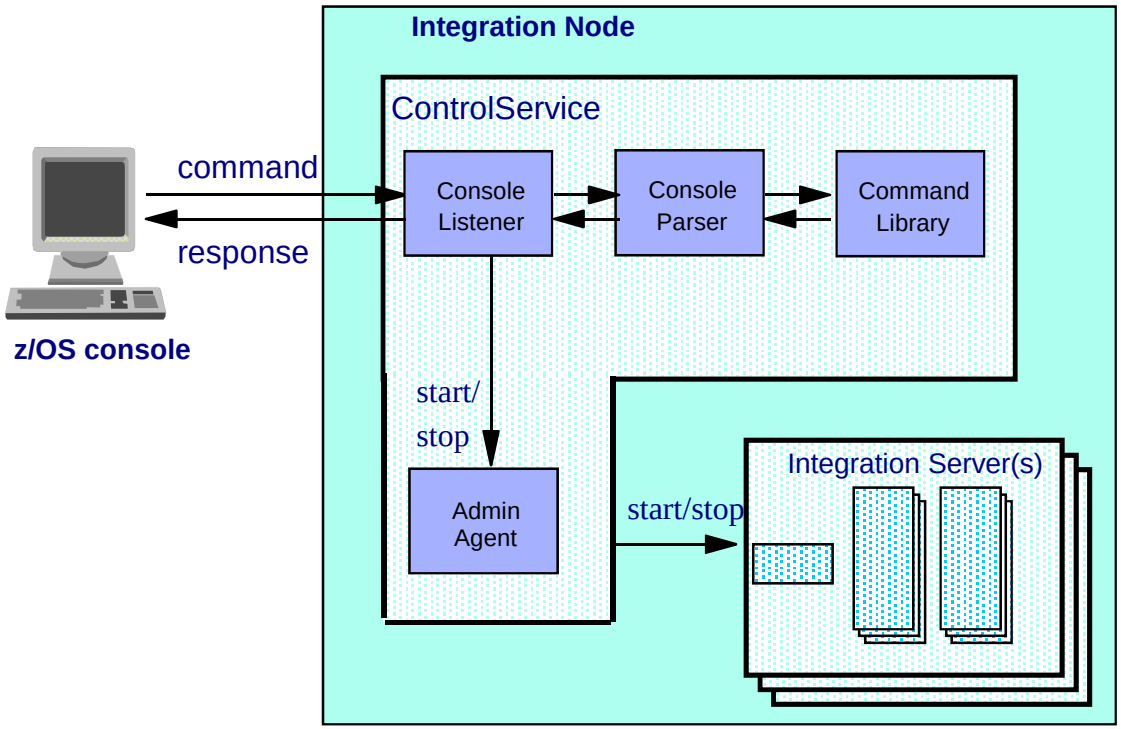
Using SDSF for performance monitoring

- In SDSF view the integration server address spaces for memory, paging and CPU utilisation:

NP	JOBNAME	StepName	ProcStep	JobID	Owner	SrvClass	RptClass	SysName	C	Pos	DP	Real	Paging	SIO	CPU%	ASID	ASIDX	EX
	MQ05BRK	LMEXPERT	BROKER	STC63060	MQ05BRK	STCFAST	COLIN	MVD1		IN	F4	55,031	0.00	3.62	0.03	205	00CD	
	MQ05BRK	MQ05BRK	BROKER	STC63059	MQ05BRK	STCUSER	MQ05	MVD1		IN	EC	30,901	0.00	0.00	0.05	208	00D0	
	MQ05BRK	DAVE	BROKER	STC63061	MQ05BRK	STCUSER	DQ05BASE	MVD1		IN	EC	56,046	0.00	3.62	0.03	240	00F0	

  - Real = 4k pages

Console commands



Short	Long
SC	Start component
PC	Stop component
CT	Change trace
RT	Report trace
L	List
RE	Reload
CB	Change broker
CS	Change flow stats
RS	Report flow stats
DP	Deploy
CX	Change flow user exits
RX	Report flow user exits
RC	Reload security
CM	Change flow monitoring
RM	Report flow monitoring
CR	Change resource stats
RR	Report resource stats

Start an integration node	/S <Broker>
Stop an integration node	/P <Broker>
Modify an integration node	/F <Broker>,cmd

Notes : Console commands

- A number of IBM Integration Bus commands can be entered directly from the console.
- The commands are listed on the previous slide.
- Administrative changes can also be made using JCL, the IIB Explorer, CMP API or the Web UI.

Console command output

```
F PF01BRK,L
+BIP8071I PF01 2 Successful command completion.
F PF01BRK,L E='default'
+BIP8071I PF01 2 Successful command completion.
```

List commands

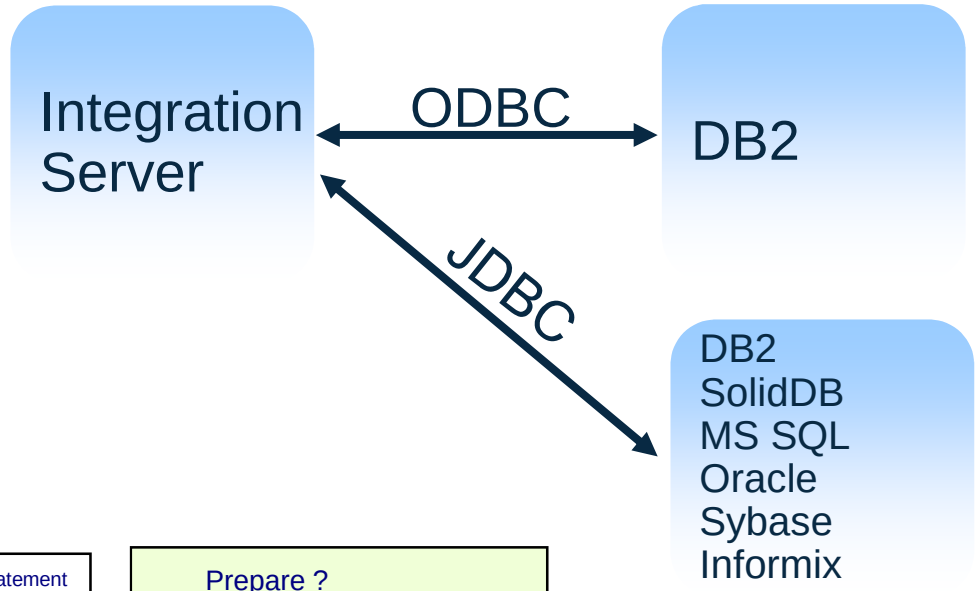
```
+BIP1286I PF01 2 EXECUTION GROUP 'default' ON BROKER 'PF01' IS RUNNING. : ImbNativeTrace(723)
+BIP8071I PF01 2 Successful command completion.
+BIP1288I PF01 2 MESSAGE FLOW 'V6.inputOutput' ON EXECUTION GROUP 'default' IS RUNNING. : ImbNativeTrace(723)
+BIP8071I PF01 2 Successful command completion.
```

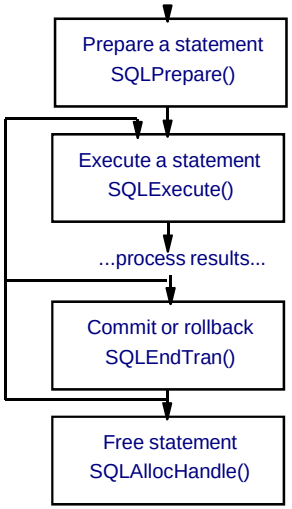
Output from list commands

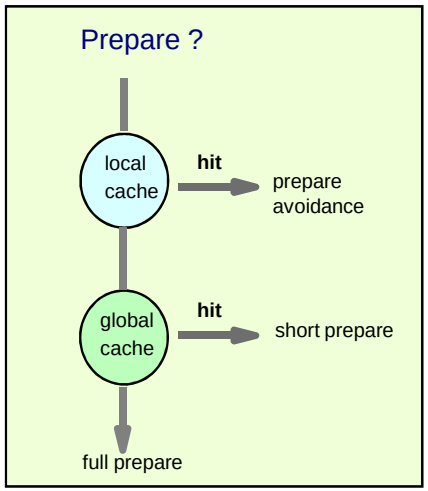
Notes : Console command output

- Console command output is written to the control address space and to the system log.

