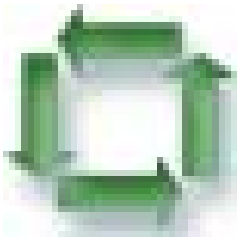
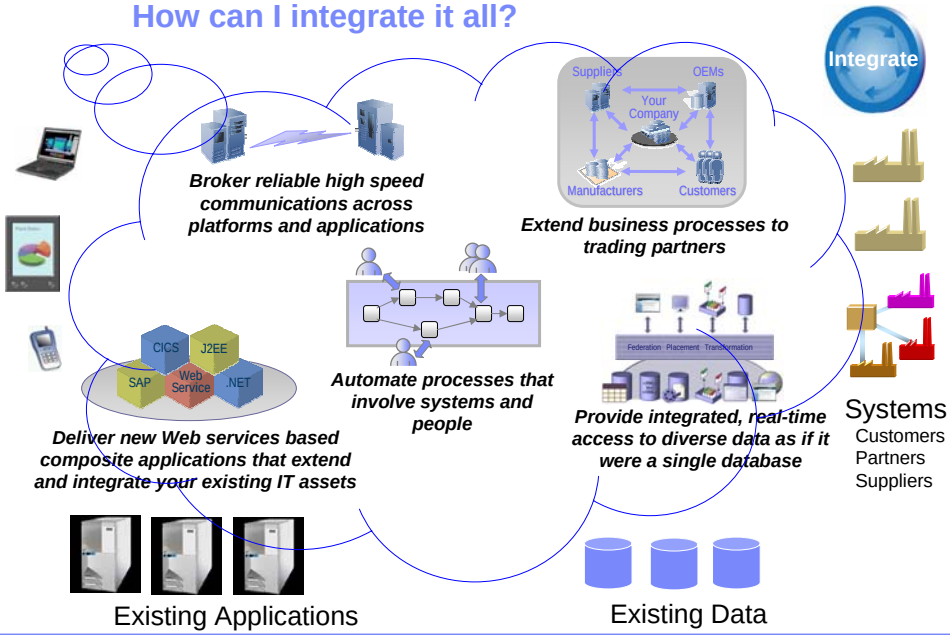


WebSphere Business Integration Server Foundation for z/OS

Stephen Matulevich

Washington Systems Center

How can I integrate it all?



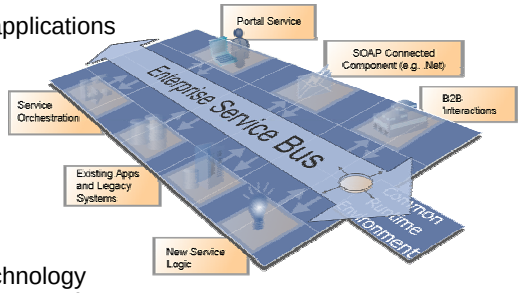
A Service Oriented Architecture is Key

The flexibility to treat business processes and the underlying infrastructure as defined components that can be mixed and matched at will

What is SOA?

SOA enables flexible connectivity of applications or resources by:

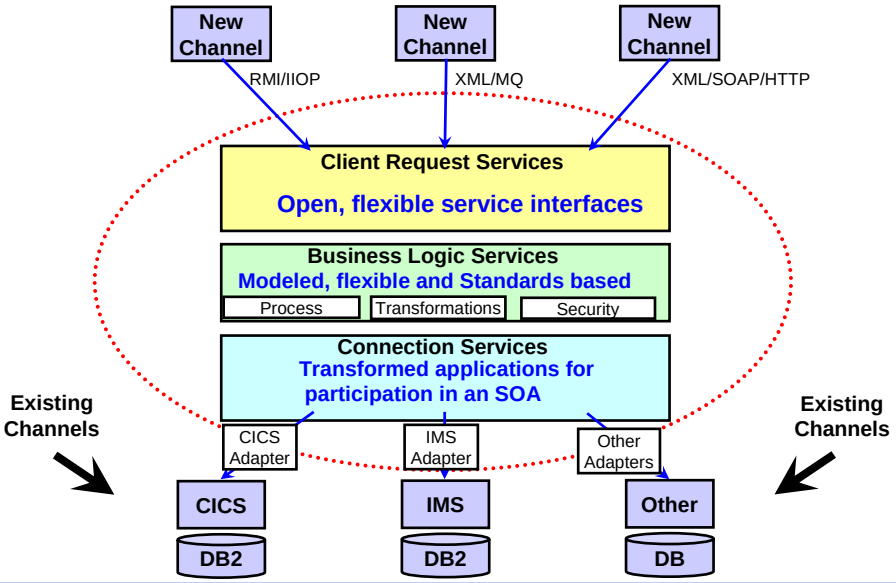
- Representing every application or resource as a service with a standard interface
- Enabling them to exchange structured information



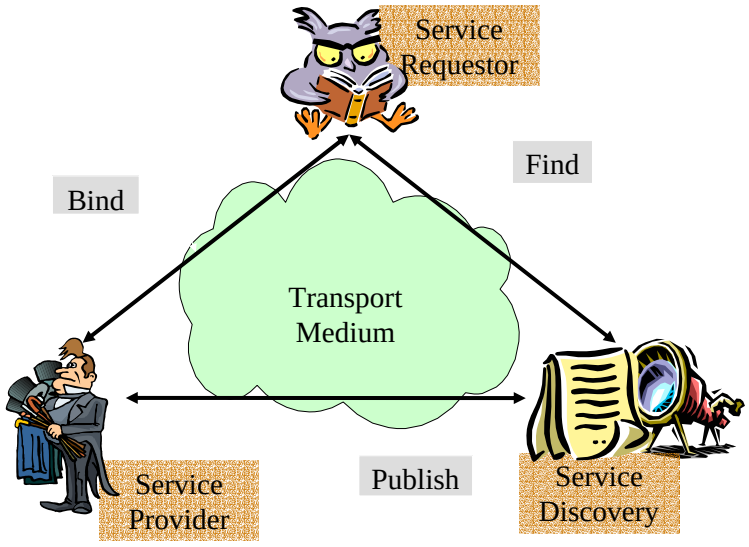
Why do you care?

SOA helps introduce flexibility in a technology environment. There is growing acceptance of SOA as an approach to integration and to structure collections of interacting applications.

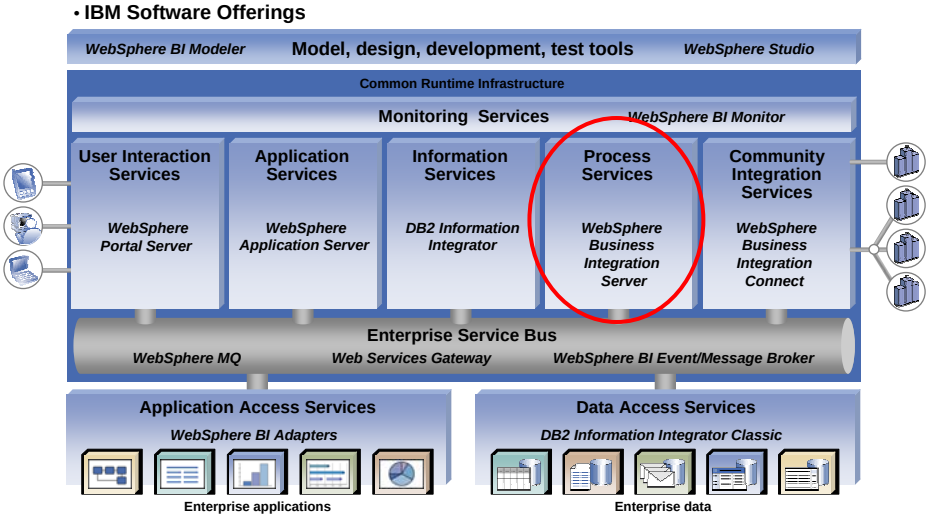
A model of an SOA



Service Oriented Architecture (SOA)

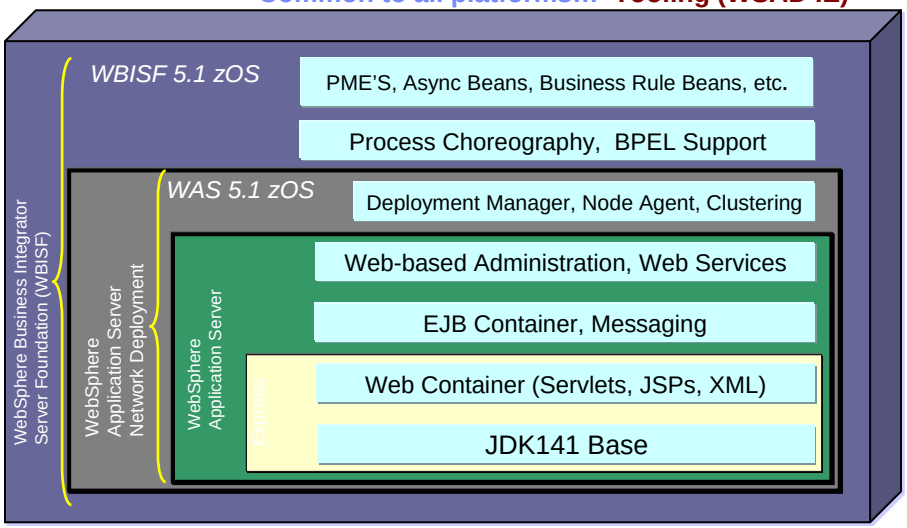


Business Integration Reference Architecture



WBISF for z/OS Server Structure

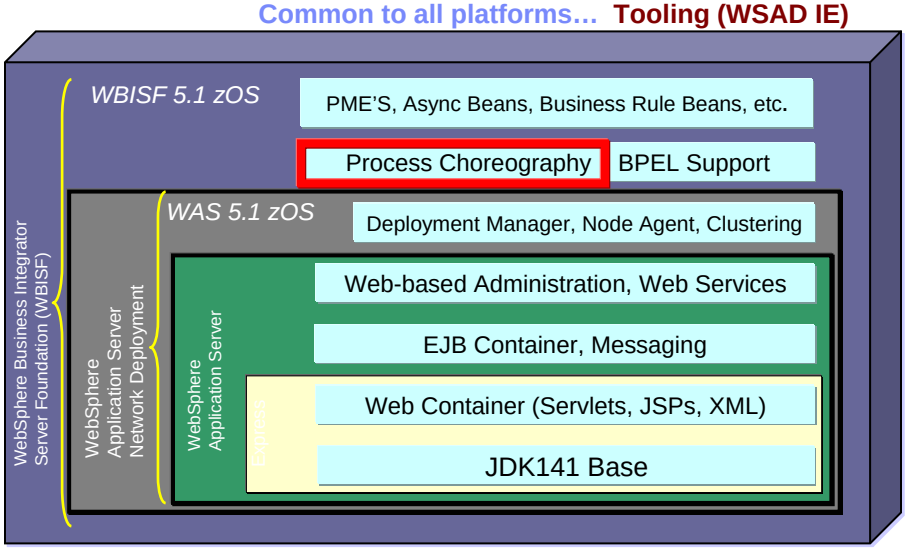
Common to all platforms… Tooling (WSAD IE)