Database Connectivity on z/OS







Local Dynamic Statement Caching

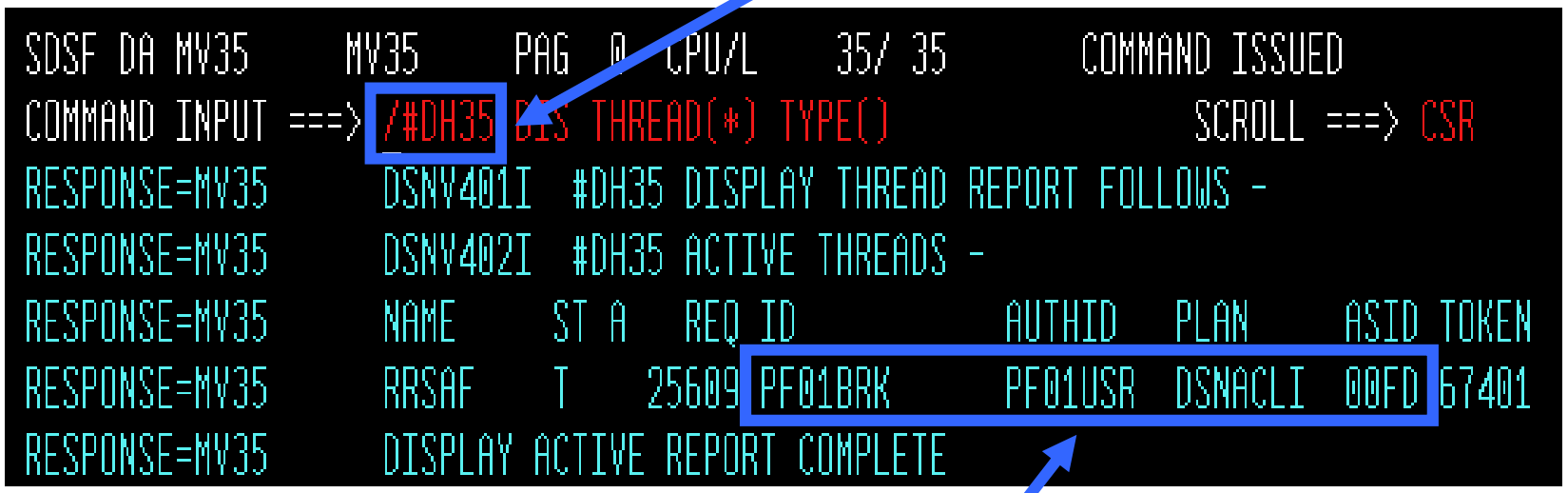
Global Dynamic Statement Caching

- Easy access to user databases
- Access DB2 via ODBC
  - *full statement caching and prepare avoidance is used to give high performance access.*
- Access DB2 and other Databases via JDBC

Notes : Database Connectivity

- **What is Open Database Connectivity (ODBC)?**
  - You may well be used to using Structured Query Language (SQL) to write programs that access and update data in a database.
  - SQL does not have a call level interface, rather it is a declarative language which is translated by means of a product specific preprocessor which changes the SQL statements into product specific calls to a database plan. This plan is a representation of the access your program requires to the database.
  - You're probably also aware that even though SQL is standard, the process of translating and binding SQL is not! ODBC is designed to address this problem. It is important to IBM Integration Bus as it provides uniform data source access on any platform to support the Compute, Filter and Database nodes for routing, warehousing and data enrichment without product specific SQL preprocessing.
- **Benefits of using ODBC.**
  - ODBC SQL is not precompiled or bound; it uses standard functions to execute SQL statements and related services at run time.
  - ODBC enables you to write portable applications that are independent of any particular database product translation or binding scheme. This independence means applications do not have to be recompiled or rebound to access different data sources, but rather just connect to the appropriate data source at run time.
  - One of the disadvantages of ODBC is that plans have to be generated at runtime, and are invalidated after commit processing. As the majority of cost is associated with the plan, ODBC is as expensive as dynamic SQL. We'll see how this cost can be alleviated.
- **ODBC transaction processing model.**
  - A program using OBDC, and your deployed message flow is just that (!), takes the SQL statement (issued by the filter, compute, database nodes) and prepares it using the SQLPrepare call. This is exactly analogous to the binding process that static SQL has - it's just that the plan is built at run time. (Configure ODBC tracing in the BIPDSNAO file to observe application behaviour!)
  - The prepared statement is then executed as part of a transactional unit of work, and the results processed. The statement can be re-executed before the transaction is committed and this greatly improves performance. However, once the transaction has been committed, as long as you're using local and dynamic statement caching, this prepare is only performed once.
  - If you're using DB2, use GDSC (ZPARM CACHEDYN=YES) to improve the performance of ODBC type applications by caching prepared statements in using a skeleton statement kept beyond transaction commit. The integration server uses Local Statement Caching for prepare avoidance, significantly improving database performance, bringing it close to static SQL.
- **Check SOE for DB2 APARs required for 64bit ODBC driver!**

Displaying integration node threads in DB2

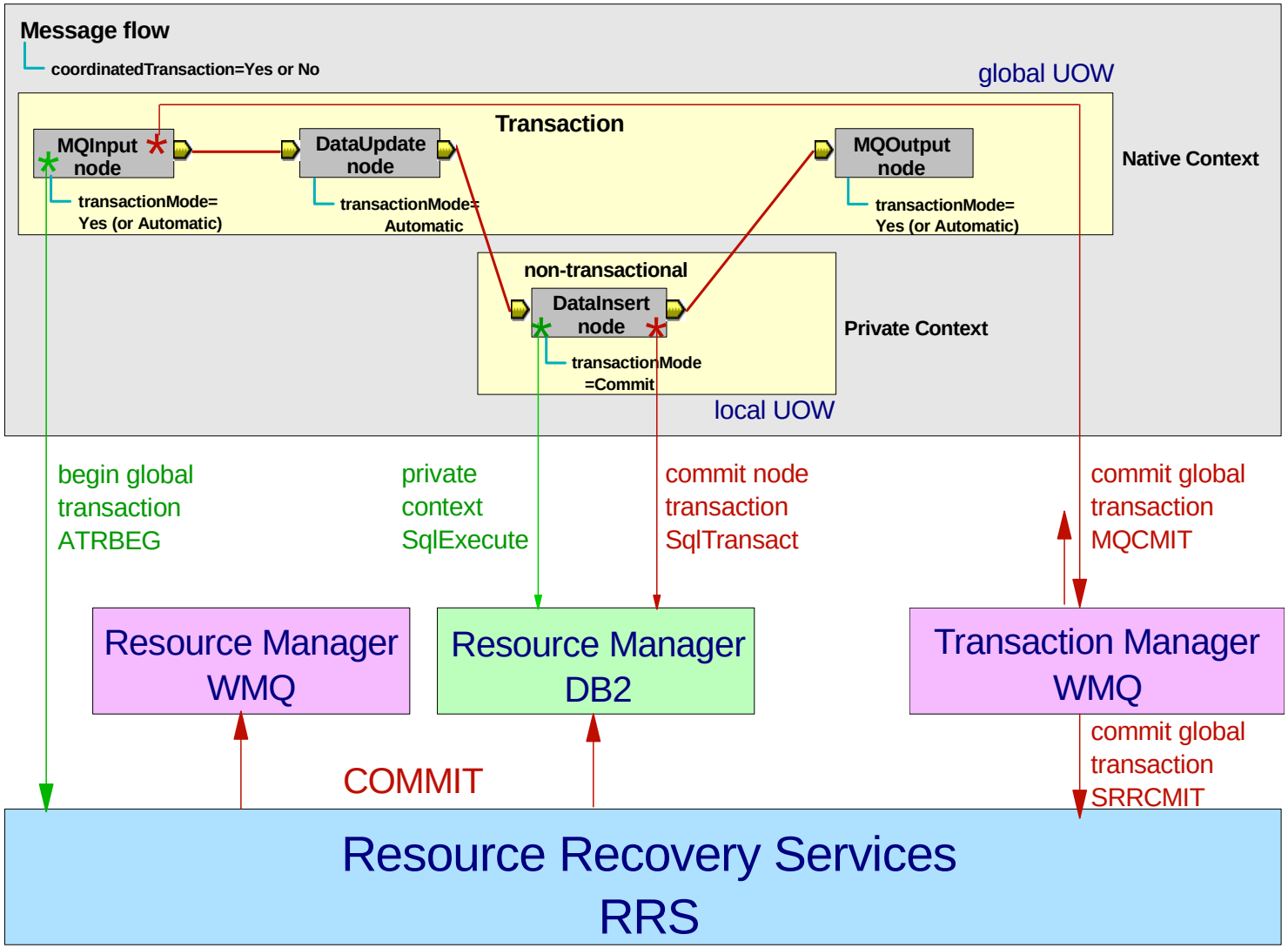


Notes : Displaying integration node threads in DB2

- By displaying active threads in the DB2 subsystem, we can see that the integration server has a connection.
- This shows RRS as the transaction coordinator, it also shows the USERID and PLAN.

Transaction Model

The z/OS integration node has a global transaction model exactly as you'd expect. It is possible for nodes to elect to commit outside this transaction. RRS is used for context management & commitment control between the flows resource managers, but only when required.



Notes : Transaction Model

- **Transactional message flows are important**
  - A message flow which transforms and routes data often has a need to be transactional. That is, the message flow must complete either *all or none* of its processing. Remember, from an end-to-end application perspective, the message flow is *part* of the application.
- **Transactional data flows and data nodes.**
  - A message flow can be identified as transactional using the Coordinated Transaction checkbox on the assigned message flow. The intention behind this attribute is that all node operations within the message flow can be coordinated under the same, global, transaction. On z/OS, this option is always used for message flows, whether selected or not.
  - A node performs its operations within the envelope of this message flow global transaction, and can elect to be within the global transaction or not. A Transaction Mode checkbox enables this for WMQ and database nodes. Note the visibility (ACID) implications!
- **Resource Recovery Services (RRS) is *NOT* always the transaction coordinator.**
  - As message flows run in essentially a batch type address spaces, RRS is the global transaction coordinator, if required.
  - Integration servers are linked with an MQ RRS stub, so WMQ registers an interest with RRS for commitment control.
  - Specifying the keywords CONNECTTYPE=2, AUTOCOMMIT=0, MULTICONTEXT=0, and MVSATTACHTYPE=RRSAF in the initialization file BIPDSNAO enables global transaction processing.
- **RRS Context**
  - RRS Context is a concept that enables a program to have different roles. It's like one person having many ways of behaving which don't interact with each other. It means that applications can simultaneously be different things to different systems.
  - Flows have two contexts. A *native* context is used whenever it wants to perform the role of including node operations under the global transaction. A *private* one has the effect of excluding database node operations from a global transaction.
  - Plug-in nodes are always within the global transaction. A message flow is always in *native* context for these nodes.
- **WebSphere MQ**
  - *Transaction Mode* within a message queuing node governs whether MQPUT and MQGET operations are explicitly performed either inside or outside syncpoint on a per call basis. These nodes therefore always use the native, and never the private, RRS context.
- **Database**
  - For database nodes, *Transaction Mode* determines under which RRS context the transaction will be performed. If the node is within the global transaction, then the native context is used. For a Transaction Mode of *commit*, the private context, so that DB2 and RRS see the operation from a logically different party. These nodes commit (using SQLTransact) their operations as the node is exited.
- **Commitment Control**
  - The global transaction is begun implicitly when a resource manager communicates with RRS. The overall message transaction is committed (or backed out!) control returns to the input node. At COMMIT time, WMQ will pass control to RRS only if required.
  - RRS will call all registered resource managers (WMQ, DB2) in a two phase commit protocol to ensure a global transaction. Recall that nodes which elected for a Transaction Mode of commit had resources updated (and externally visible!) close to their point of issuing. If RRS is not required WMQ will perform the commitment control and delete any RRS interests.

Configuring an integration server address space as non-swappable

- Configure 1, many, or all integration server address spaces as non-swappable
  - Cannot be set as NOSWAP in the PPT due to being USS processes
- Set environment variable in the ENVFILE
  - MQSI_NOSWAP=yes
    - Set all servers as non swappable
  - MQSI_NOSWAP_egname=yes
  - MQSI_NOSWAP_uuid=yes
    - Set specific server as non swappable
- Broker started task userid needs READ access to the BPX.STOR.SWAP facility class
- Employ caution when you use this as it can cause additional real storage to be converted to preferred storage

Notes : Configuring an integration server address space as non-swappable

- Because broker execution groups run as processes in UNIX System Services they cannot be set as NOSWAP in the PPT.
- Instead, you can set the following environment variables in the broker environment file so that some, or all, of the address spaces of the broker execution groups request that they become non-swappable by the system; see Creating the environment file for further information on adding an environment variable to a broker.
- MQSI_NOSWAP=yes
  - sets the address spaces of all the execution groups to be non-swappable.
- MQSI_NOSWAP_egname=yes
  - issues a request to the system, for each execution group labelled egname, that the address space be set as non-swappable.
- MQSI_NOSWAP_uuid=yes
  - issues a request to the system, for each execution group with the UUID labelled uuid, that the address space be set as non-swappable.
- In order for the above requests to succeed, the broker's started task ID needs READ access to the BPX.STOR.SWAP facility class through their external security manager, for example, RACF®.
- When an application makes an address space non-swappable, it can cause additional real storage in the system to be converted to preferred storage. Because preferred storage cannot be configured offline, using this service can reduce the installation's ability to re-configure storage in the future.

Specifying an alternative user ID to run an execution group on z/OS

- Run an integration server (execution group) as a different userid to the main broker user ID
- Means you can access resources according to the permissions assigned to the different user ID, rather than the permissions assigned to the main broker user ID
- Requires a new RACF started task profile with a new user ID
  - New user ID must be created with an OMVS segment including a unique UID, home directory, and the ability to create data sets under the broker's HLQ and alias
  - The new started task procedure name must start with the same four characters as the main broker started task
  - New user ID has to have the same RACF primary group as the existing broker user ID
  - Also ensure that the new user ID has the required privileges to the existing broker filesystem and dataset
  - Ensure that the MQ and SMF authorizations are updated for the new user ID
- Copy the existing broker started task JCL to the new started task JCL in the PROCLIB.
- Main broker user ID requires permission to the *SUPERUSER.PROCESS.KILL* RACF profile
- User ID changed by adding the appropriate environment variable to the broker's ENVFILE
  - If you specify more than one environment variable, they are read in the following order
    1. MQSI_STARTEDTASK_FIXED_executionGroupName
    2. MQSI_STARTEDTASK_MULTI_executionGroupName
    3. MQSI_STARTEDTASK_DEFAULT
  - For example:
    - MQSI_STARTEDTASK_FIXED_DEFAULT=MQ01EG1 (server name last 8 characters = 'DEFAULT')
    - MQSI_STARTEDTASK_MULTI_TEST=MQ01EG2 (server name last 8 characters start with 'TEST')
    - MQSI_STARTEDTASK_DEFAULT=MQ01EG3 (changes all not explicitly overridden servers)

Notes: Specifying an alternative user ID to run an execution group on z/OS

You can change the user ID under which an execution group runs so that it can access resources according to the permissions assigned to it, rather than the permissions assigned to the main broker user ID.