IBM Washington Systems Center

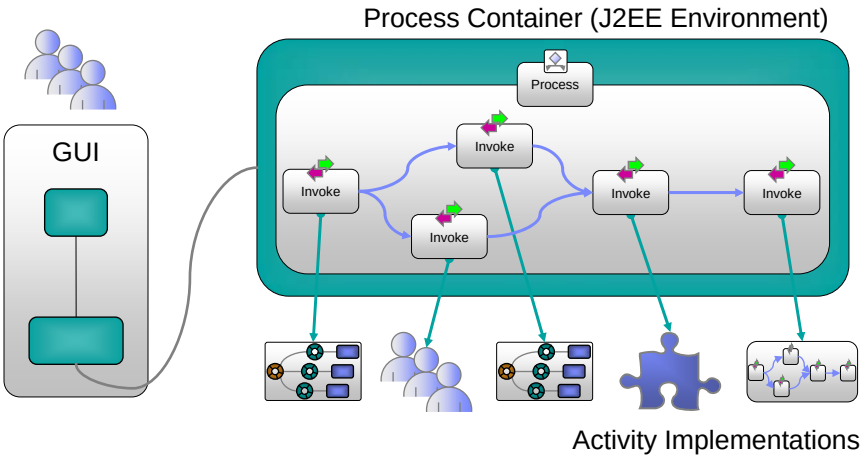
WBISF for z/OS Server Structure

Common to all platforms… Tooling (WSAD IE)

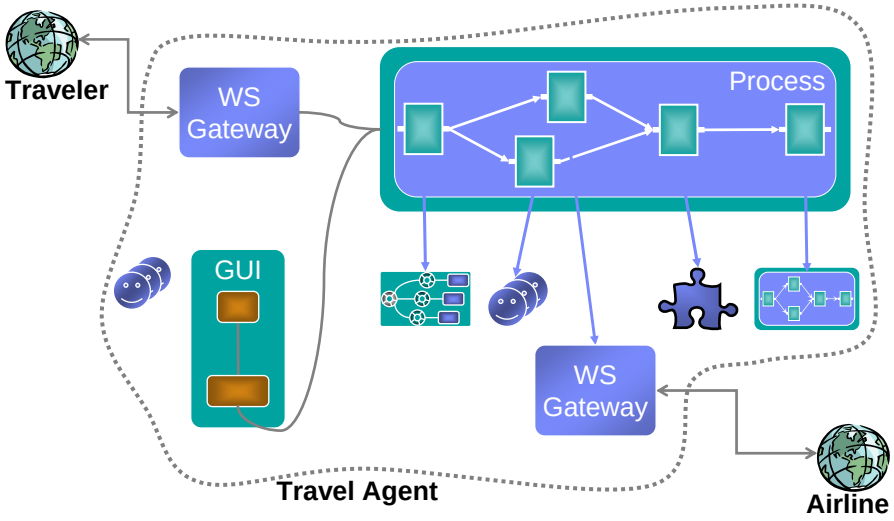


© 2004 IBM Corporation

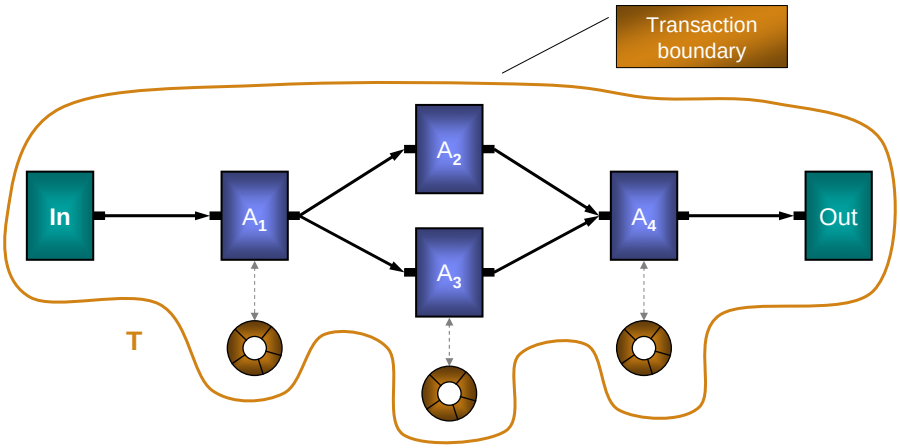
Workflow-based Application



Services-Oriented Process



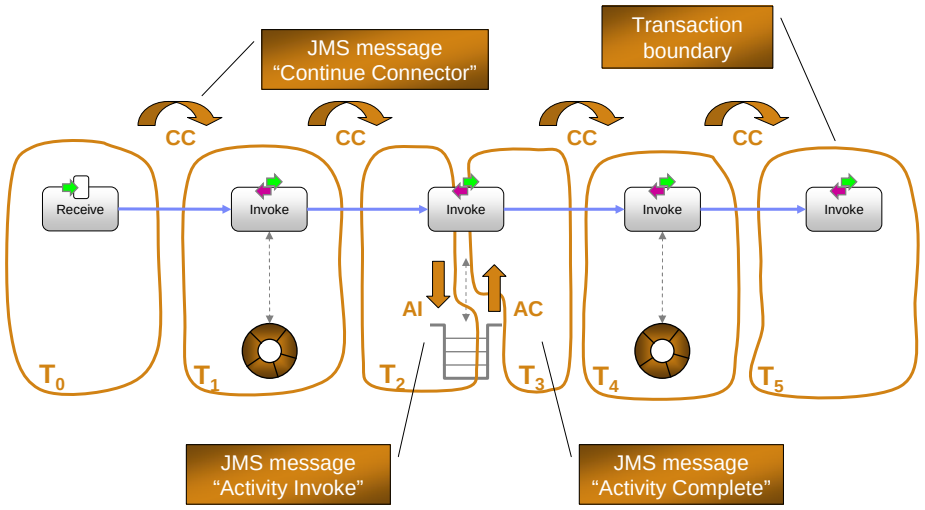
Transactions: Microflow



Characteristics of a Microflow

- Runs in memory on one thread
- Runs as part of a single transaction
- Non-interruptible
- No database / message queueing system involved by the process engine
  - Transient navigation triggers
  - In-memory process state
- Fast !!!
  - Rule of thumb: 10 – 100 times faster navigation performance compared with a long-running process

Transactions: Macroflow (Long-running Process)



Characteristics of a Long-Running Process

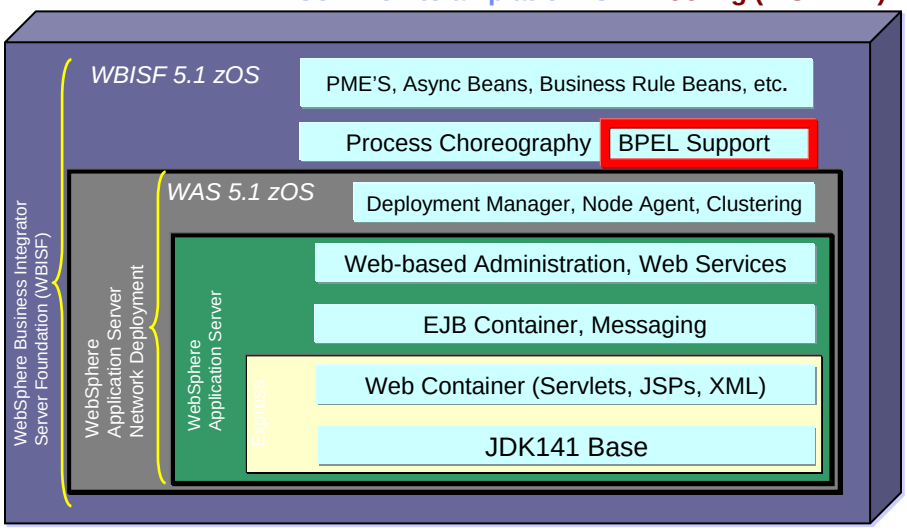
- Run as a set of stratified transactions
  - Each step in the process is fully recoverable
  - Process navigation involves database read/writes and interaction with persistent message queues (get and put)
- Supports all types of activities, including the ones not supported by microflow processes:
  - Basic activities with asynchronous activity implementation
  - Activity implementations that involve human interaction
  - Pick activities
- Supports compensation
- Supports parallel execution of activities
- Provides load balancing
  - Within one application server by dispatching process flow via JMS
  - Between application servers when using MQ clustered queues

Process Choreographer 5.1 – Quick Facts

- Provides choreography for
  - Web Services
  - Local J2EE Components (e.g. EJB, Java Class, other processes)
  - Adapters
  - People based steps
- Integral Module of WebSphere Application Server (J2EE Application)
- J2EE based Process Engine
- Exploiting WAS Base features, e.g. Clustering, Security, Admin
- Customizable Web-based Client

WBISF for z/OS Server Structure

Common to all platforms… **Tooling (WSAD IE)**



What is BPEL4WS?

- Business Process Execution Language for Web Services
- A language to specify behavior of business processes
  - between Web services
  - and as Web services
- Published by IBM, Microsoft, and BEA (+ SAP and Siebel)
- OASIS (Organization for the Advancement of Structured Information Systems) promotes BPEL4WS
- IBM Extensions to BPEL4WS
- Key Value
  - Industry wide language for business processes (common programming skill)
  - An important bridge between the J2EE and the .Net worlds
  - Portable business processes (run in engines from different vendors)
  - Choice of process engines (consumer can “shop” for BPEL engines)

BPEL is Standards-Based

- BPEL itself submitted to standards body
- Based on WSDL and other XML standards
  - WSDL defines interface of composed service and services used by composite
  - XML Schema and XPath for data context handling and business rules specification

BPEL4WS (Business Process Execution Language For Web Services)

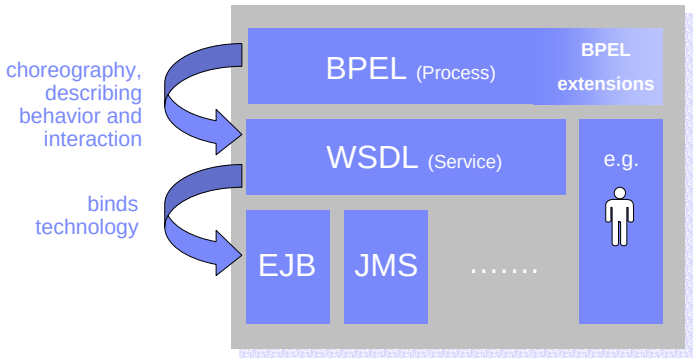
WSDL (Web Services Description Language)

XPath (XML Path Language)

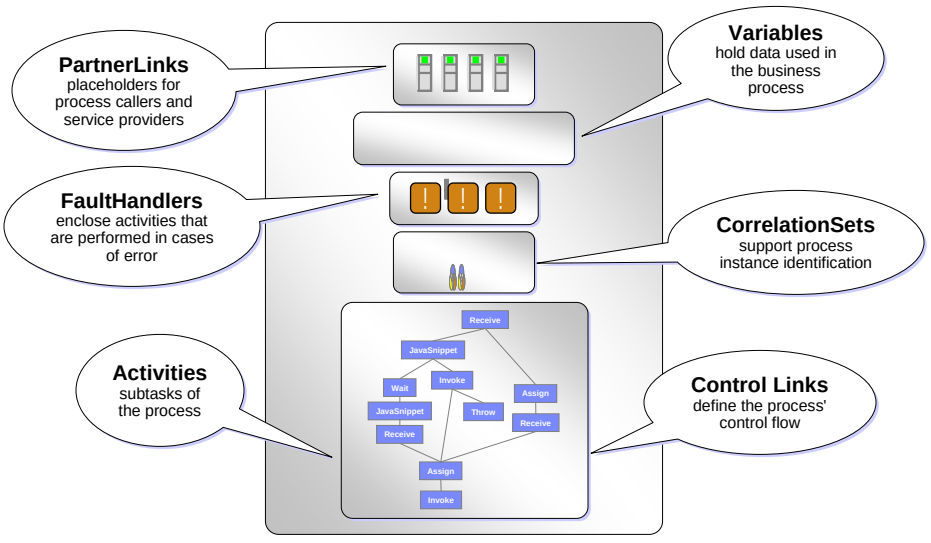
XSD (XML Schema)

XML

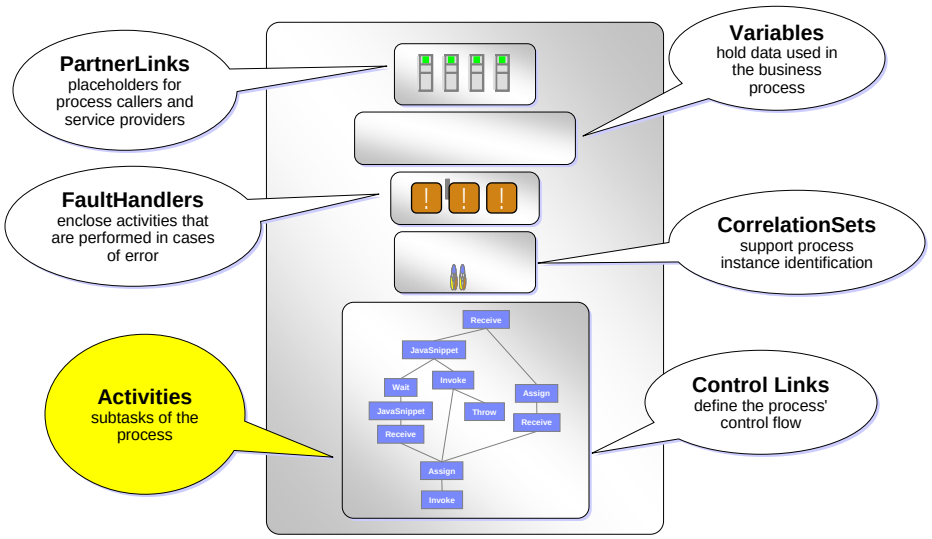
BPEL4WS – Conceptual View



Elements of a BPEL4WS process

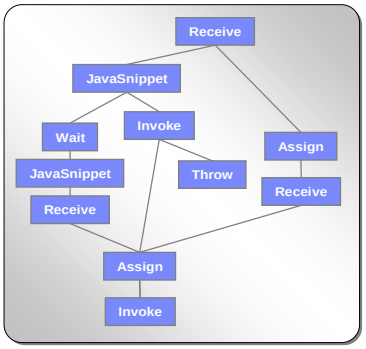


Elements of a BPEL4WS process



BPEL Activities

- Basic activities allow to define the tasks that make up the business process
- Structuring activities help to define the control flow of the business process



BPEL Basic Activities

Receive
Wait for a message to arrive. Optionally start a new process instance when the message arrives.

Pick
Wait for one of multiple messages to arrive or for a time-out alarm to go off.

Reply
Send a message in reply to a message that was received through a Receive.

Empty
A "no-op" instruction in the business process.

Invoke
Invoke a one-way or a request-response operation offered by a partner.

Terminate
Immediately terminate the process instance.

IBM Washington Systems Center

BPEL Basic Activities

Assign
Update the values of variables with new data.

Staff
Invoke an interaction with a human user.
BPEL Extension

Wait
Wait for a given time period or until a certain time has passed.

JavaSnippet
Invoke an inline snippet of Java code.
BPEL Extension

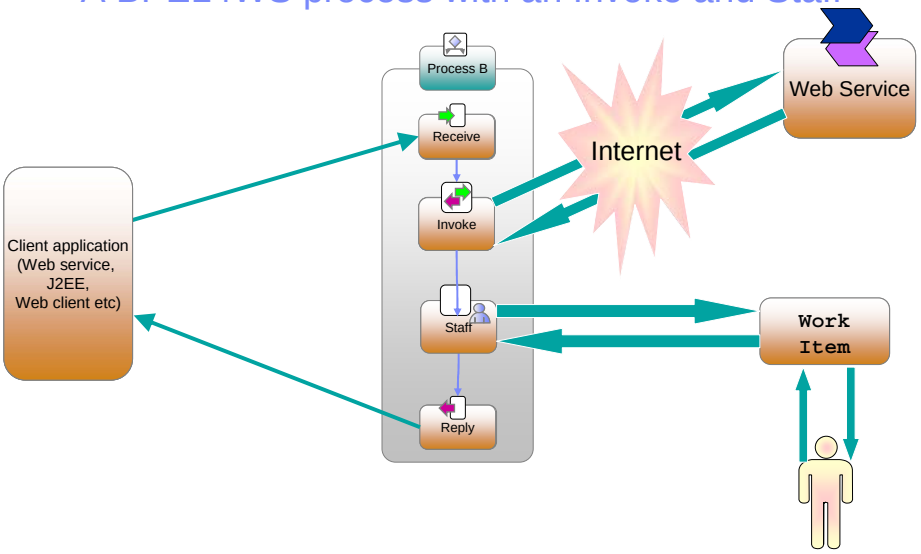
Throw
Generate a fault from.inside the business process.

Transformer Service
Invoke a previously defined message transformation
BPEL Extension

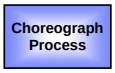
Staff Activity (BPEL Extension)

- Represents step in the process that is performed by a person
- Associated with a WSDL port type & operation
- People are assigned via staff queries
  - Resolved at runtime against enterprise directory
  - Potential owners, Editors, Readers
  - Example: `managerOf(%process.starter%)`
- GUI rendering details

A BPEL4WS process with an Invoke and Staff



Staff Support Activity



- **Business Process Web Client**
  - A Web application that interacts with the Process
    - Easy to customize
  - People can see and control the status the tasks assigned them
- **Staff Directory**
  - Metadata - organizational structure
  - Data - the people and the activities they can perform
- **Staff Plug-in Provider**
- Defines how a Business Process interacts with people
  - A flow engine plug-in for interaction with a ***Staff Directory***

Specify JSPs to Customize

Web Client Settings:

Name	Value	Context Root
InputMessageJSP	/LoanOfficerInput.jsp	/BPEWebClient
OutputMessageJSP	/LoanOfficerOutput.jsp	/BPEWebClient
MessageMappingJSP	/LoanOfficerMM.jsp	/BPEWebClient
ReplaceJSP	<default>	

Customized Web Client

Customer 1 has credit rating of 'B' and requested a loan of $75000.0.