Complete the following steps to specify an alternative user ID for the execution group, to be used instead of the broker's user ID:

1. Create the new RACF® started task profile with a new user ID, which will be used to run the execution group. Consider the following points when you are creating the new started task:
   - The new started task must be created with an OMVS segment including a unique UID, home directory, and the ability to create data sets under the broker's HLQ and alias.
   - The started task procedure name to be used for the execution group address space must start with the same four characters as the main broker started task. For example, if the main broker started task is MQ01BRK, the started task name for the execution group could be MQ01EG1 but not MQ02EG2. As a result, consistency is maintained between the main broker started task, the execution group, and the queue manager, which helps to identify the relationship between them. If the first four characters are not the same, the execution group is started using the main broker started task JCL.
2. Ensure that the new user ID associated with the new started task JCL has the same RACF primary group as the existing broker user ID, so that they can access shared resources. Also ensure that the new user ID has the required privileges to the existing broker filesystem and dataset (which it should have through the primary group access).
3. Ensure that the MQ and SMF authorizations are updated for the new user ID; for more information, see Summary of required access (z/OS).
4. Copy the existing broker started task JCL to the new started task JCL in the PROCLIB.
5. Ensure that the main broker user ID has been granted permission to the SUPERUSER.PROCESS.KILL RACF profile. This permission is required so that the main control address space can recover any existing execution group address spaces in the event of a failure.
6. Refresh the started RACF classes to implement the updates.
7. Change the user ID by adding the appropriate environment variable to the broker's profile.
   - The execution group name specified in the environment variable is the last 8 characters of the execution group, after any overrides have been applied. This is the same 8-character name that is displayed as the STEPNAME against the execution group address space in SDSF.
   - Ensure that the execution group name contains only characters that are valid in the environment variable. If invalid characters are used, the user ID cannot be overridden.
   - If you specify more than one environment variable, they are read in the following order (with MQSI_STARTEDTASK_FIXED_executionGroupName taking precedence):
     A. MQSI_STARTEDTASK_FIXED_executionGroupName
     B. MQSI_STARTEDTASK_MULTI_executionGroupName
     C. MQSI_STARTEDTASK_DEFAULT
   - where executionGroupName is the name of your execution group. For example:
     - export MQSI_STARTEDTASK_FIXED_DEFAULT=MQ01EG1 changes any execution group which has the last 8 characters equal to DEFAULT to started task MQ01EG1
     - export MQSI_STARTEDTASK_MULTI_TEST=MQ01EG2 changes any execution group which has the last 8 characters starting with TEST to started task MQ01EG2
     - export MQSI_STARTEDTASK_DEFAULT=MQ01EG3 changes all execution groups which are not overridden by MQSI_STARTEDTASK_FIXED_executionGroupName or MQSI_STARTEDTASK_MULTI_executionGroupName to started task MQ01EG3.
8. Submit BIPGEN to the broker's ENVFILE.
9. Restart the broker.

Session objectives

- IBM Integration Bus Overview
- IBM Integration Bus on z/OS.
- Value of IBM Integration Bus on z/OS.
  - High Availability
  - WLM
  - Accounting and Chargeback
  - Reduced TCO
  - Using AT-TLS with Integration Bus
  - z/OS specific nodes

High Availability

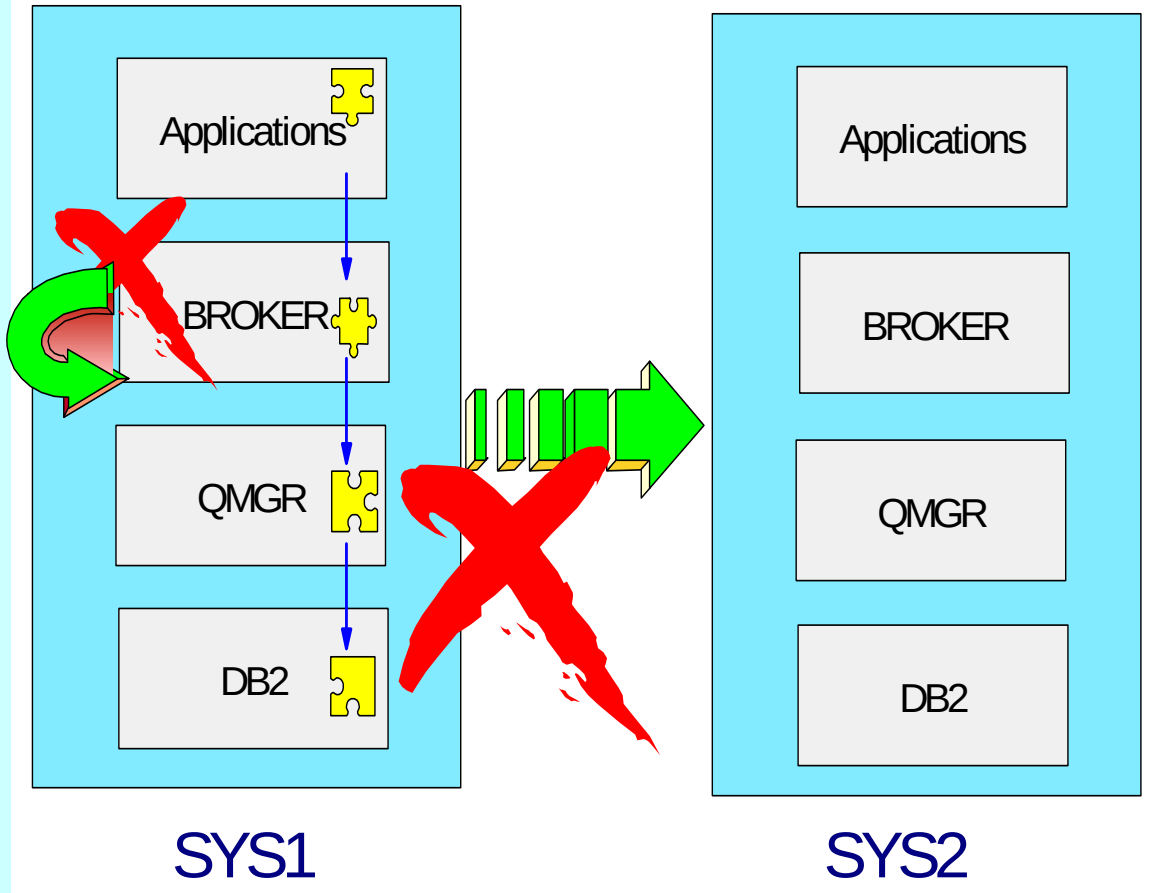
- **Parallel SYSPLEX**
  - Message flows deployed to integration nodes on z/OS can benefit from using resource managers such as WebSphere MQ (Shared Queues) and DB2 (Data Sharing Group) which are Parallel SYSPLEX aware.
- **Automatic Restart Manager (ARM)**
  - ARM is a z/OS subsystem which ensures appropriately registered applications and subsystems can be restarted after a failure.
  - Applications and subsystems can either be restarted on the image on which they failed, or in the case of image failure, on another available image in the SYSPLEX.
- **WebSphere MQ Clustering**
  - For z/OS users not using Parallel SYSPLEX, MQ clustering provides an excellent way to exploit high availability for new messages in the event of failure.
  - Although MQ clustering is a “shared nothing solution”, in many scenarios it can be good enough to enable the availability of new work, especially when combined with ARM.

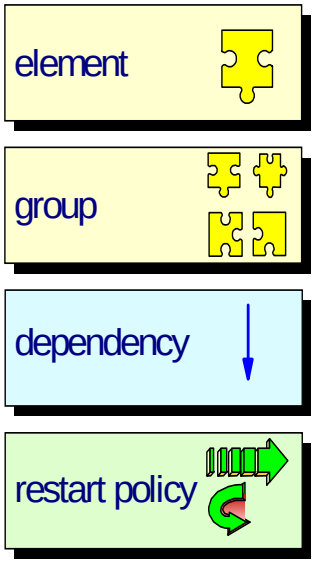
Notes : High Availabilty

- Parallel SYSPLEX.
  - Message flows deployed to integration nodes on z/OS can benefit from using resource managers such as WebSphere MQ and DB2 which are Parallel SYSPLEX aware.
  - Objects including WebSphere MQ queues and DB2 databases are available as a shared resource within a SYSPLEX, making them simultaneously available to all applications within the SYSPLEX. As long as one z/OS image is available these resources are available for use.
  - What's really powerful though, is the fact that this is a truly shared resource, and in the event of failure, the remaining images will make the failed work available immediately, without the need for restart. This is high availability in its highest sense.
  - For example, a message flow deployed to an integration server in a WebSphere MQ *Queue Sharing Group* transforming XML messages from a shared input queue, may perform that processing *anywhere* in the SYSPLEX.
  - As long as one queue manager in the group remains available, messages can continue to be transformed.
  - What differentiates this from an availability perspective over "shared nothing" type solutions is that after a failure, rolled back messages are made available immediately within the SYSPLEX such that they can be processed by other flow instances servicing the shared input queue.
  - Similar capabilities exist for DB2 resources when using DB2 *Data Sharing Groups*.
- Automatic Restart Manager (ARM).
  - ARM is a z/OS subsystem which ensures appropriately registered applications and subsystems can be restarted after a failure.
  - Applications and subsystems can either be restarted on the image on which they failed, or in the case of image failure, on another available image in the SYSPLEX.
  - IBM Integration Bus is fully ARM enabled.
  - It is simple for an operator to assign an ARM classification and name to the integration node. When the integration node is started or stopped, it will perform the necessary registration with ARM, enabling it to be restarted as defined in the ARM policy by the z/OS systems programmer.
- WebSphere MQ Clustering
  - For z/OS users not using Parallel SYSPLEX, MQ clustering provides an excellent way to exploit high availability for new messages in the event of failure.
  - In this mode of operation, integration nodes in a domain have a partitioned copy of the input queue, and clustering ensures that in the event of a failure, new messages are sent only to queues which are still available.
  - Although MQ clustering is a "shared nothing solution", in many scenarios it can be good enough to enable the availability of new work, especially when combined with ARM.

Restart Management using ARM

Restart is defined by the Systems Administrator to organize applications into dependent groups for recovery after failure. Failures can include application failure or system failure and restart can be performed in place or on an adjacent Sysplex image.





Notes : Restart Management using ARM

- **What is Automatic Restart Manager (ARM)?**
  - ARM is part of z/OS XCF and is used to restart failed address spaces after their failure or the failure of an image upon which they were executing. Subsequently, they can be either restarted on the same image, or another, if this has failed.
  - ARM therefore improves the availability of specific batch jobs or started tasks.
  - The design of IIB restart is to make only the Control Process address space ARM restartable. The hierarchical relationship between the Control Process address space and in turn, its relationship to the integration servers means that this is sufficient.
- **Elements and Restart Groups.**
  - An address space that is using ARM services is referred to as an "element." The systems programmer, rather than the integration node, defines a restart group as a set of named elements that have affinities and must be restarted together in the event of a system failure. The restart group also governs the restart sequence of elements, so that subsystem dependencies can be modelled and enforced in terms of subsystem restart.
  - An element is comprised of an 8 character type and a 16 character name. For IIB, ARM usage is indicated and configured in the profile BIPBPROF. For example USE_ARM=YES, ARM_ELEMENTNAME='MQ03BRK' and ARM_ELEMENTTYPE='SYSWMQI'.
- **How to Use and Configure.**
  - The ARM couple data set is the repository of the ARM policies of an installation and also of the specifics of elements with ARM status. This data set is separate from other couple data sets (e.g., those of workload manager), may have an alternate data set, and must be connected to all systems where registration and restart might occur. The ARM policy is a set of instructions from an installation about how and where (and whether) restarts are to be done. The main purpose of a policy is to define the elements comprised by a group, with particulars about dependencies in the group, overriding sources of restart techniques and parameters, selection criteria in cross-system restarts, and pacing of restarts.
  - When you configure an integration node to use ARM, you'll probably want to make it dependent on many prerequisite address spaces. These include WMQ and DB2, as their operation is vital to the integration node.
  - In a similar way, you might like to have ARM policies for important, restartable applications that use the integration node.
- **Reference.**
  - For more information on ARM, consult Sysplex Services Guide GC28-1771.

Workload Management

- **Workload Scaling**
  - z/OS SYSPLEX brings a huge amount of potential processing capacity to the IBM Integration Bus with up to 101 EC12 processors.
  - Message flows have a built-in multiprocessing capability. Each *instance* of a message flow can execute on a separate thread (z/OS TCB) allowing the integration server to easily exploit multiple CPUs
- **Workload Isolation**
  - Simply assign separate integration server address spaces to different types of work.
- **z/OS Individual Integration Server Workload Management**
  - Integration Servers automatically assigned to **JOBACCT** token
  - Use WLM Classification panels to map **JOBACCT** to Service and Report classes

NP	JOBNAME	StepName	ProcStep	JobID	Owner	SrvClass	RptClass	SysName	C	Pos	DP	Real	Paging	SIO	CPU%	ASID	ASIDX	EX
	MQ05BRK	LMEXPERT	BROKER	STC63060	MQ05BRK	STCFAST	COLIN	MVD1		IN	F4	55,031	0.00	3.62	0.03	205	00CD	
	MQ05BRK	MQ05BRK	BROKER	STC63059	MQ05BRK	STCUSER	MQ05	MVD1		IN	EC	30,901	0.00	0.00	0.05	208	00D0	
	MQ05BRK	DAVE	BROKER	STC63061	MQ05BRK	STCUSER	DQ05BASE	MVD1		IN	EC	56,046	0.00	3.62	0.03	240	00F0	

Notes : Workload Management

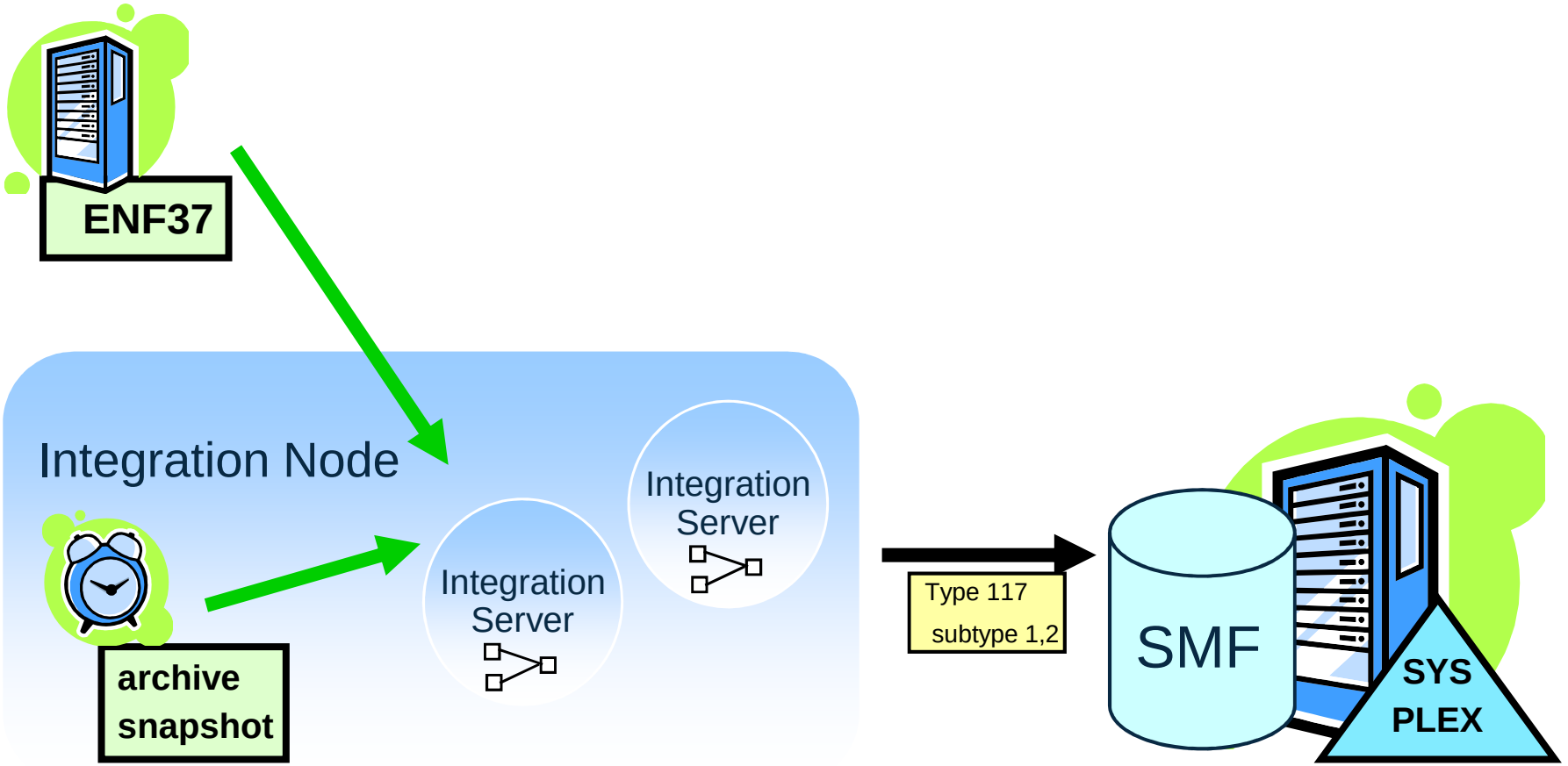
- Goal Oriented Resource Allocation (WLM).
  - When an IBM Integration Bus integration server address space starts, it is assigned a JOBACCT token. This can then be classified in Workload Manager (WLM) to service and report classes. This enables systems programmers to assign different goals (typically response time) to this service class through WLM configuration choices
  - The ability to assign WLM service classes to message processing workload has two significant benefits
    - As work passes through subsystems which are WLM enabled, the service class can be maintained. Resources such as CPU and IO “follow” users’ work requests to make sure these requests meet, if possible, the goals set by WLM
    - WLM classification means that in the event of resource constraint (at peak time for example), high priority workloads can receive appropriate resource to ensure that critical workloads are completed at the expense of less important work
- Workload Scaling.
  - z/OS SYSPLEX brings a huge amount of potential processing capacity to the IBM Integration Bus.
  - Not only are z196 processors extremely powerful on their own, but 96 can be configured in different LPARs across the SYSPLEX configuration.
  - IBM Integration Bus is designed to exploit all of these capabilities without users having to explicitly design their message flows differently .
  - Message flows have a built-in multiprocessing capability.
  - Each *instance* of a message flow can execute on a separate thread (z/OS TCB) allowing the integration server to easily exploit multiple CPUs.
  - If a flow is not able to process enough messages due to CPU constraint, it is a simple step to assign more instances to the message flow to dynamically increase the number of eligible CPUs for processing. This does not involve flow design or outage .
  - And as we’ve seen, it’s simple to scale message flows across the SYSPLEX by deploying to multiple integration servers within the SYSPLEX domain. Again, this reconfiguration process is dynamic and does not require restart .
- Workload Isolation
  - From a storage isolation perspective, integration servers are completely separated, storage in one address space is not visible to another. Moreover, if the event of a failure, an integration server is restarted without affecting any other integration servers owned by the integration node; failures are isolated to integration servers.