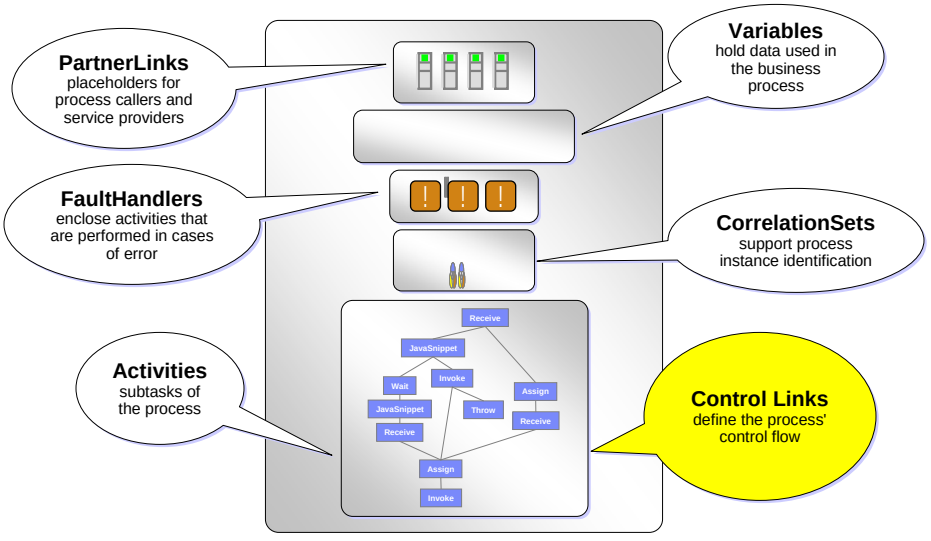
Please approve or reject the loan for customer 1.
Customer Credit Query - For more information you can query this customer's credit history.

- Approve the loan.
- Reject the loan.

Staff Plugin Providers:

Name	Description
User Registry Staff Plugin Provider	This staff plugin provider can be used for User Registry based staff queries.
LDAP Staff Plugin Provider	This staff plugin provider can be used for LDAP based staff queries.
System Staff Plugin Provider	This staff plugin provider may be used for System based staff queries.

Elements of a BPEL4WS process



BPEL Structured Activities

Flow
Holds multiple activities that are performed concurrently.

Link
Synchronizes two activities that are enclosed in a Flow (i.e, enforces a certain execution order).

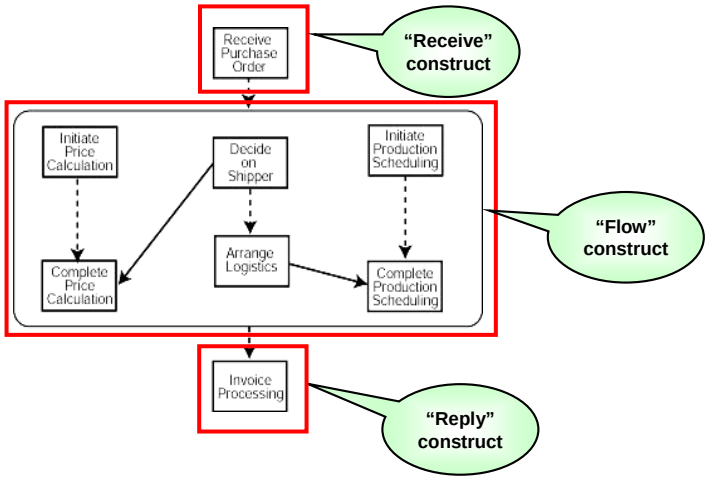
Sequence
Holds multiple activities that are performed sequentially in lexical order.

Switch
Selects one branch of activity from a set of choices.

While
Holds a (basic or structuring) activity that is repeated until a success criteriy has been met.

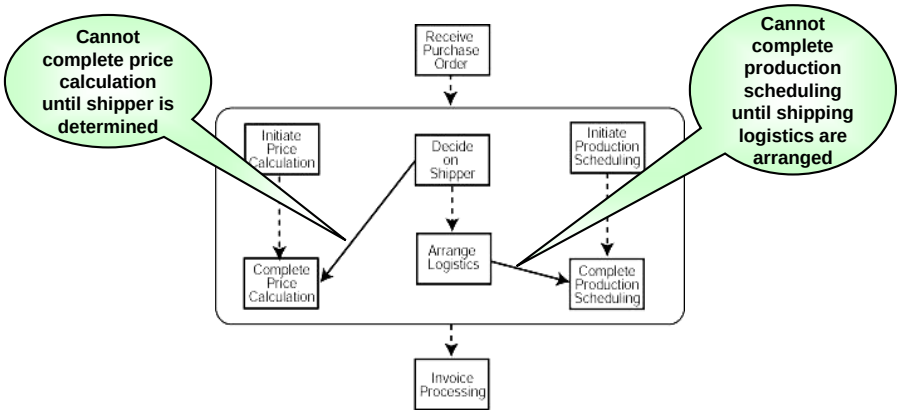
The “receive”, “flow” and “reply” constructs are the main BPEL4WS constructs used to represent process flows

- The purchase order example uses **all three constructs**

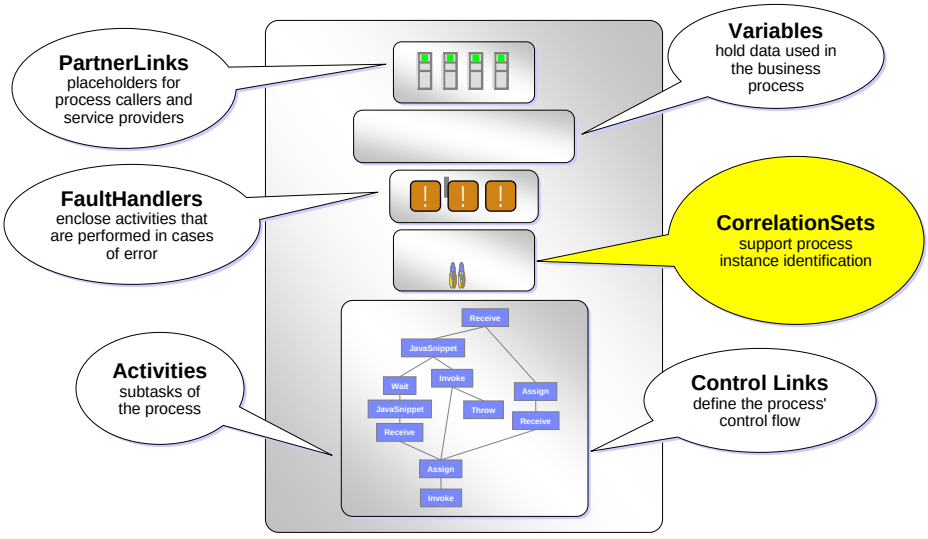


BPEL4WS is also capable of modeling dependencies between activities

- There are **several dependencies** in the purchase order example



Elements of a BPEL4WS process

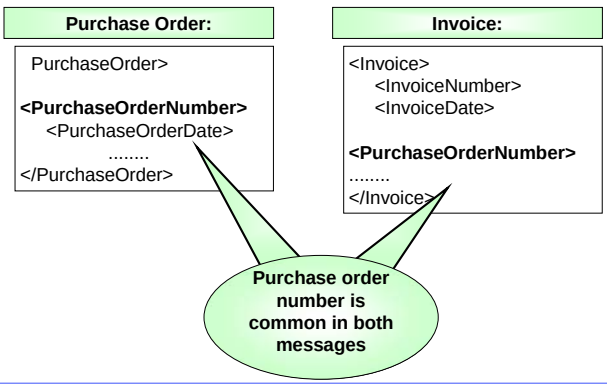


CorrelationSets

- Defined by groups of properties
- Used between partners to identify process instances
- Values of a correlation set initialized once, thereafter associated with process instance
- Properties in incoming messages later used to correlate message with process instance

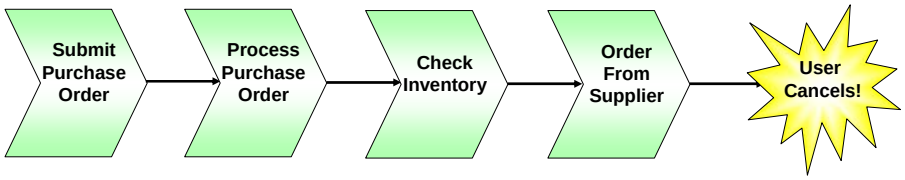
Message correlation involves the association of two or more messages with each other in an asynchronous environment

- This may be done by **associating contents** in a given message with its correlating message
  - For example, in a purchase order/invoice scenario, the invoice may contain the corresponding **purchase order number**



Business processes are often of long duration, which means that a business process may need to be cancelled after many transactions have been committed during its progress

- Consider a situation in which a user **cancels a purchase order:**

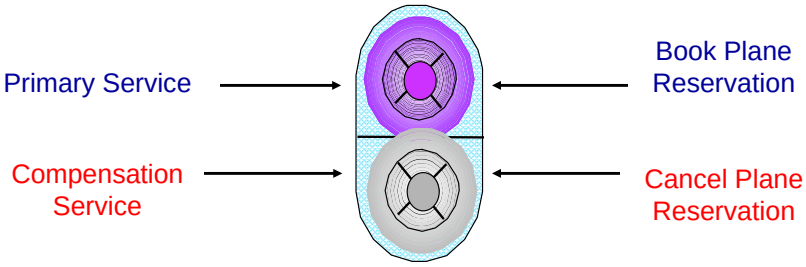


Revert back to original state

- In this situation, it is **not possible to lock system resources** (ex: database records) for extended periods of time.
  - Therefore, the **partial work must be undone** as best as possible

Compensation Pairs

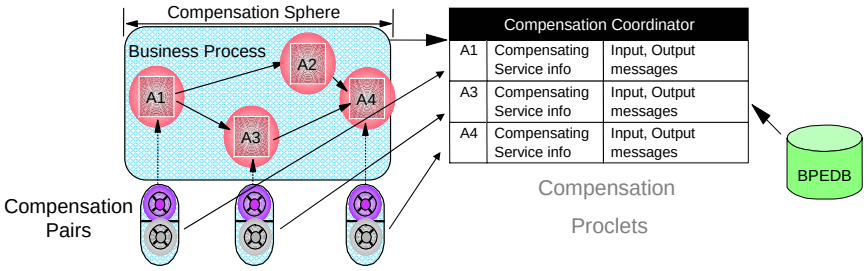
- With compensation, for a given business service activity, you define an 'undo' business service.
- These two services are associated into a pair called a Compensation Pair.



- Compensation Service is transparent to normal Business Process execution,.

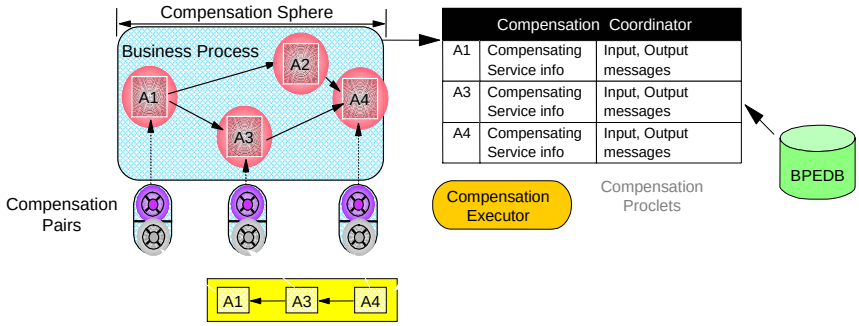
Compensation in a Business Process

- When a Business Process starts, a **Compensation Coordinator** is created.
- The Compensation Coordinator creates a **Compensation Sphere** of possible **Compensation Pairs**.
- During execution: When an activity is reached which has a compensation service (pair) defined, a Compensation Proclet is registered with the Compensation Coordinator



Compensation in a Business Process

- When a Process starts, a **Compensation Coordinator** is created
- After an Activity is executed Compensation Coordinator creates **Compensation Proclet** and stores it in BPEDB database
- If a fault is thrown and unhandled, **Compensation Executor** is created
- Compensation Executor calls compensating services from registered proclets in Last In-First Out sequence
- Process instance marked **COMPENSATED**



Compensation …

Compensation support is available for <u>both</u> Long-running and Short-running Processes

- Compensation is triggered if a fault is thrown and unhandled
- The Compensation Sphere spans all sub-processes of the overall process
  - Child Processes share Compensation Spheres of top-level Process

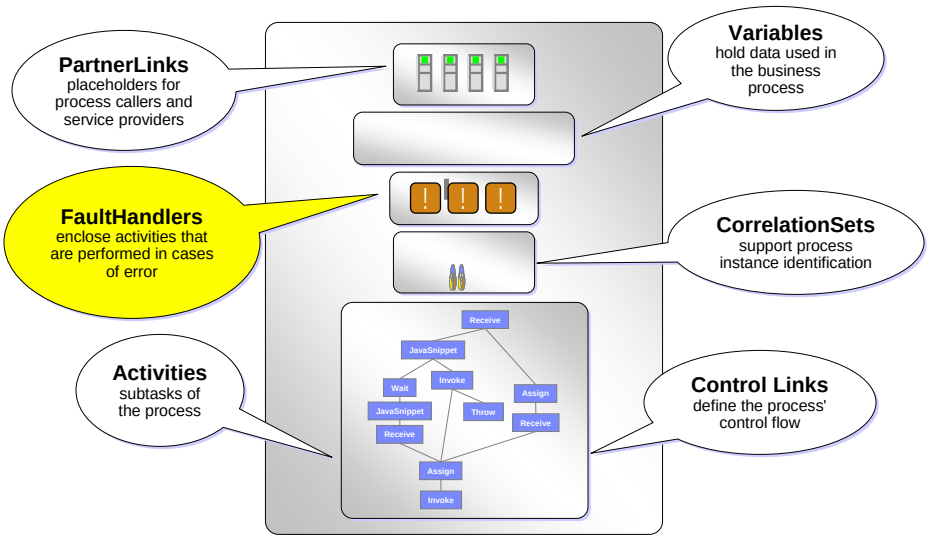
Compensation for Short-running

- Compensation is triggered on rollback of the work unit

If an error occurs during compensation processing, the compensation action requires manual resolution to overcome the error

- You can use the Process Web Client to repair these compensation actions in error.

Elements of a BPEL4WS process



Fault Handler

- Associated with a scope
- Handles faults thrown in their scope.
- A fault reaching a fault handler means that regular processing within this part of the business process can no longer be successfully completed.
  - All active work within the scope will be stopped
- Specifies activities that must be performed if a fault has been thrown.
- Allows association of different fault handling activities with different kind of faults (Catch blocks for different faults).

Fault Handler

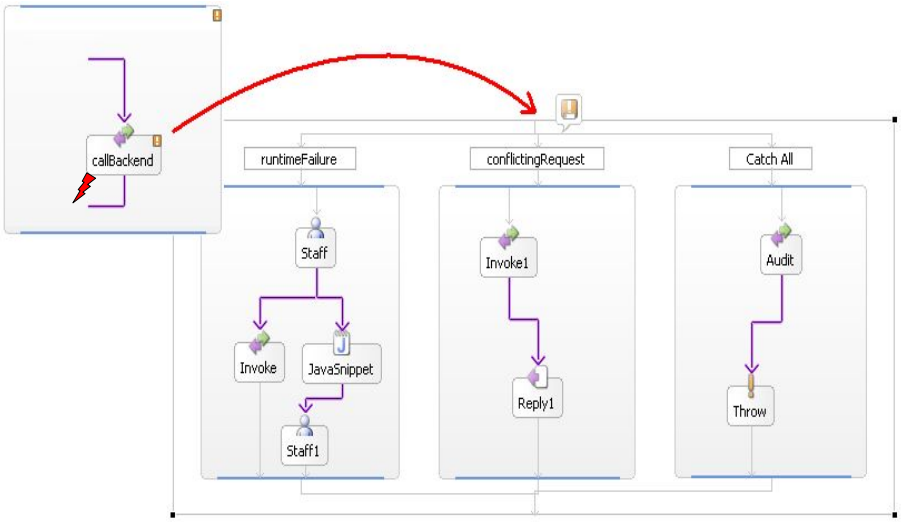
If a Fault has occured in a Scope...

(1)FaultHandler is defined and Fault is caught
- → Scope in state FAILED
- → Fault handling (as specified in catch block)
- → Enclosing scope continues with normal processing

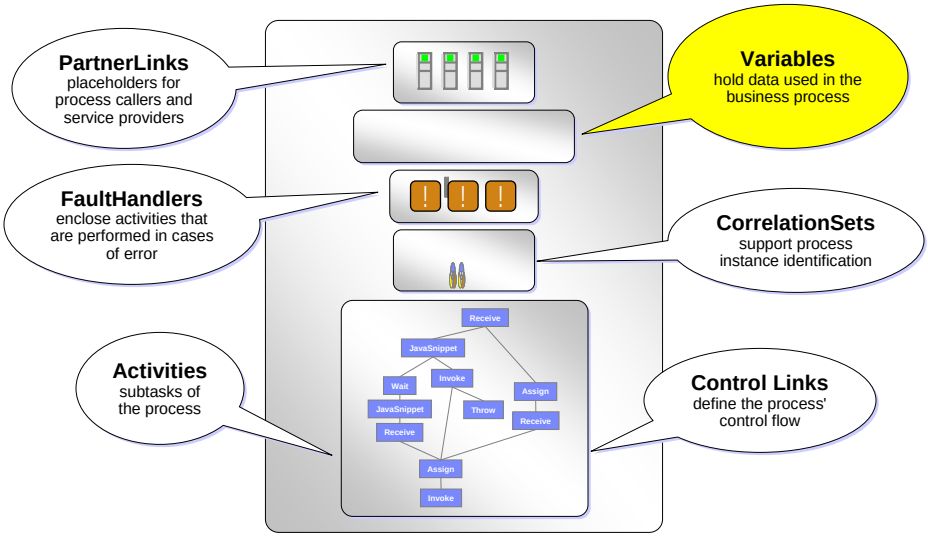
(2)FaultHandler is defined, but Fault is not caught
- → Scope in state FAILED
- → Fault rethrown to next enclosing scope

(3)No FaultHandler defined
- → Same behavior as in (2)