Reporting & Chargeback

- **System Management Facilities (SMF)**
  - IBM Integration Bus allows operational staff to control the gathering of SMF data on a per message flow basis. The message flow developer does not need to instrument their design in any way.
  - Enables efficient and effective tuning of message flows.
  - Allows infrastructure departments to charge users according to their utilization of integration node facilities.

- **Coordinated Reporting**
  - SMF also allows subsystems to report performance and accounting information at the same time for a given processing interval.
  - Makes it possible to correlate statistics between IBM Integration Bus, MQ and DB2 subsystems and artefacts.

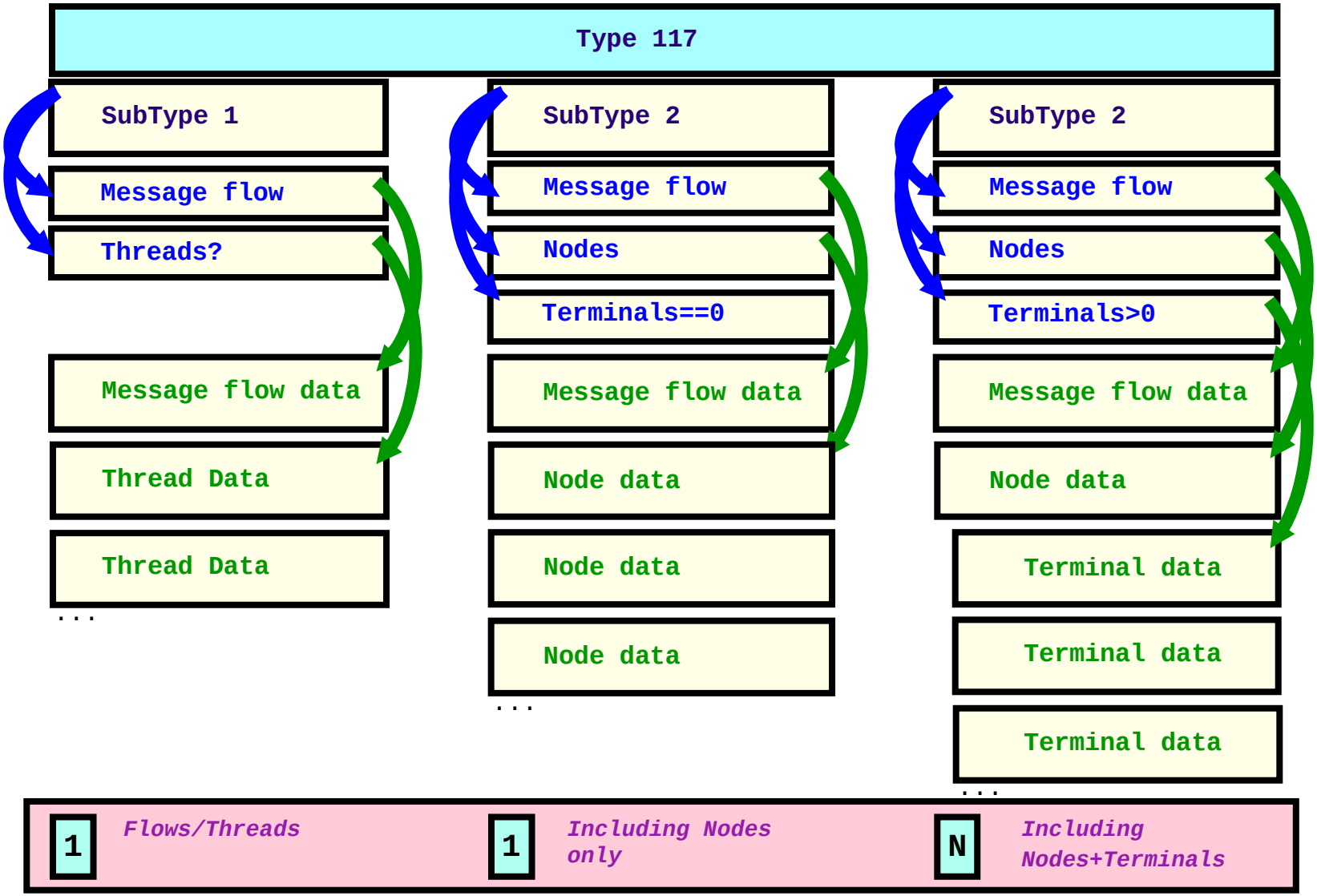
SMF output


ENF37
Integration Node
Integration Server
Integration Server
archive snapshot
Type 117
subtype 1,2
SMF
SYS PLEX

Notes : SMF Output

- Accounting and Statistics Timers
  - Information is gathered at regular intervals according to timers. There are two classes of timers, internal and external.
  - Archive and Snapshot timers are internal timers set by integration node parameters which govern when these data are written to their destinations.
  - An external timer is available on z/OS, namely ENF37. ENF is also important to allow consolidated reporting of SMF information across major subsystems, e.g. you might coordinate queue manager and integration node activity to best understand how to tune your queue manager for particular flows.
- A Variety of Output Destinations and Formats
  - It's possible to gather this information in different formats according to technology used to access it.
  - On z/OS there is the option to generate SMF type 117 records, having subtypes 1 and 2 depending on the granularity of information requested by the user for a particular flow.
  - There is also the option of generating Publish Subscribe (XML) and UserTrace.
  - You may request different output destinations for Snapshot and Archive Stats by Message Flow.
- Options for Reporting Scope and Granularity
  - z/OS SMF can be collected for any integration node within a SYSPLEX. It can be integrated with all other SMF enabled z/OS products and subsystems - literally hundreds!
  - UserTrace is collected for the integration node. This data is human readable and relatively useful if you're prototyping.
  - PubSub can be collected for any integration node throughout the domain. This is incredibly powerful; it means you can sit anywhere in a domain and request information about particular nodes in a flow on a particular integration server on a integration node on a different machine! OR you could put in wild card subscriptions to gather information from several different sources! XML is very structured, which makes it ideal for multi platform reporting. These message can also be consumed and reported using the IBM Integration Bus Explorer.

z/OS SMF Record Structure



BipSmf.h

Notes : z/OS SMF Record Structure

- SMF 117 records describe the A&S information
  - The subtype of SMF 117 records written depends on the data currently being collected.
  - A type1 record is produced when a flow is only collecting message flow data or Threads data. A single type1 record is produced with all threads included.
  - Type2 records are produced when a flow is collecting nodes data. When *only* nodes data is collected, a single type 2 record is written to SMF, whereas when nodes *and* terminals are being collected, multiple type 2 records are written to SMF.
- BipSMF.h describes the SMF records
  - You can use the BipSMF.h header file to produce your own reports.
- The integration node/server userid must be permitted to the BPX.SMF facility class profile to write SMF records.