Process Fault Handling

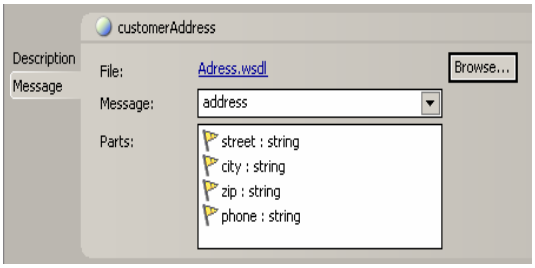


Elements of a BPEL4WS process

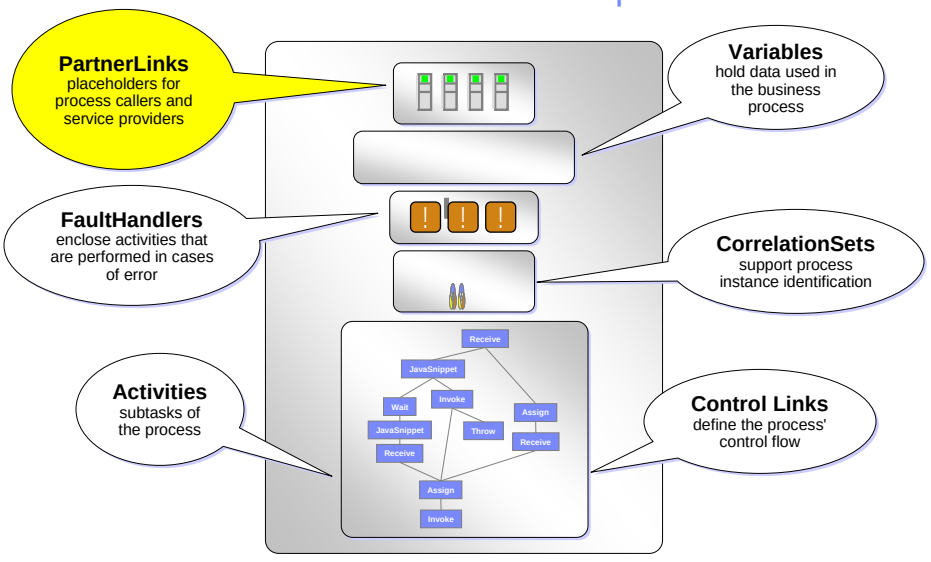


Variables

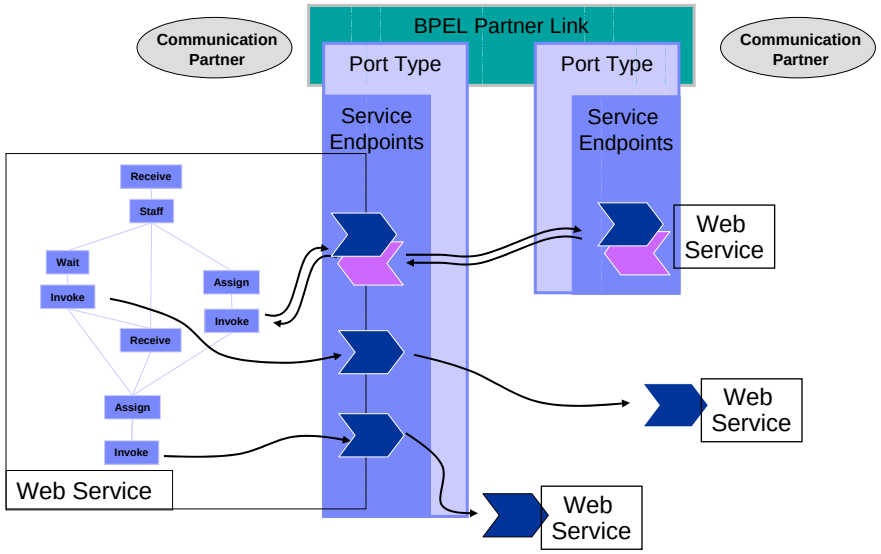
- Hold data that constitute the state of a process
  - May be received from or sent to partners
  - Can be specified as input or output variables for invoke, receive, and reply activities
  - May hold state data related to the process and never exchanged with partners
- Associated with WSDL message types



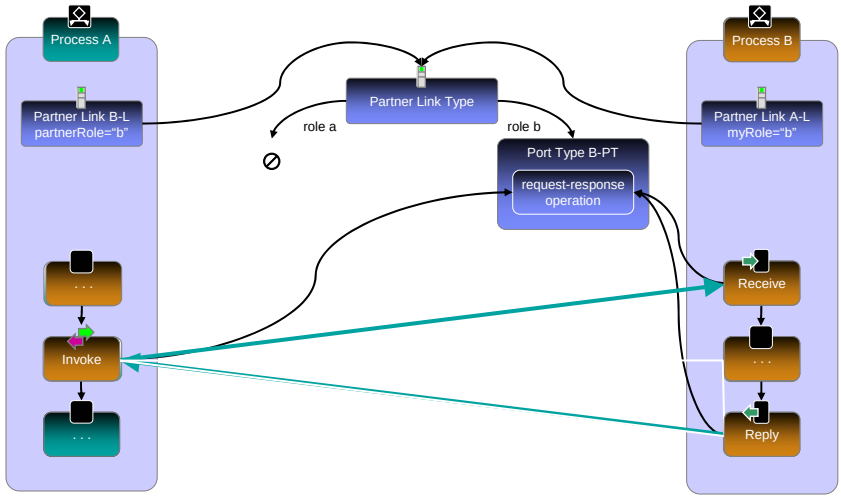
Elements of a BPEL4WS process



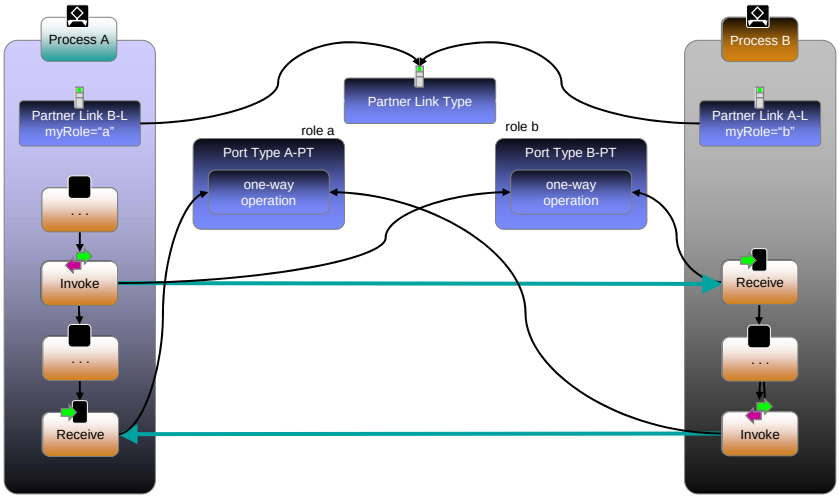
Web Services Choreography with BPEL4WS



BPEL Invocation Scenario: Synchronous Interface

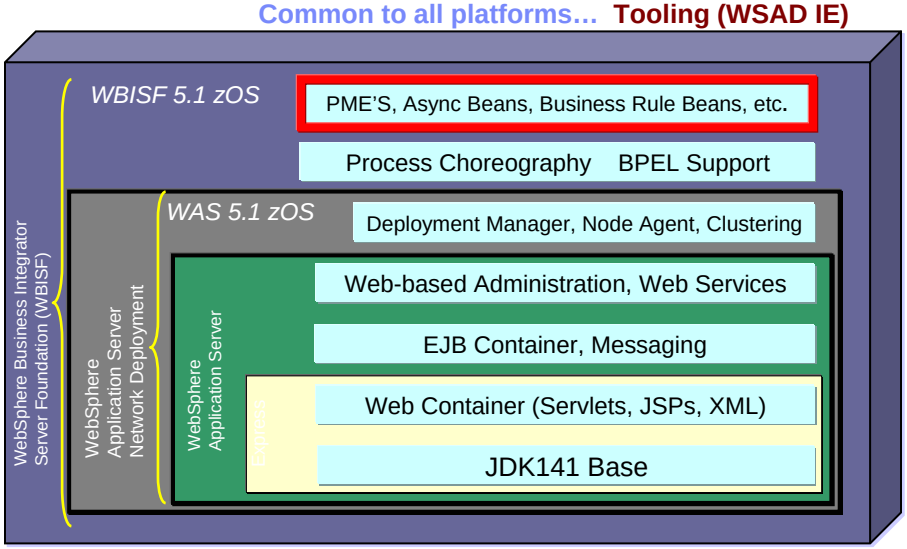


BPEL Invocation Scenario: Asynchronous Interface



WBISF for z/OS Server Structure

Common to all platforms… Tooling (WSAD IE)



Programming Model Extensions (PMEs)

- Programming Model Extensions address:
  - Integrated J2EE based workflow
  - Advanced Transactional Connectivity
  - Accelerate large-scale application development
  - Optimize application performance
  - Increase Development Productivity
  - Enable real-time application flexibility

IBM Washington Systems Center

Programming Model Extensions (PMEs)

- Integrated J2EE based workflow
  - Business Process Choreographer
  - Human Interaction
  - Event Triggering
  - Compensation Pairs
  - Flexible Workflow Design

Programming Model Extensions (PMEs)

- Advanced Transactional Connectivity
  - **Dynamic application adapter support** – Offers the ability to build and deploy rich, open standards-based application adapters for popular Enterprise Information Systems such as SAP and IBM CICS
  - **Last participant support** - provides automated coordination for transactions that include two-phase commit resources and a single one-phase commit resource
  - **Activity session services** - provides the ability to extend the scope of and group multiple local transactions

IBM Washington Systems Center

Programming Model Extensions (PMEs)

- Accelerate large-scale application development
  - WBISF V5.1 and WSAD IE V5.1 leverage the latest innovations that build on today's J2EE standards to help deploy a high performance e-business infrastructure designed to cut costs, build customer loyalties, promote business agility, and gain a competitive advantage.