Accounting Origin

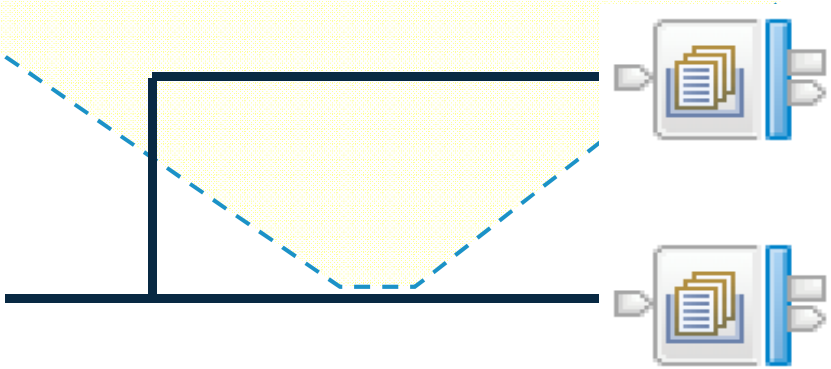
```
SET Environment.Broker.Accounting.Origin =
            InputRoot.XML.DepartmentName; -- Dept A,B,C
```

Dept A

Dept B

Dept C

(Anonymous)



```
<WMQIStatisticsAccounting RecordType="Archive"
  <MessageFlow BrokerLabel="TestBroker3" BrokerUUI
               ExecutionGroupName="default" Execu
               MessageFlowName="ParentFlow" Start
               AccountingOrigin="Dept A"/>
...
```

...

Notes : Accounting Origin

- AccoutingOrigin Allows you to Classify Reports
  - In a consolidated flow, i.e. one being used by several different users or classes of users, it's important to be able to identify the costs associated with a particular user or class of user. From Version 5, you can request Accounting and Statistics reports to be generated to identify the originator of the input messages.
  - This allows integration nodes to have a minimum number of flows for accounting purposes and still chargeback, benefitting from economy of scale, and administrative burden reduction.
  - The foil shows messages originating from three different departments - A, B and C. These messages are all processed by the same message flow, but the gathered reports identify the cost breakdown by originating department.
  - If you examine the report snippet, you can see that the AccountingOrigin tag identifies the report as belonging to "Dept. A".
  - The Accounting origin is set inside the flow when it has determined the origin of the message. Any technique can be used to do determine this; for example a flow might use a field in the MQMD, or a field in the body of the message.
- Messages are Classified by the Flow
  - As a message passes through a message flow, if the flow at some points decides that the message needs to be classified, it may do so.
  - It sets the **`Environment.Broker.Accounting.Origin`** tree element to identify the origin. When the message has been finished with, the data related to its processing is collected separately to messages with different AccountingOrigin.
  - When the message has been processed by the flow the information identifying the origin is stored with all the other information for this origin and separate to information from other origins. Different origins can therefore be collected and reported separately.
- AccountingOrigin needs to be Enabled
  - This function is enabled using a new option on the mqsichangeflowstats command (see later). It is not always enabled because there is extra processing associated with this processing and although not excessive, it would have an impact on performance.
  - You should be aware that enabling this function could generate a large volume of reports, that's because you will get a different report for each AccountingOrigin identified by your flow in a given time period.
  - If the function is enabled, and **`Environment.Broker.Accounting.Origin`** is not populated then the Statistics and Accounting data collected while processing that input message will be accumulated to the default Accounting Origin.
    - The default value for the AccountingOrigin is "Anonymous".
  - If the Accounting Origin function is disabled then the default action is to accumulate all Accounting and Statistics data to the default Origin.

Reduced Cost of Ownership

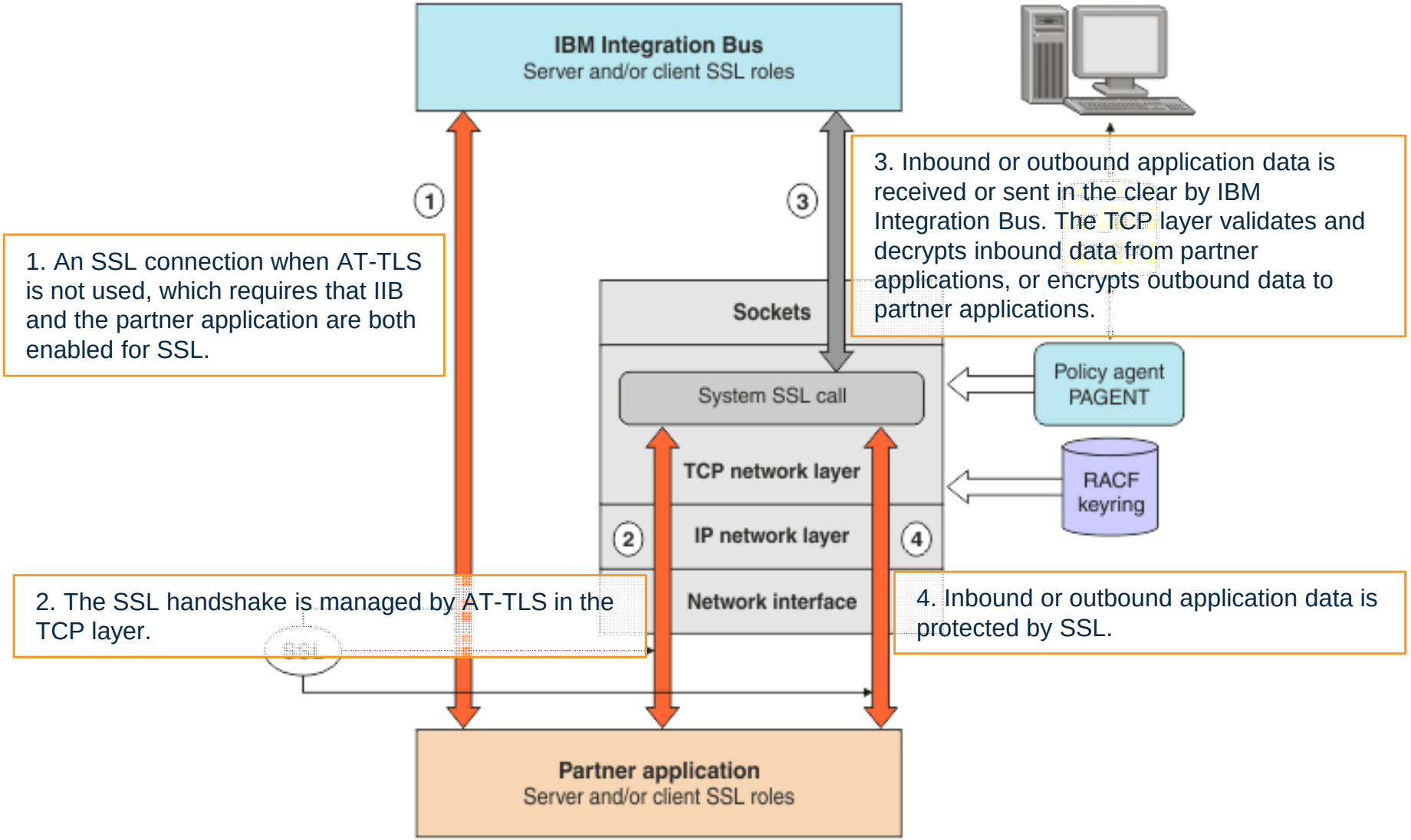
- **zSeries Application Assist Processor (zAAP) Exploitation**
  - Machine instructions generated by the Java Virtual Machine can be offloaded to dedicated processors called zAAPs .
  - IBM Integration Bus has several features which directly exploit zAAP technology, such as the Java Compute node, XLST / JMS and GDM nodes .
  - Message transformations using the above nodes can offload the processing to zAAP's.
- **Standard Edition Pricing**
- **Continuous performance improvements**

Notes : Reduced Cost of Ownership

- zAAP Exploitation.
  - Machine instructions generated by the Java Virtual Machine can be offloaded to dedicated processors called zAAPs.
  - zAAP costs significantly less than regular central processor
  - zAAP capacity is not included in MSU capacity
  - Java based applications can be off-loaded without increase in software costs.
  - IBM Integration Bus has several features which directly exploit zAAP technology, for example the Java Compute node, XLST / JMS and GDM nodes.
  - It should be noted, however, parsing operations are still performed by non Java components and these are not eligible for offload.