Programming Model Extensions (PMEs)

- Optimize application performance
  - **Asynchronous beans** - enables J2EE applications to decompose operations into parallel tasks in order to speed performance
  - **Startup beans** - enables J2EE applications to execute business logic automatically, whenever an application starts or stops normally
  - **Scheduler service** - enables tasks to be executed at a requested time; when used in conjunction with asynchronous beans, it enables batch processing applications within J2EE
  - **Object pools** - enables an application to avoid creating new Java objects repeatedly

IBM Washington Systems Center

Programming Model Extensions (PMEs)

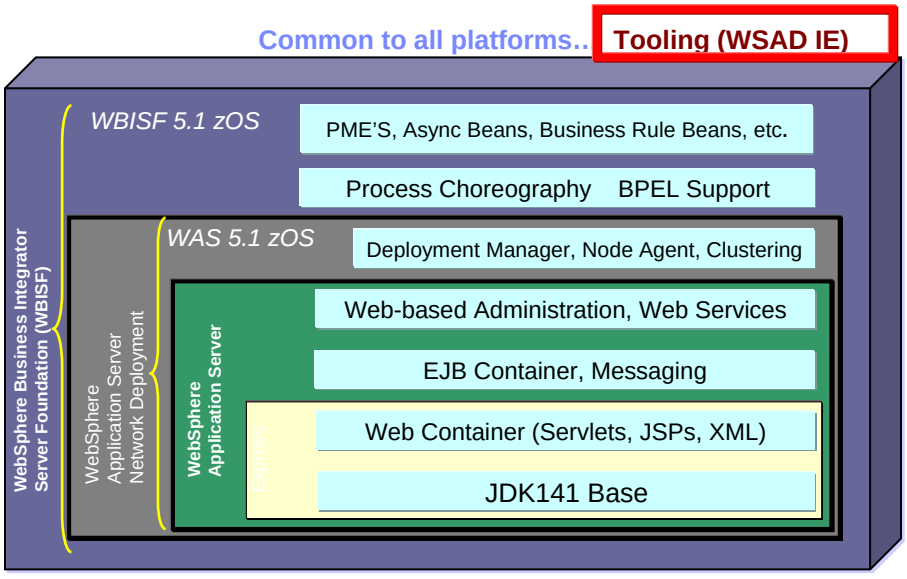
- Increase Development Productivity
  - **Extended Messaging** – quickly create applications that integrate with other systems through a messaging infrastructure.
  - **Internationalization Service** – automatically recognize the calling client's time zone and location information so your application can act appropriately.
  - **Work Areas** – provide a "global variable" like ability to efficiently share information across a distributed application.
  - **Cheat Sheets** – make new or complex tasks easy by providing a checklist for common development patterns.

Programming Model Extensions (PMEs)

- Enable Real-time Application Flexibility
  - **Business Rule Beans** – Offer a powerful real-time framework for defining, executing, and managing business rules that encapsulate business policies that vary based on changes in the business environment. For example, a simple business rule might be, “If a customer's shopping cart is greater than $X, then offer a Y% discount.”
  - **Dynamic Query Service** – Delivers unprecedented application flexibility by allowing you to dynamically build and submit queries that select, sort, join, and perform calculations on application data at runtime. Dynamic query service provides the ability to pass in and process Enterprise Java Beans query language (EJB QL) queries at runtime eliminating the need, as with today's EJB 2.0 standards, to hard-code required queries into the deployment descriptors during development.

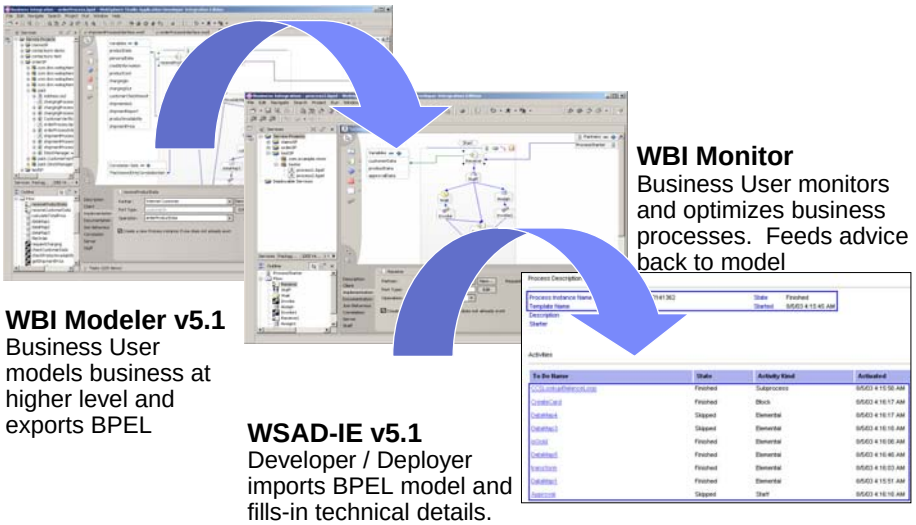
IBM Washington Systems Center

WBISF for z/OS Server Structure



© 2004 IBM Corporation

WBI Tools



WBI Modeler v5.1
Business User models business at higher level and exports BPEL

WSAD-IE v5.1
Developer / Deployer imports BPEL model and fills-in technical details.

WBI Monitor
Business User monitors and optimizes business processes. Feeds advice back to model

Integration Edition in WebSphere Studio Tools Family

Integration Edition

- User:
  - Advanced J2EE Developer
- Function:
  - Services Oriented Architecture
  - Business Process Choreography tools:
    - BPEL and FDML Process editors
    - Services wizards
  - J2C Connectors Tools
  - Connectors for IMS, CICS-ECI/EPI, IMS, HOD
- Target Runtime:

Application Developer

Site Developer

- User:
  - Web Developer
- Function:
  - HTML, JSP, Servlet, JSF, Struts Tools
  - EGL Tools
  - Web Services Tools
  - XML Tools
  - Java Visual Editor

- User:
  - J2EE Developer
- Function:
  - EJB Tools
  - Performance and
  - Trace tools
  - Component Test tools
  - ClearCase LT
- Target Runtime:
  - WAS, WebLogic

Enterprise Developer

- User:
  - Enterprise Developer
- Function:
  - COBOL EGL Tools
  - Legacy z/OS development
  - COBOL, PL/I, ASM Tools
  - COBOL XML ToolsJ2C Connectors Tools
  - Connectors for IMS, CICS-ECI/EPI, IMS, HOD
- Target Runtime:

WebSphere Studio Workbench

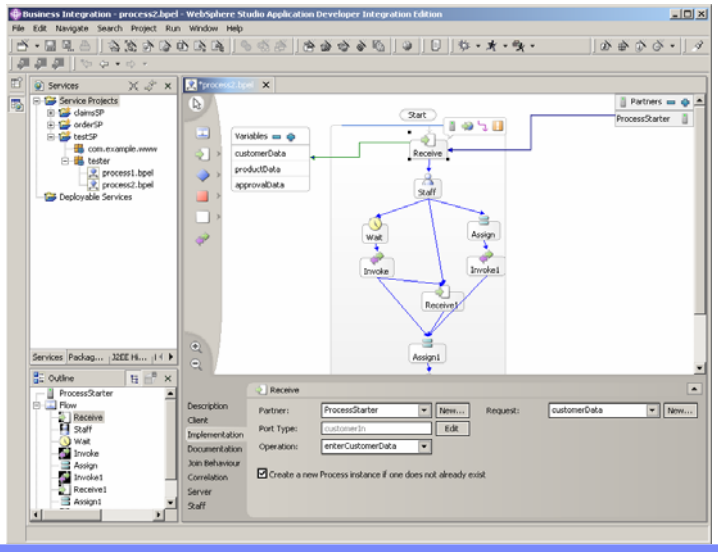
- IBM's commercially supported version of Eclipse Workbench

Eclipse Workbench

- Open Source Universal Tool Platform Initially Developed: core Java IDE, core VCM API/CVS Plug-in, etc...

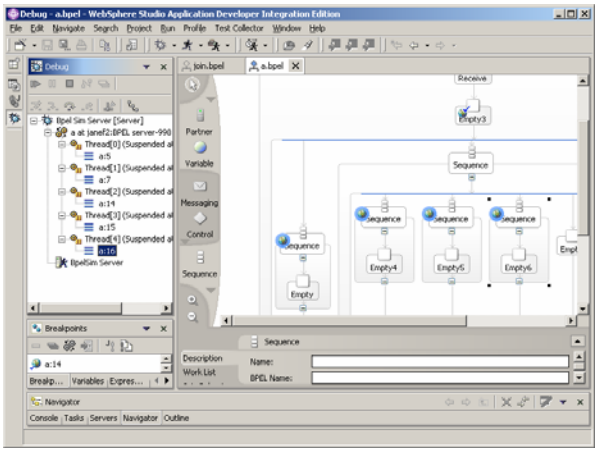
Build BPEL4WS processes in WebSphere Studio

- WebSphere Studio Application Developer Integration Edition V5.1



Process Debugging

- Provides graphical debugging of BPEL4WS processes
  - Inspection and update of variables
  - Breakpoints on control links
  - Single stepping of process execution
  - Ability to step into Java code of conditions and Java snippets

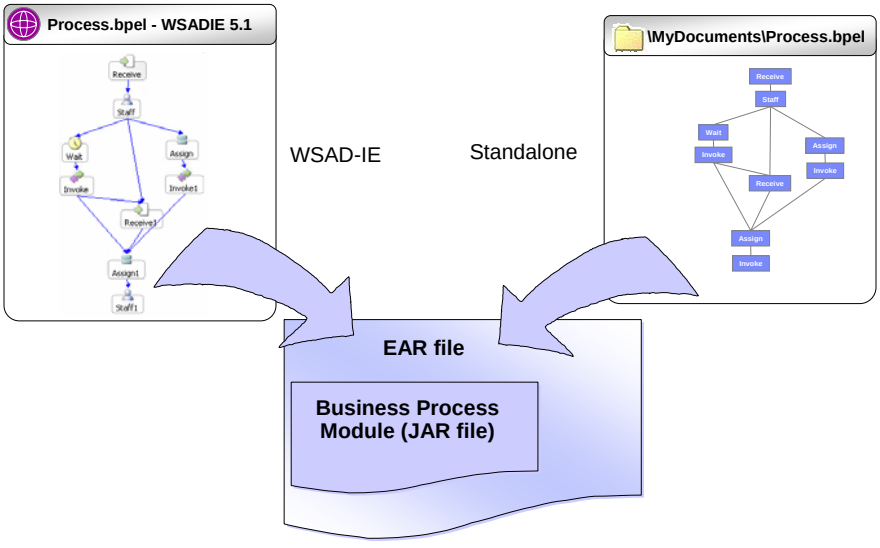


Sample BPEL4WS Code
XML Based
(Created from WSAD IE)
"Partner Link and Partner Link Type"

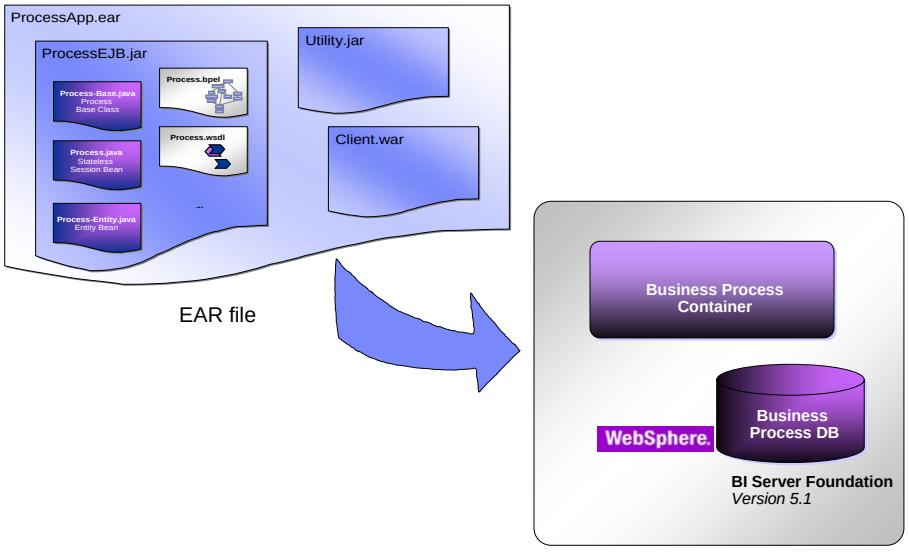
```
<partnerLink name="invoiceProvider"
                partnerLinkType="lns:invoiceLT"
                myRole="invoiceRequester"
                partnerRole="invoiceService"/>
```

```
<plnk:partnerLinkType name="invoiceLT">
        <plnk:role name="invoiceService">
                <portType name="pos:computePricePT"/>
        </plnk:role>
        <plnk:role name="invoiceRequester">
                <portType name="pos:invoiceCallbackPT"/>
        </plnk:role>
</plnk:partnerLinkType>
```

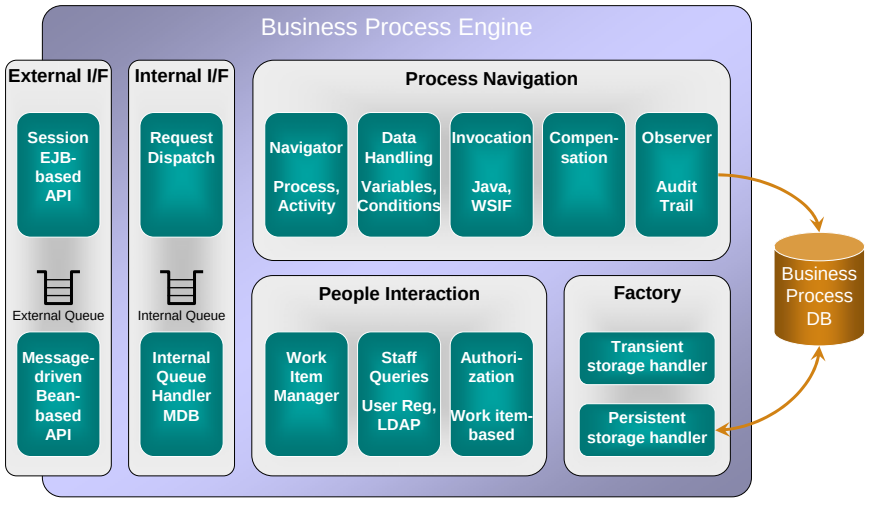

Packaging a Business Process Application



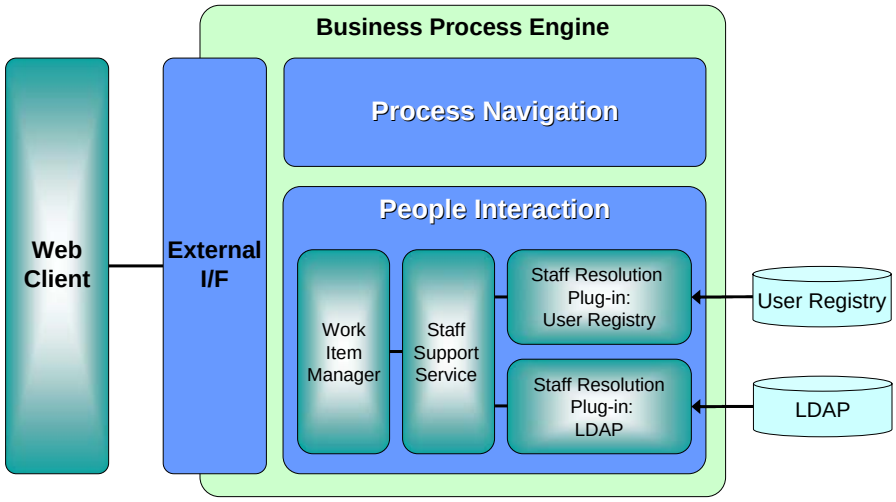
Deployment of the Business Process Application



Process Container

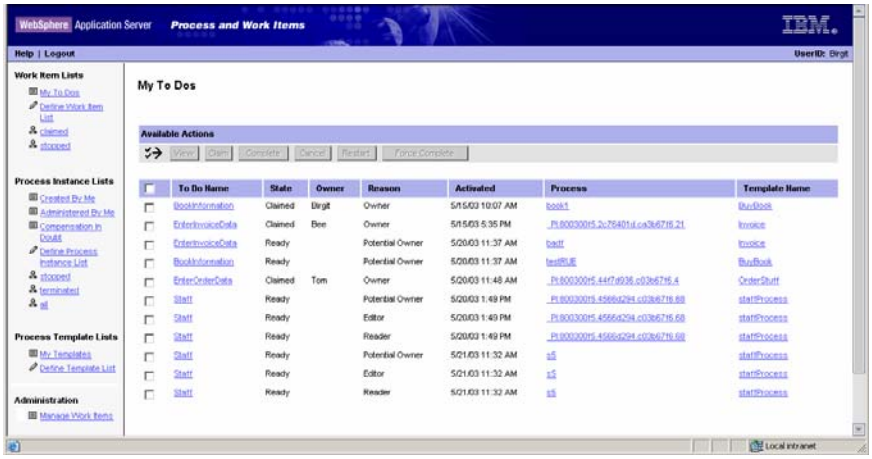


People Interaction Components



Interacting with Process Instances

- Process Choreographer Web Client
  - Start and administer process instances
  - Handling of work items / staff activities



Why Deploy on zSeries?

- **Available: the (zero downtime)/OS brand promise**
  - Consistently delivering expected service regardless of unanticipated workload spikes or failures.
- **Selective: managing resources towards achievement of business goals**
  - The ability to guarantee service levels for specific types of customers and workloads as defined by business needs.
- **Integrated: enabling reuse of existing assets**
  - Composition and integration with multiple z/OS resource managers with optimal performance, better availability, and faster recovery.
- **Secure: maintaining the availability and integrity of resources**
  - The industry's most stringent processes, tools and techniques for access control and asset protection.
- **Efficient: maximizing people and system resources**
  - Lower total cost of ownership through reduction in trained system programmers and fuller utilization of existing capacity.

Why Deploy on zSeries?

- High availability with system clustering – z/OS images may be clustered to construct an environment with no single points of failure.
- z/OS – Robust operating system for e-business – 64 bit z/Architecture
- Manage the workload to optimize performance – Workload Manager (WLM) for z/OS
- TCP/IP networking enhancements – Virtual IP Address (VIPA) and Sysplex Distributor

zSeries is at the Forefront of Autonomic Computing

- **z/OS Self-Optimizing capabilities are highly leveraged by WebSphere**
  - Server resources are automatically shared among a variety of workloads.
  - Automatic resource allocation crosses OS images and physical servers.
  - Work can be differentiated and prioritized across a Parallel Sysplex based on service level requirements.

- **z/OS Self-Healing capabilities are integral to WebSphere**
  - A zSeries server can survive a CP failure without loss of availability of the system.
  - A middleware software subsystem outage can be survived without loss of availability of the system.
  - Capacity can be dynamically added in response to unexpected spikes in the workload.
  - OS and middleware software can be dynamically upgraded without loss of availability of the system.

WebSphere for z/OS V5 – Connectors

- **IMS 7.1 /8.0 IMS Connect 1.2 /2.0**
  - APPC based connector - RRS enabled
  - Local (OTMA based) - RRS enabled
  - Remote (TCPIP based) - 1PC capable
    - Will supercede APPC based connector once 2PC capable
- **CICS CTG 5.0.1**
  - Local (EXCI based) - 2PC capable
  - Remote (TCPIP based) - 1PC capable
  - Remote (ECI based) - 1PC capable
- **JDBC to DL/I and VSAM**
- **WebSphere will support any fully compliant JCA 1.0 Connector**
- **DB2**
  - Local 2PC capable
  - High performance RRS Enabled
- **XA Transaction Coordination**
  - New in V5
  - RRS and XA resources in a single transaction

Summary

- Process Choreographer is a business process workflow engine
- It supports the BPEL4WS open standard
- Processes can encapsulate Web services, Enterprise Information Systems, messaging, J2EE components, adapters, connectors, and many more components
- Development tool is WebSphere Studio Application Developer Integration Edition V5.1
- Runtime tool is WebSphere Business Integration Server Foundation V5.1
- Running WBISF v5.1 on z/OS brings many added benefits.