Using AT-TLS with Integration Bus

- Use Application Transparent Transport Layer Security (AT-TLS) to provide Secure Sockets Layer (SSL) services to IIB



Notes : Using AT-TLS with IIB

- You can use Application Transparent Transport Layer Security (AT-TLS) to provide Secure Sockets Layer (SSL) services on behalf of IBM® Integration Bus on z/OS®. AT-TLS is part of z/OS Communication Server.
- You can enable SSL by following the instructions in the infocenter to setup a normal JKS keystore PKI. You can also use an alternative method that uses AT-TLS to enable SSL without the need to complete configuration steps in IBM Integration Bus. It is all done at the system level. AT-TLS provides the following benefits when using SSL/TLS protocols with IBM Integration Bus on z/OS:
  - AT-TLS uses RACF® key rings and certificates.
  - The Policy Agent (PAGENT) manages the rules and policies that define how SSL is used to connect to IBM Integration Bus.
  - PAGENT can distribute the rules and policies in a z/OS SYSPLEX environment.
  - HTTP or SOAP nodes in message flows can have standard HTTP settings (no SSL/HTTPS).
- To configure AT-TLS in your z/OS environment for IBM Integration Bus you need to complete the following steps:
  1. Create a RACF key ring by following the instructions in Creating a RACF key ring.
  2. Configure and activate PAGENT by following the instructions in Configuring and activating the policy agent (PAGENT).
  3. Define and install AT-TLS policies for IBM Integration Bus by following the instructions in Defining and installing AT-TLS policies.
  4. Test and verify AT-TLS by using IBM Integration Bus, as described in Testing and verifying AT-TLS.
- AT-TLS is a service provided by the z/OS Communication Server Policy Agent (PAGENT) and the TCP/IP stack. The AT-TLS service manages SSL connections on behalf of applications that are running on z/OS. The z/OS applications are unaware that SSL is used in the connection with partner applications.
- To create a RACF key ring, you must first generate a RACF CA certificate and a personal certificate for IBM Integration Bus, then connect the certificates to the key ring.
- Configure PAGENT by updating the TCP/IP profile, granting RACF permission to TCP/IP resources, preparing the PAGENT startup JCL, and activating syslogd.
- Define and install AT-TLS policies by using the IBM Configuration Assistant for z/OS Communications Server.
- Test and verify AT-TLS by using the SOAP Nodes sample.
- To help with problem determination with PAGENT and AT-TLS, activate tracing and logging.

z/OS Specific Nodes

- **QSAM Nodes (IA11)**
  - These are similar in concept and usage to the VSAM nodes, but oriented around sequential files, rather than record oriented files.
    - FileDelete
    - FileRead
    - FileRename
    - FileWrite

- **VSAM Nodes (IA13)**
  - A suite of nodes allowing users to perform record oriented processing on VSAM files.
    - VSAMDelete
    - VSAMInput
    - VSAMRead
    - VSAMUpdate
    - VSAMWrite

Notes : z/OS Specific Nodes

- VSAM Nodes
  - The broker can process VSAM records in the same way as it processes messages from or to other data sources.
  - A suite of 5 nodes allowing users to perform record oriented processing on VSAM files: Input, Read, Write, Update and Delete operations.
  - Users can combine these nodes for VSAM access with other nodes to integrate VSAM processing into message flows, or use VSAM files to drive message flow processing

- QSAM Nodes
  - These are similar in concept and usage to the VSAM nodes, but oriented around sequential files, rather than record oriented files.

Summary

- Runtime environment for integration
- Functionally complete and consistent
- z/OS and z/Series exploitation
- Extensive collateral technology
- Significant improvements with each version
- Advanced and mature platform

This was Session # 17065. The rest of the week ……

	Monday	Tuesday	Wednesday	Thursday	Friday
08:30			17060: Understanding MQ Deployment Choices and Use Cases	17051: Application Programming with MQ Verbs [z/OS & Distributed]	16544: Why Shouldn't I Be Able To Open This Queue? MQ and CICS Security Topics Room: Willow B
10:00	17036: Introduction to MQ - Can MQ Really Make My Life Easier? [z/OS & Distributed]		17052: MQ Beyond the Basics - Advanced API and Internals Overview [z/OS & Distributed] 17035: MQ for z/OS, Using and Abusing New Hardware and the New V8 Features [z/OS] Room: Willow B	17054: Nobody Uses Files Any More do They? New Technologies for Old Technology, File Processing in MQ MFT and IIB [z/OS & Distributed]	17057: Not Just Migrating, but Picking up New Enhancements as You Go - We've Given You the Shotgun, You Know Where Your Feet Are [z/OS & Distributed]
11:15	17041: First Steps with IBM Integration Bus: Application Integration in the New World [z/OS & Distributed]		16732: MQ V8 Hands- on Labs! MQ V8 with CICS and COBOL! MQ SMF Labs! Room: Redwood	17046: Paging Dr. MQ - Health Check Your Queue Managers to Ensure They Won't Be Calling in Sick! [z/OS]	17053: MQ & DB2 – MQ Verbs in DB2 & InfoSphere Data Replication (Q Replication) Performance [z/OS]
01:45	17037: All About the New MQ V8 [z/OS & Distributed]	17034: MQ Security: New V8 Features Deep Dive [z/OS & Distributed]	17040: Using IBM WebSphere Application Server and IBM MQ Together [z/OS & Distributed]	17062: End to End Security of My Queue Manager on z/OS [z/OS]	All sessions in Seneca unless otherwise noted.
03:15	17042: What's New in IBM Integration Bus [z/OS & Distributed]	17065: Under the hood of IBM Integration Bus on z/OS - WLM, SMF, AT-TLS, and more [z/OS]	17043: The Do's and Don'ts of IBM Integration Bus Performance [z/OS & Distributed]	17039: Clustering Queue Managers - Making Life Easier by Automating Administration and Scaling for Performance [z/OS & Distributed]	
04:30	17059: IBM MQ: Are z/OS & Distributed Platforms like Oil & Water? [z/OS & Distributed]	17055: What's the Cloud Going to Do to My MQ Network?	17044: But Wait, There's More MQ SMF Data Now?!?! - Monitoring your Channels Using V8's New Chinit SMF Data [z/OS]	17068: Monitoring and Auditing MQ [z/OS & Distributed]	

Questions?

Complete your session evaluations online at www.SHARE.org/Seattle-Eval

•Trademark Statement

- IBM and the IBM logo are trademarks of International Business Machines Corporation, registered in many jurisdictions. Other marks may be trademarks or registered trademarks of their respective owners.
- Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
- Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.
- Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
- UNIX is a registered trademark of The Open Group in the United States and other countries.
- Netezza® is a trademark or registered trademark of IBM International Group B.V., an IBM Company.
- Worklight® is a trademark or registered trademark of Worklight, an IBM Company.
- Other company, product and service names may be trademarks, registered marks or service marks of their respective owners.
- References in this publication to IBM products and services do not imply that IBM intends to make them available in all countries in which IBM operates.